Deep Generative Model *for* Text Generation

李磊　周浩

ByteDance AI Lab
字节跳动人工智能实验室

"*What I cannot create, I do not understand*"

–Richard Feynman

Outline

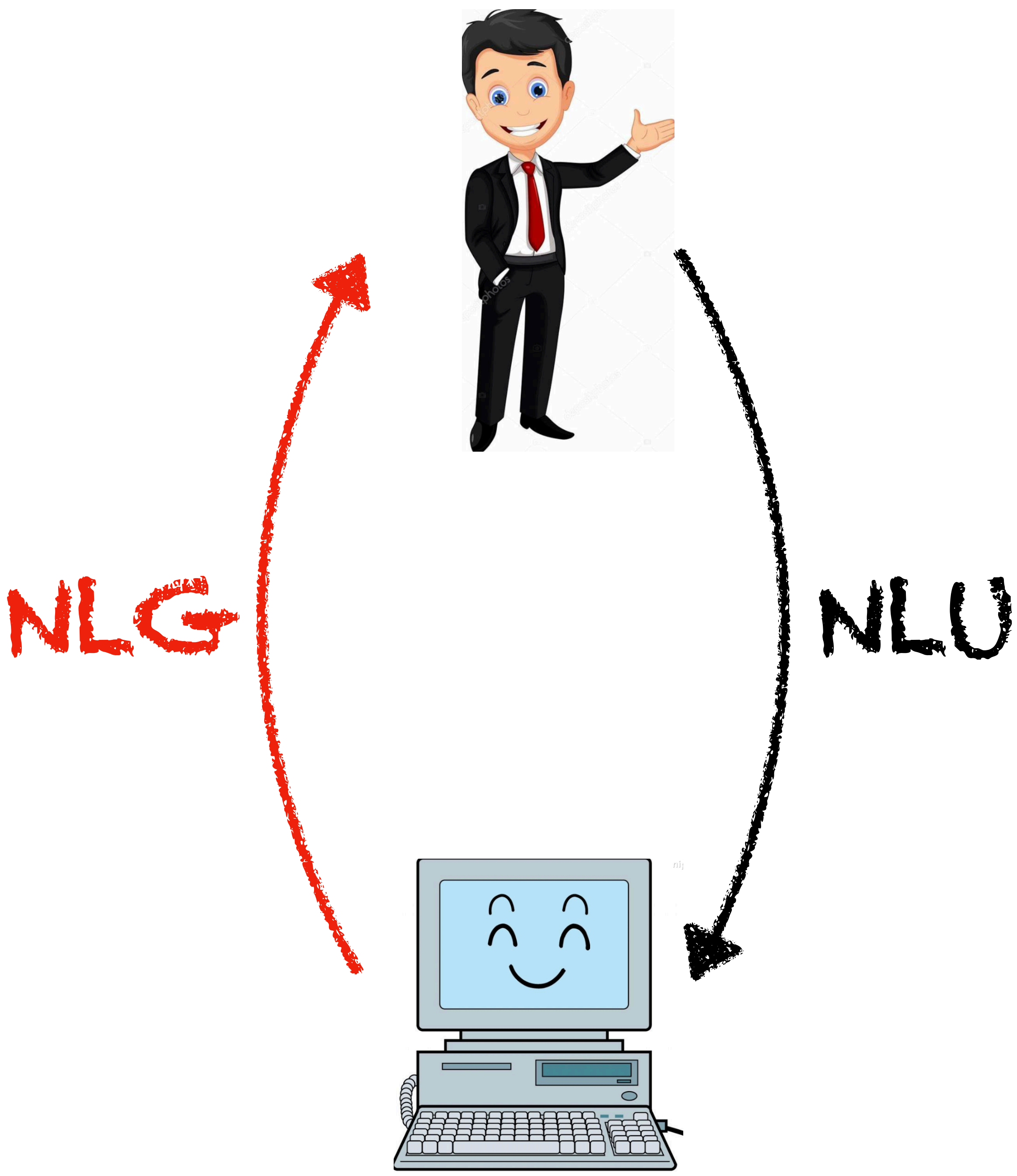
- Motivation
  1. Text Generation is Crucial but Non-trivial
- Taxonomy of deep generative models
  2. Explicit Density
     ① Density Decomposition
     ② Approximation by Variational Inference
  3. Implicit Density
     ③ Constrained Generation by Metropolis Hastings
     ④ Generative Adversarial Networks
- Conclusion

Part 1

Motivation

Why we need to study Text Generation

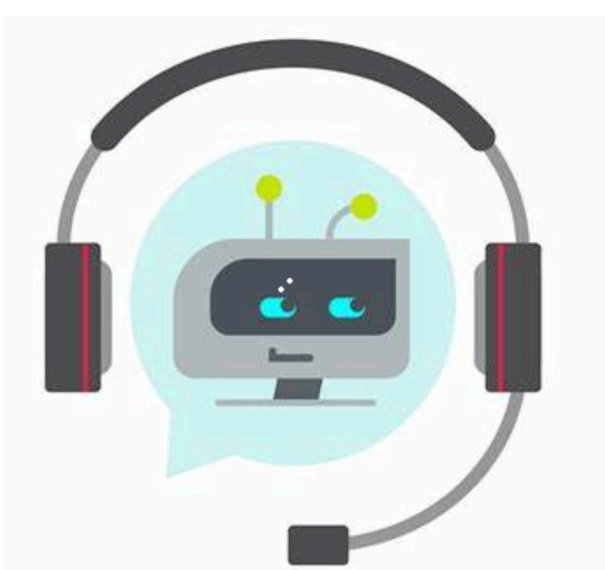
Text Generation is Important!



Natural language generation is an **indispensable** part of human-computer interaction!

Text Generation is Widely Used

Machine Translation



ChatBOT

Question Answering

Text Generation is Non-Trivial

Maximum Likelihood Estimation:

$$min\ \mathbb{E}_{x \sim p_{data}}[-log\ p_\theta(x)]$$

$$p_\theta(x') = \frac{\sigma(x')}{\sum_x \sigma(x)}$$

Text Generation is Non-Trivial

Maximum Likelihood Estimation:

$$min\ \mathbb{E}_{x \sim p_{data}}[-log\ p_\theta(x)]$$

$$p_\theta(x') = \frac{\sigma(x')}{\boxed{\sum_x \sigma(x)}}$$

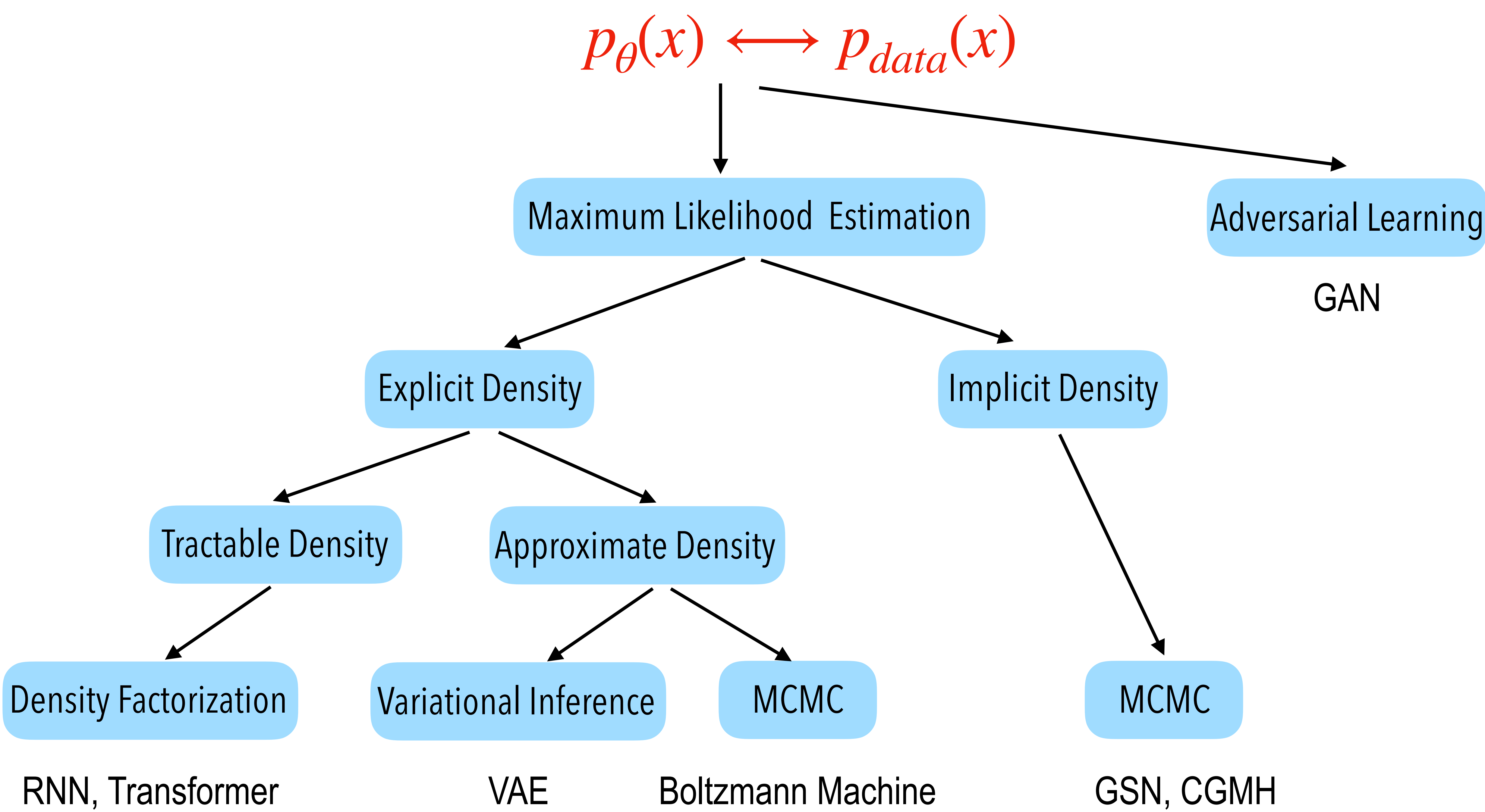
Partition function is **exponential**, intractable for computing.

Part 2

Taxonomy

Different Branches of Deep Generative Models for Text Generation

Taxonomy of DGM

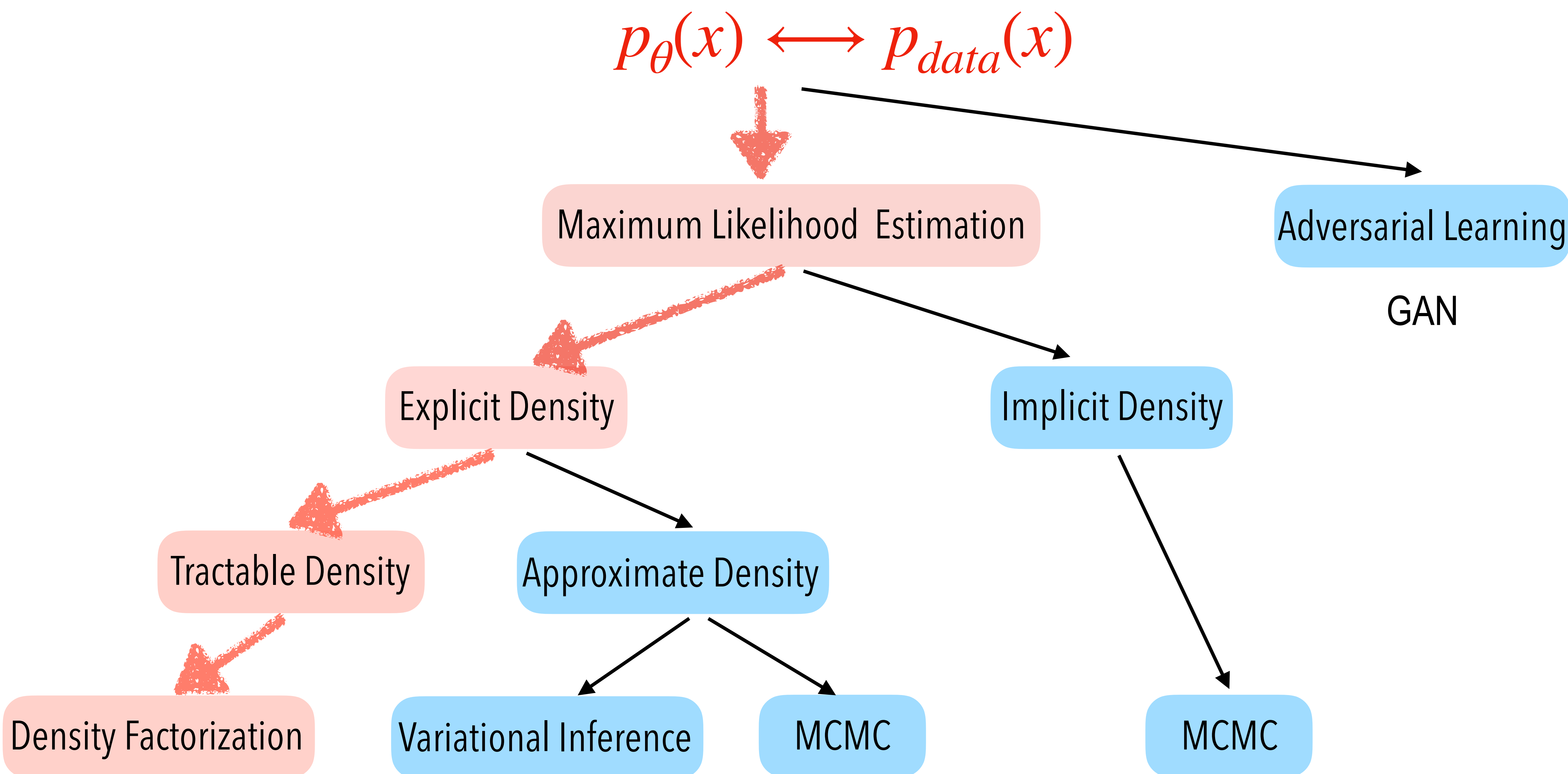


Part 3

Text Generation by Density Decomposition

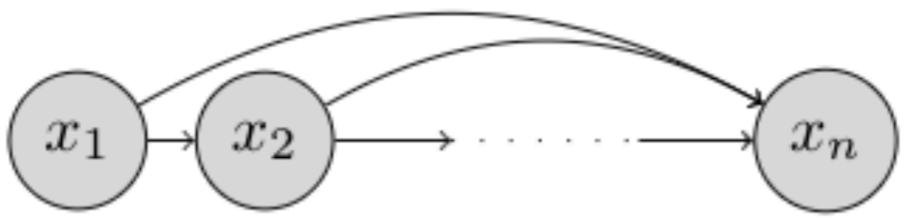
Decompose the joint distribution as a product of tractable conditionals.

Generation by Decomposition



Tractable Density by Factorization

- Directed, fully-observed graphical models:



Decompose the joint distribution as a product of tractable conditionals:

Given $x = [x_1, x_2, x_3 \ldots, x_n]$

$$p_\theta = \prod_{i=1}^{n} p_\theta(x_i \,|\, x_1, x_2, \ldots, x_{i-1}) = \prod_{i=1}^{n} p_\theta(x_i \,|\, x_{<i})$$

Parameterization by Neural Networks

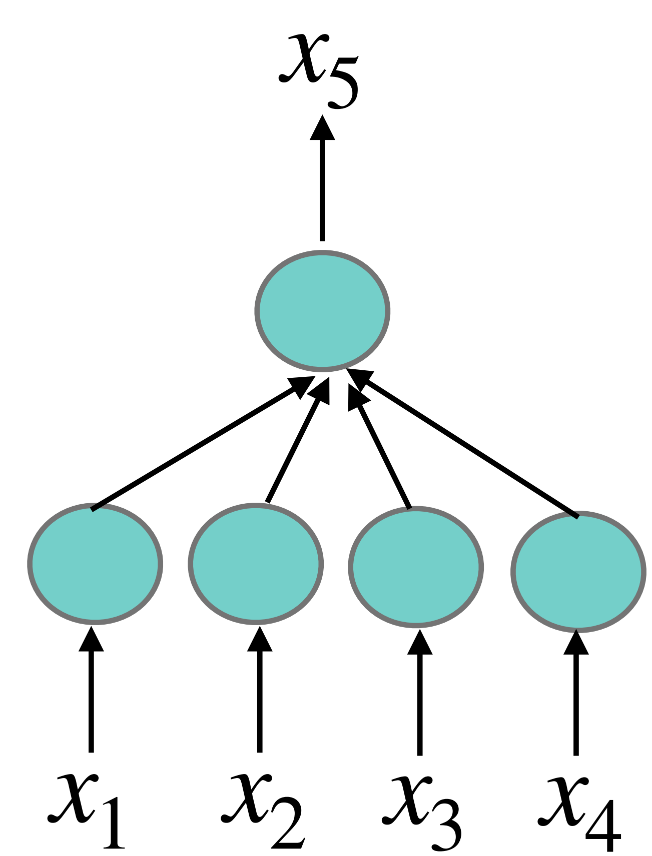
$$p_\theta(x_i' \,|\, x_{<i}) = \frac{\sigma(x_i')}{\sum_{x_i'} \sigma(x_i \,|\, x_1 \ldots x_{i-1})}$$

Vocabulary Size

 Tractable for computing

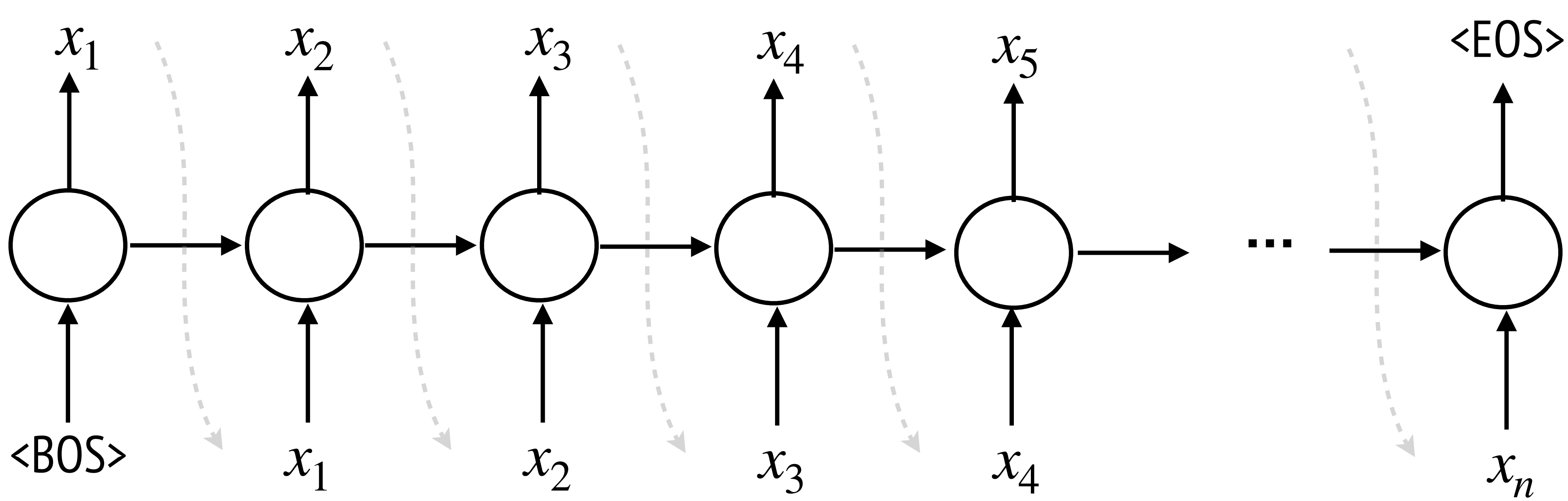
Parameterization by Neural Networks

$$p_\theta(x_5 \mid x_1, x_2, x_3, x_4)$$



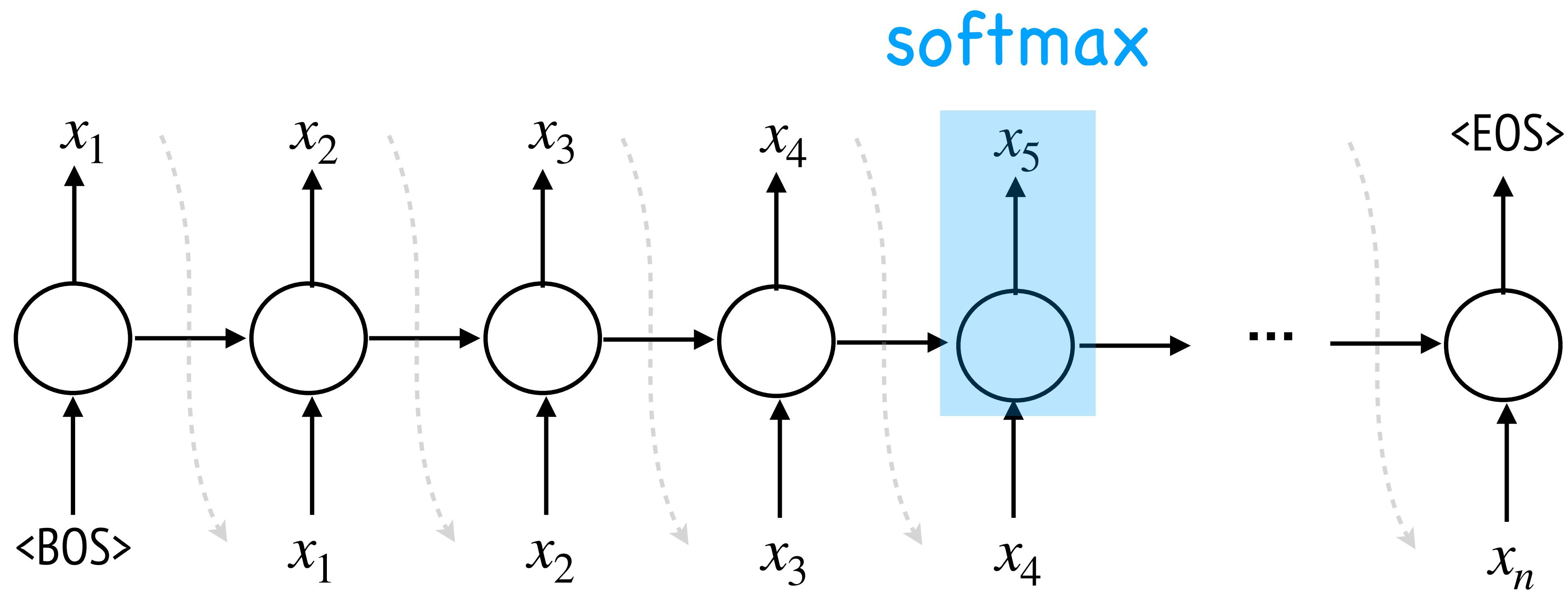
Parameterization by Neural Networks

$$p_\theta(x_i \mid x_{<i})$$



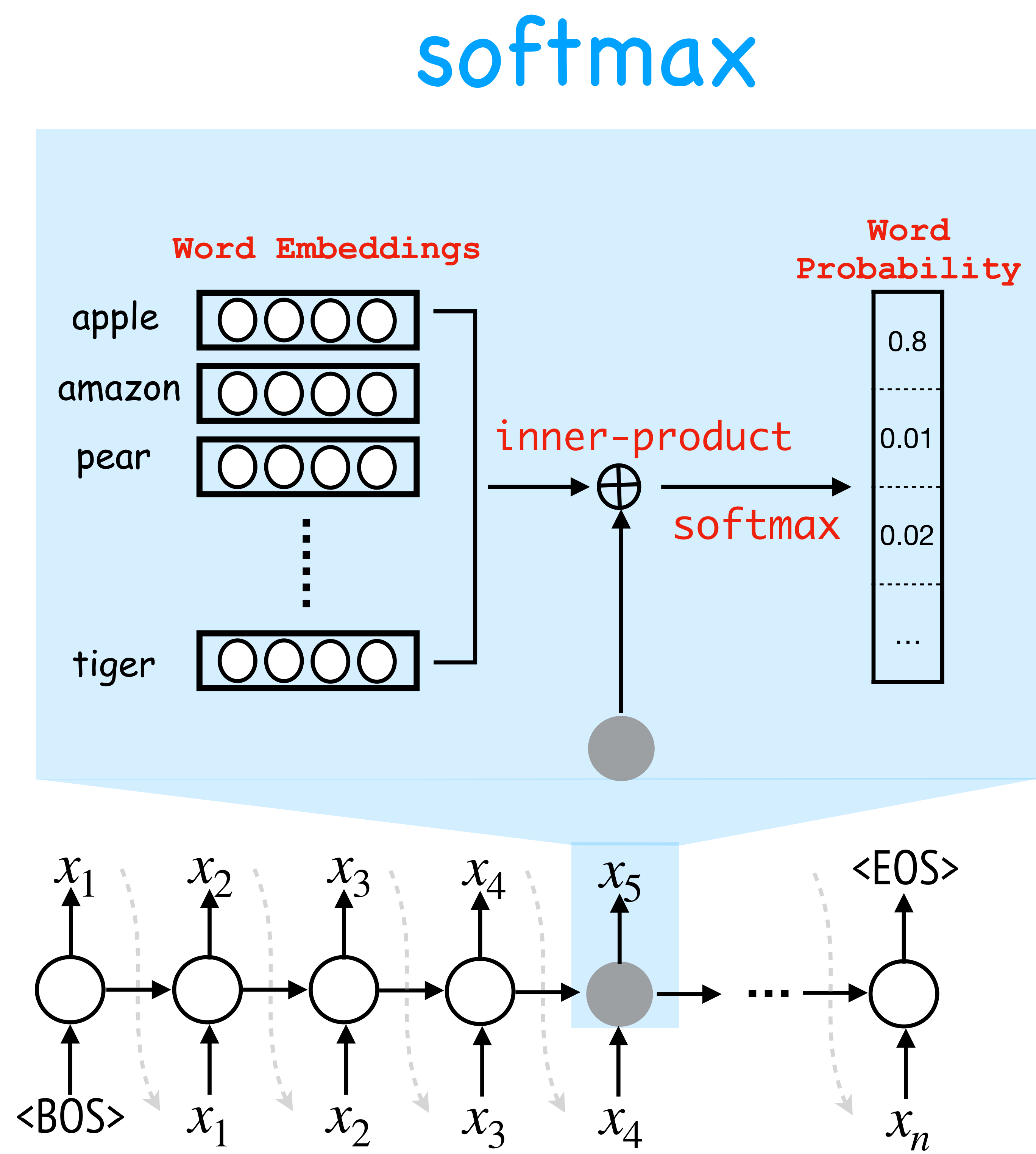
Parameterization by Neural Networks

$$p_\theta(x_i \mid x_{<i})$$



Parameterization by Neural Networks

$$p_\theta(x_i \mid x_{<i})$$



Model

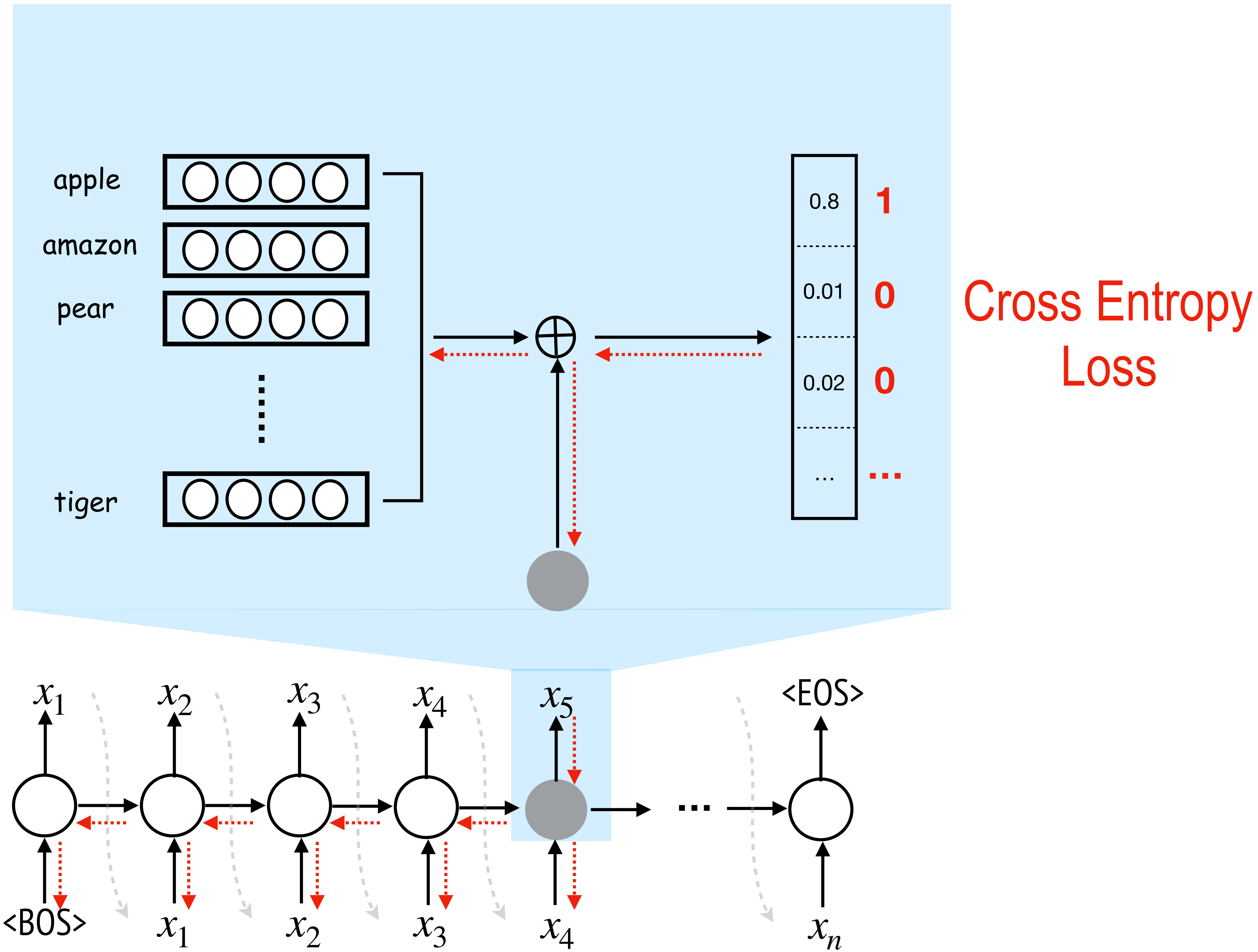
Maximum Likelihood Estimation:

$$min \ \mathbb{E}_{x \sim p_{data}}[-log \ p_\theta(x)]$$

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i \,|\, x_1, x_2, \ldots, x_{i-1}) = \prod_{i=1}^{n} p_\theta(x_i \,|\, x_{<i})$$

Parameterization by RNN

BackPropagation by MLE



Conditional

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

Conditional

$$p_\theta(\boxed{x} \mid \boxed{y})$$

Output Input

Conditional

$$p_\theta(\boxed{x} \mid \boxed{y})$$

Output Input

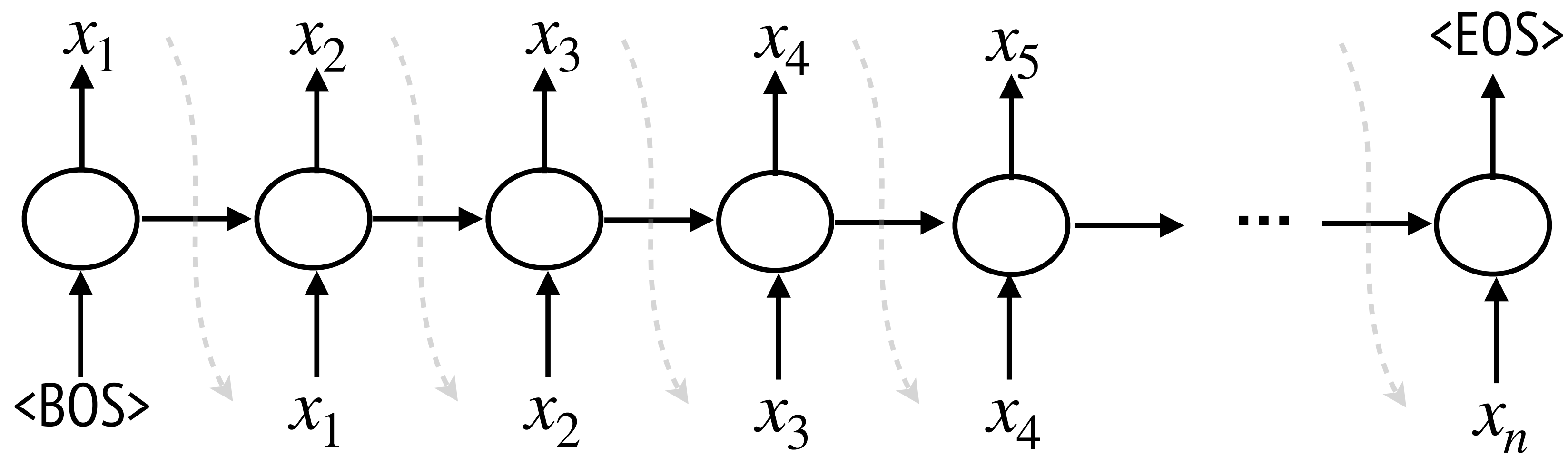
<u>Maximum Likelihood Estimation:</u>

$$min\ \mathbb{E}_{x \sim p_{data}}[-log\ p_\theta(x \mid y)]$$

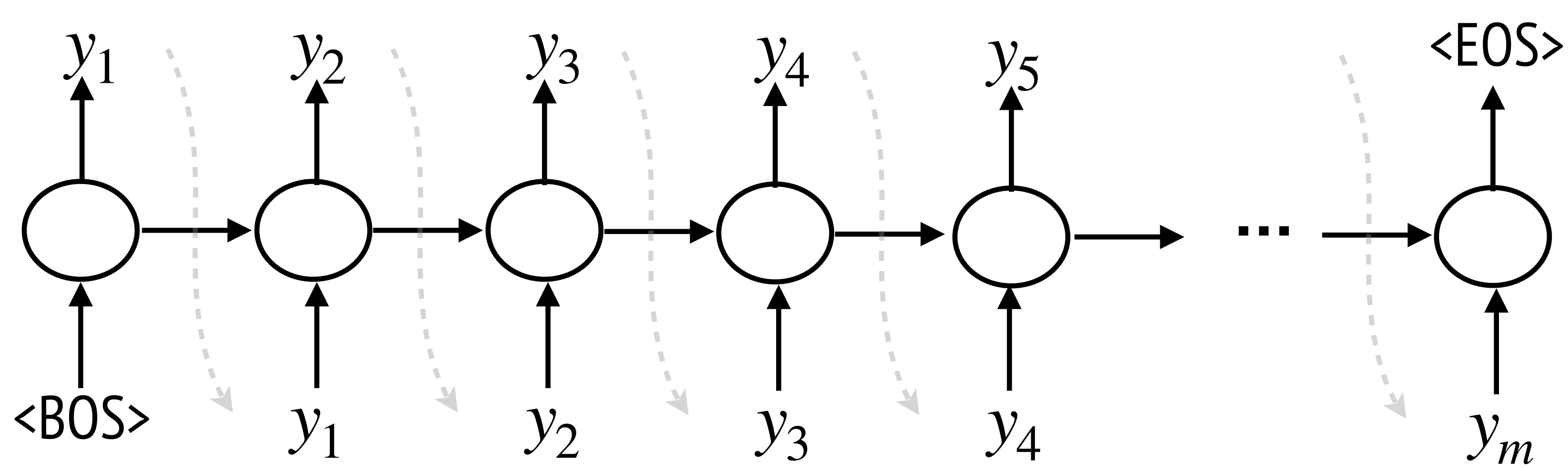
$$p_\theta(x \mid y) = \prod_{i=1}^{n} p_\theta(x_i \mid x_1, x_2, \ldots, x_{i-1}, y) = \prod_{i=1}^{n} p_\theta(x_i \mid x_{<i}, y)$$

Conditional

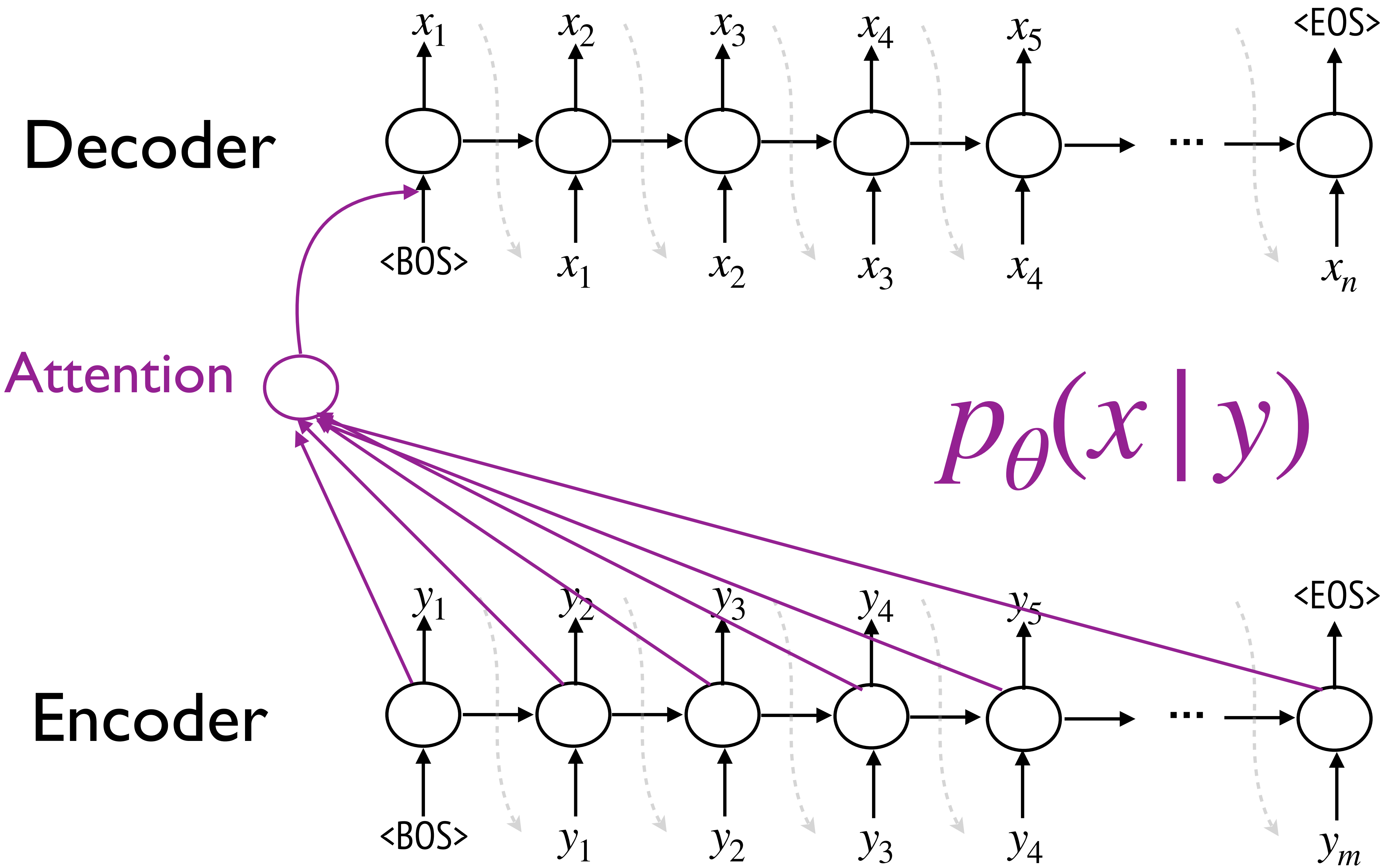
Decoder



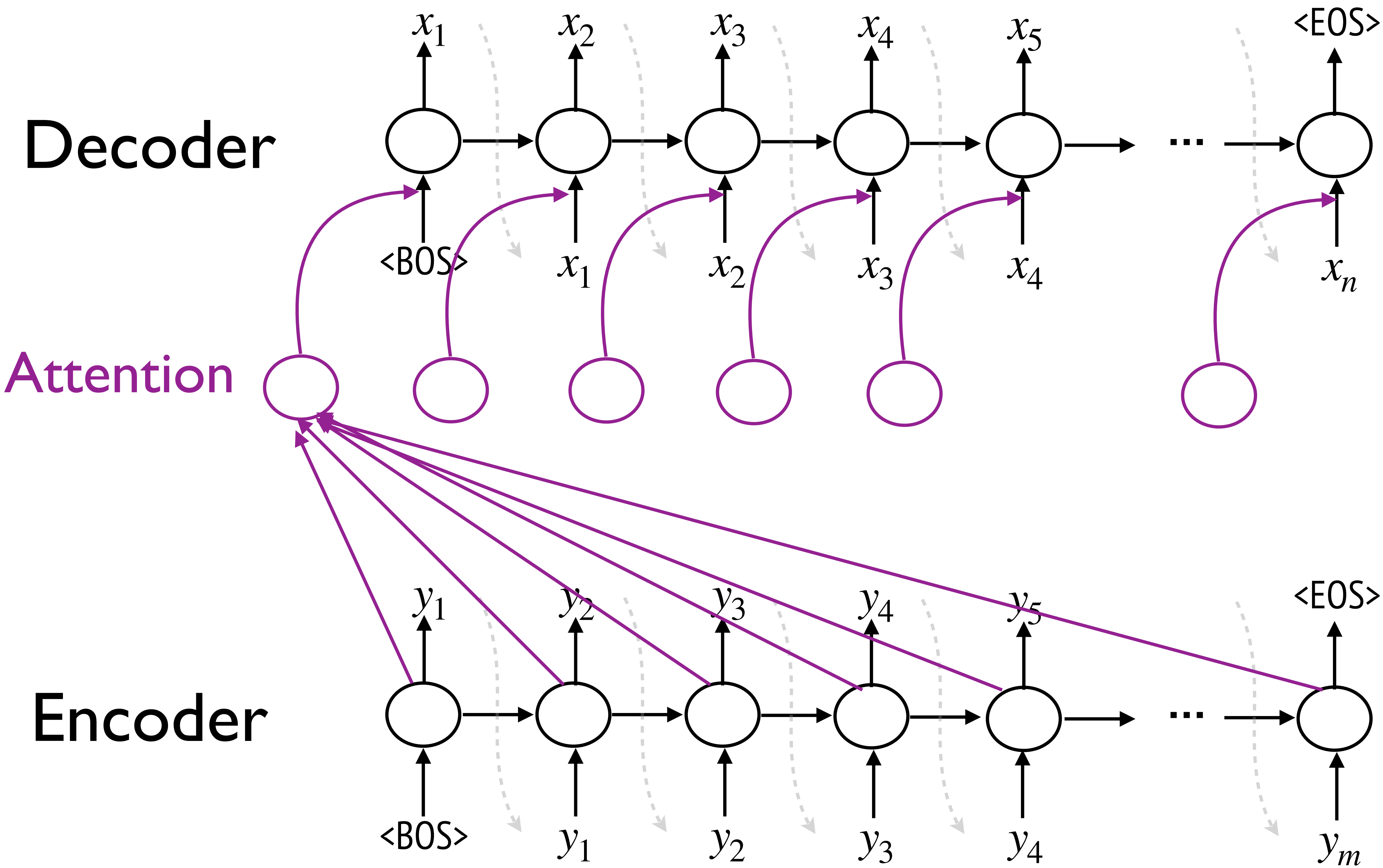
Encoder



Conditional



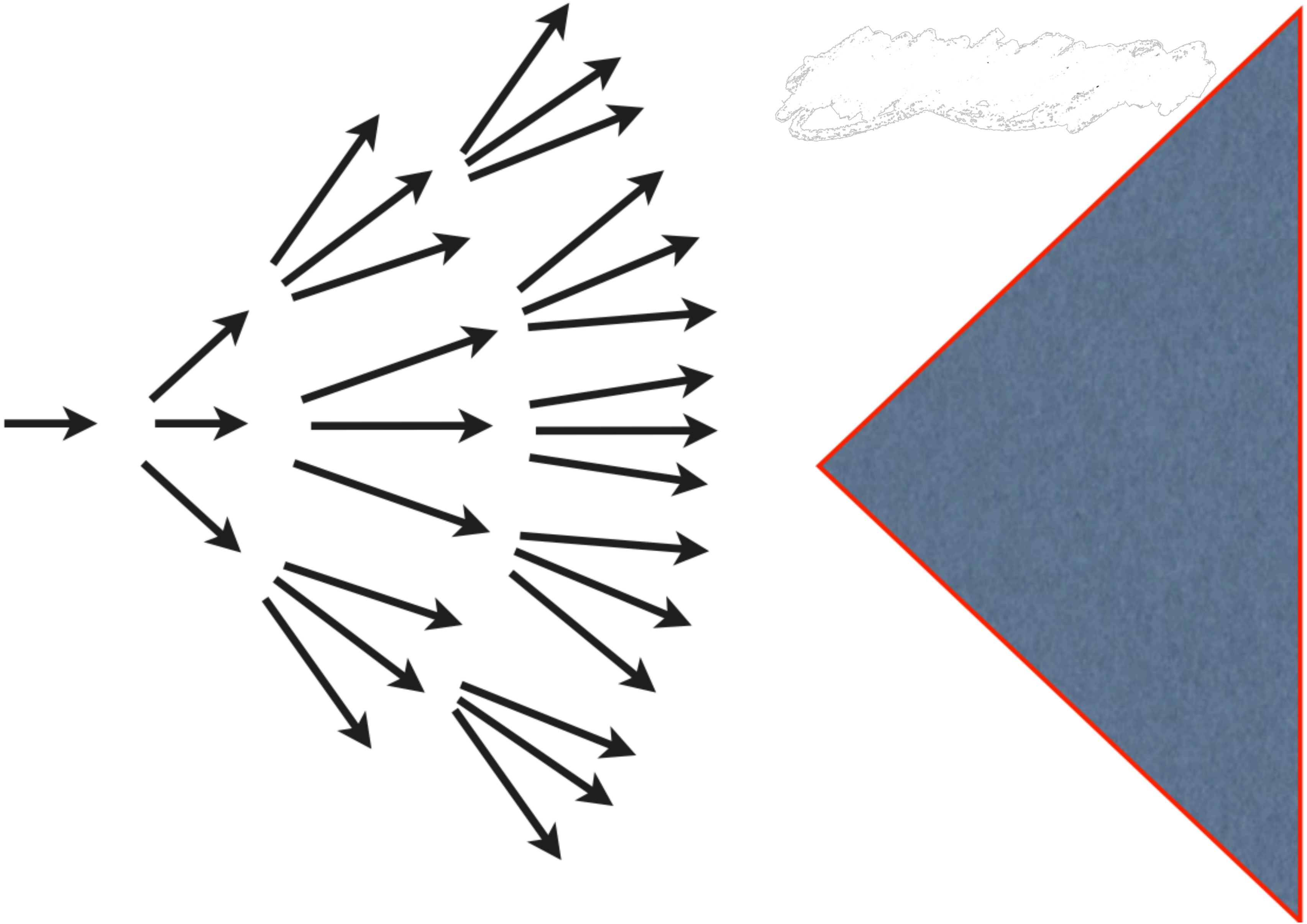
Conditional



Decoding

$$\boxed{\log p_\theta(x \mid y) = \sum_{i=1}^{n} \log p_\theta(x_i \mid x_1, x_2, \ldots, x_{i-1}, y) = \sum_{i=1}^{n} \log p_\theta(x_i \mid x_{<i}, y)}$$

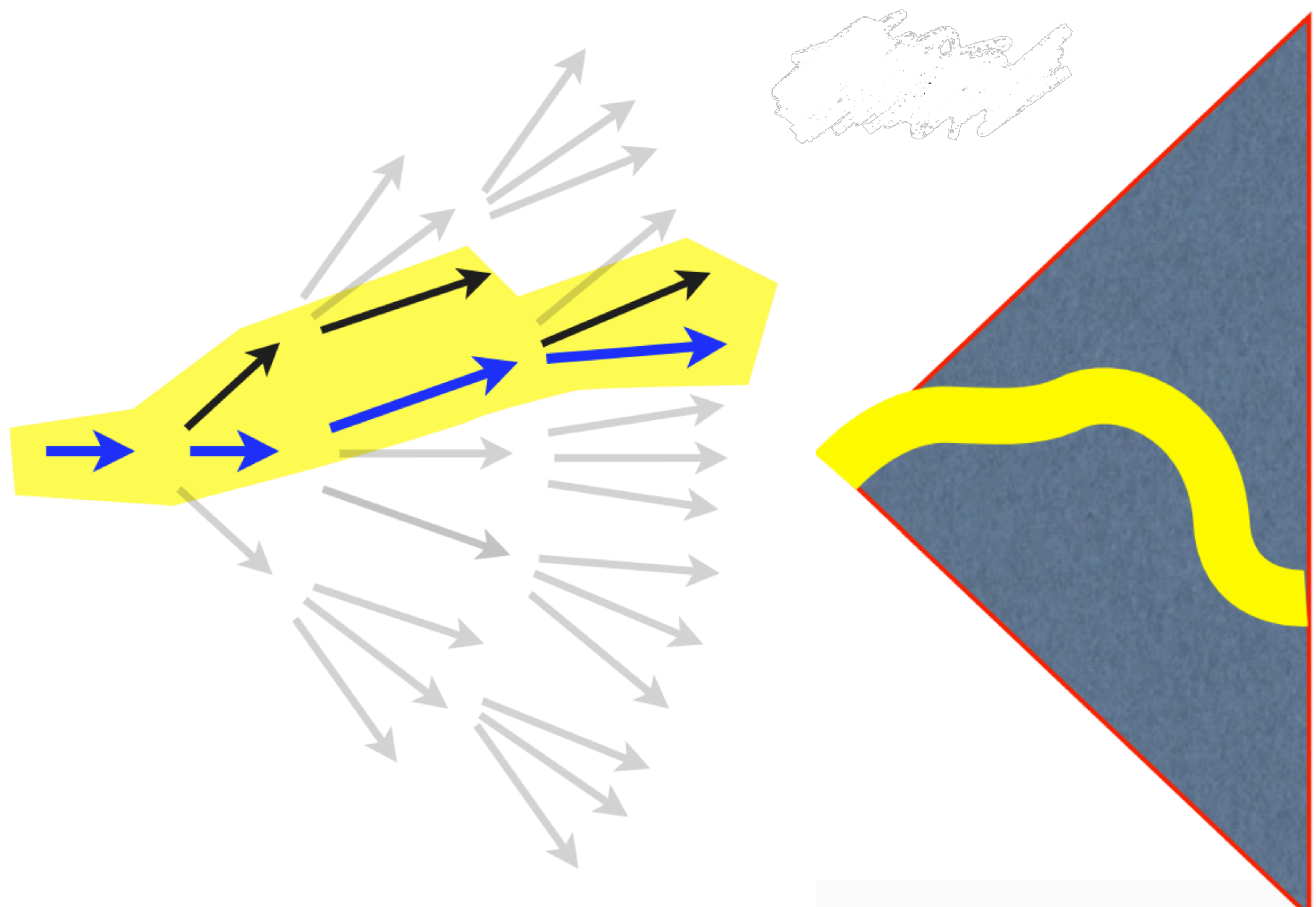
Decoding space is still exponential



Beam Search

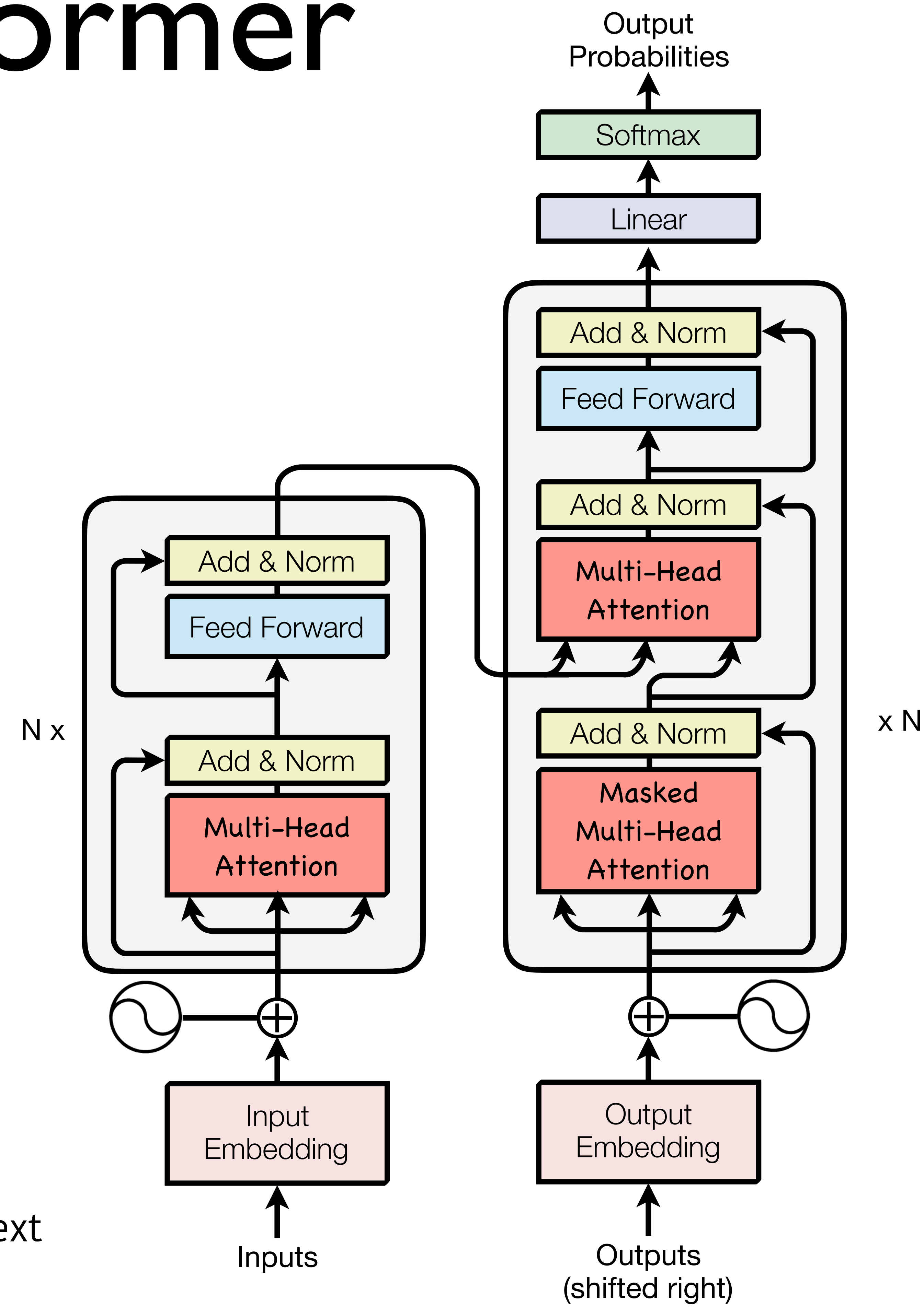
$$\log p_\theta(x \mid y) = \sum_{i=1}^{n} \log p_\theta(x_i \mid x_1, x_2, \ldots, x_{i-1}, y) = \sum_{i=1}^{n} \log p_\theta(x_i \mid x_{<i}, y)$$

Heuristic search
by beam search



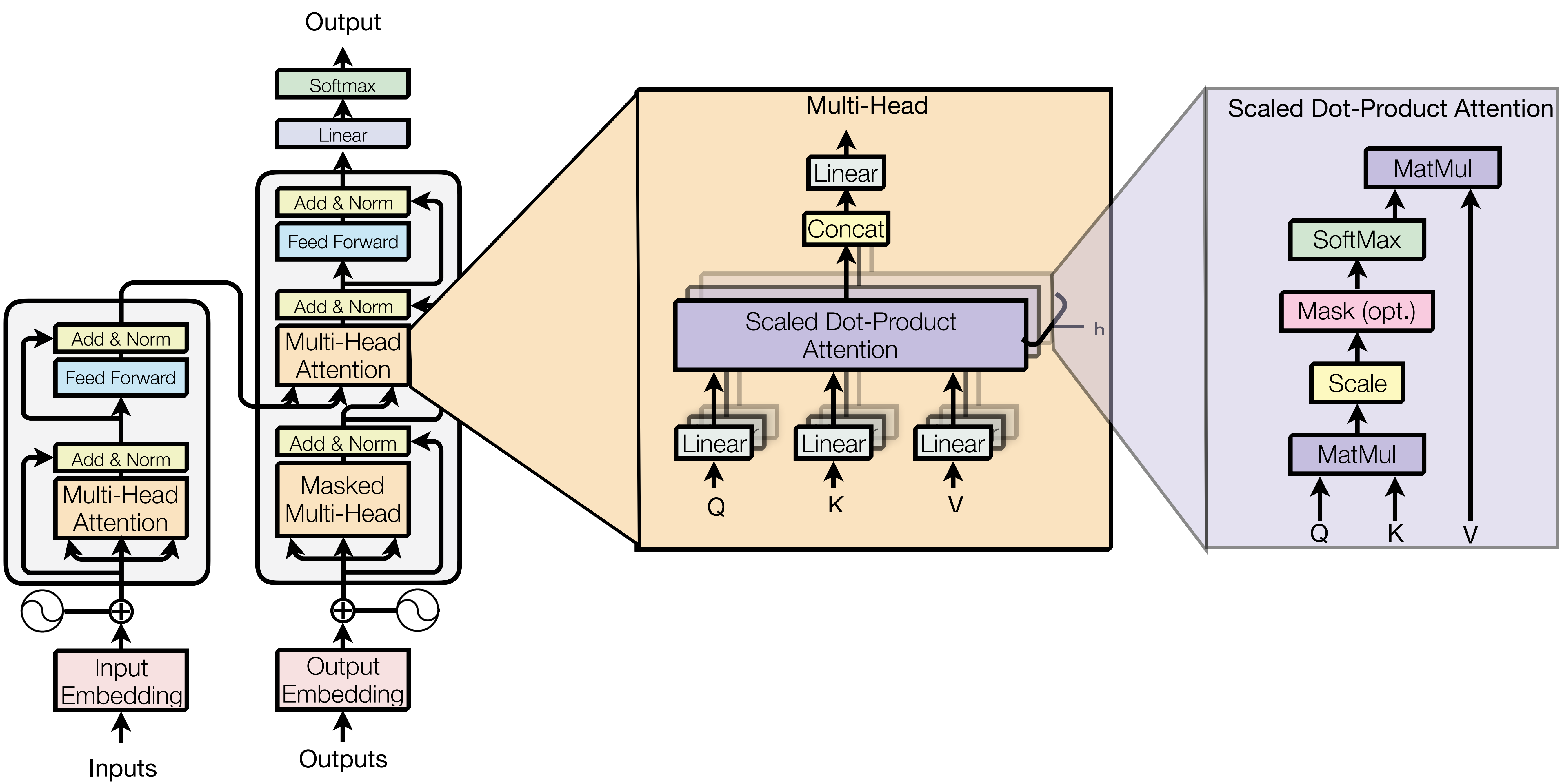
Transformer

Transformer abandon RNN by using Self-Attention!



Vaswani et al., Kernelized Bayesian Softmax for Text Generation, in NIPS, 2017.

Multi-Head Attention



Kernelized Bayesian Softmax

KerBS: Kernelized Bayesian Softmax

$$P(x_t = i) = \sum_{j \in 0,1,\ldots,N_i} P(x_t = s_i^j)$$

where $$P(x_t = s_i^j) = \frac{\exp(\mathcal{K}_{\theta_i^j}(h_t, w_i^j))}{\sum_k \sum_{r \in 0,1,\ldots,N_k} \exp(\mathcal{K}_{\theta_k^r}(h_t, w_k^r))}$$

$$\mathcal{K}_\theta(h, e) = |h||e|(a \exp(-\theta \, cos(h, e)) - a)$$

Here h is hidden state, e is embedding, θ is a parameter controlling the embedding variances of each sense and $a = \dfrac{-\theta}{2(\exp(-\theta) + \theta - 1))}$ is a normalization factor.

Miao et al., Kernelized Bayesian Softmax for Text Generation, in NeurIPS, 2019.

Why KerBS?

Model capacity of softmax is not OK 

	Word2Vec	BERT
Category	Context Independent	Context Dependent
Capacity	Low	High
Performance	Bad	Good

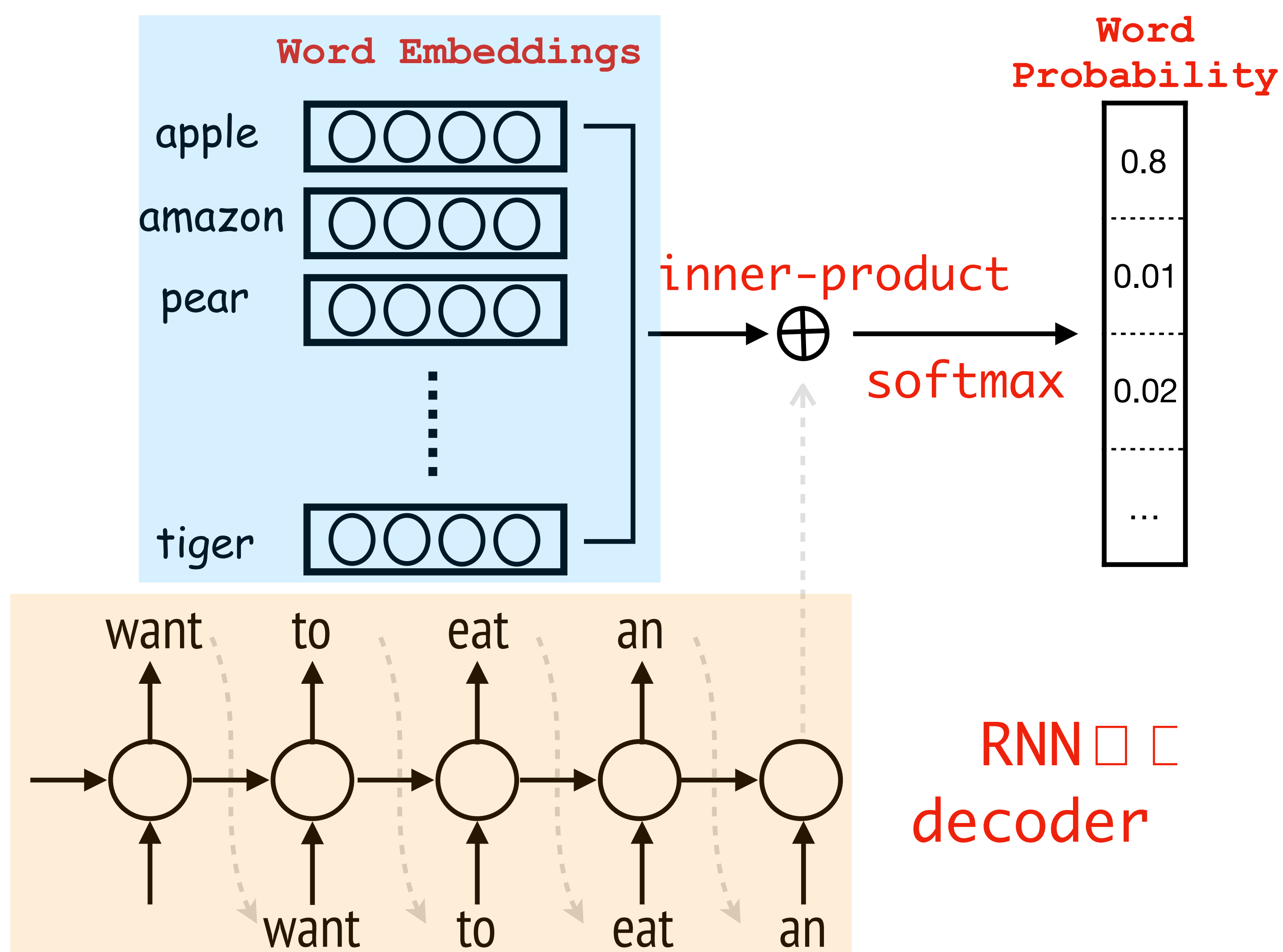
Motivated by BERT, we may need context dependent embedding for text generation!

Text Generation as Matching

Text Generation is Embedding Matching

Context Independent Embedding

Context Dependent Embedding



RNN□ □
decoder

庄子：吾生也有涯，而知也无涯。以有涯随无涯，殆已！

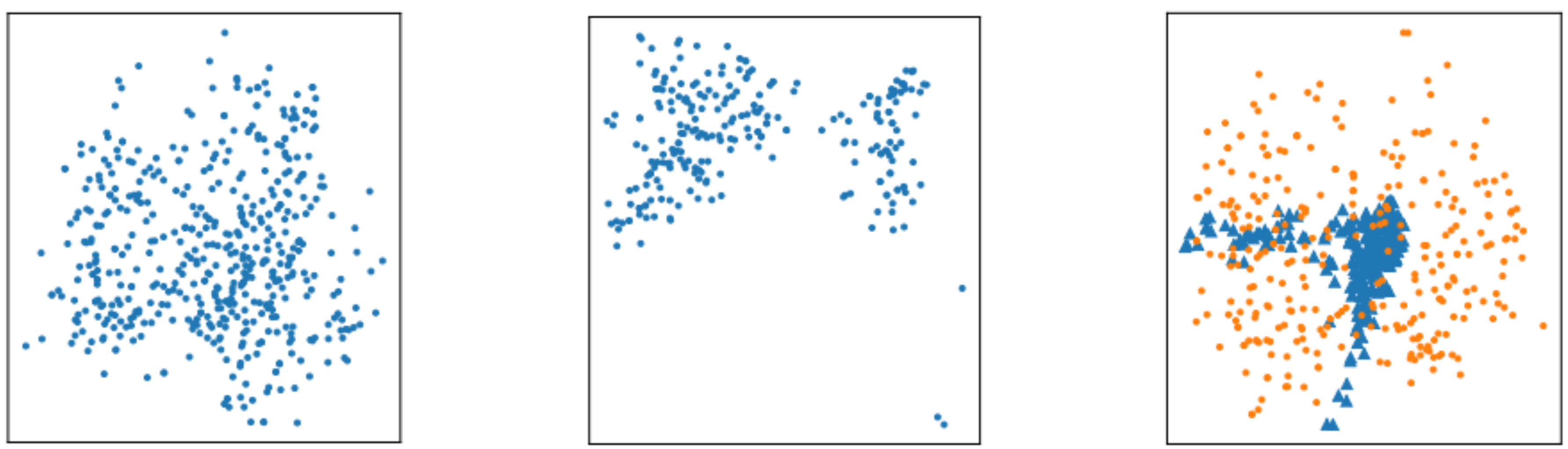
Bottleneck of Text Generation

$\boxed{\text{Bottleneck of text generation is the softmax}}$

Embedding matrix in softmax should have larger capacity.

Visualization of BERT

• Multi-Sense & Varying Variances

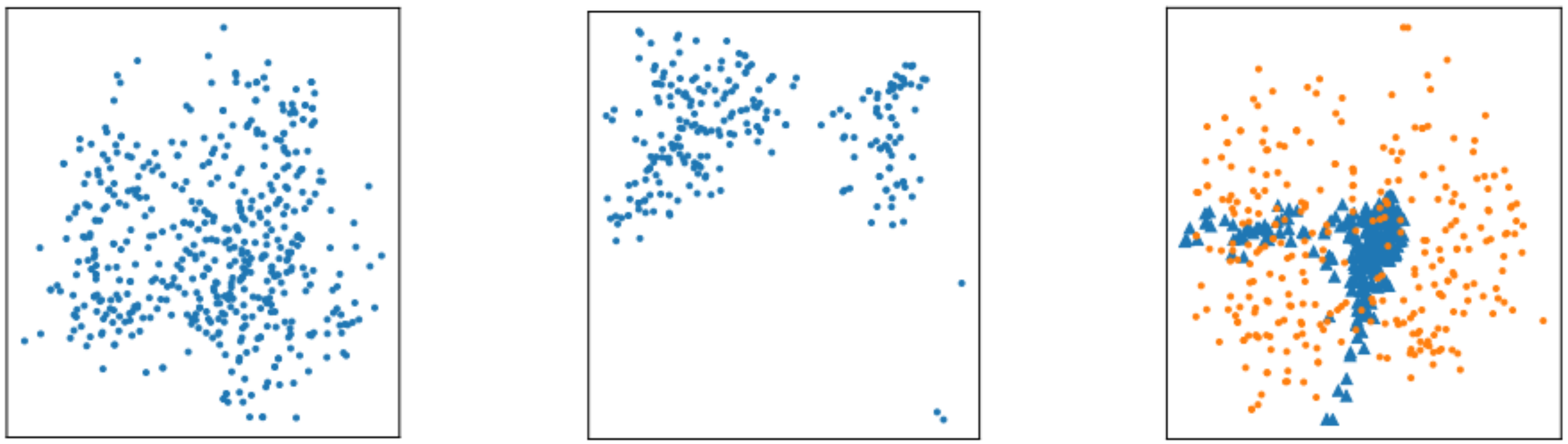


(a) computer (b) monitor (c) car and vehicle

Softmax ***can*** handle this situation

Visualization of BERT

- Multi-Sense & Varying Variances

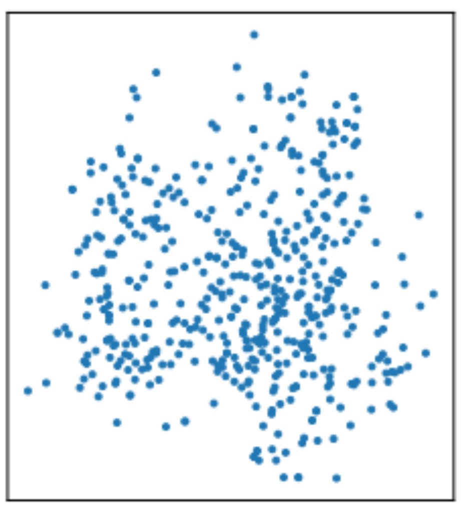


(a) computer (b) monitor (c) car and vehicle

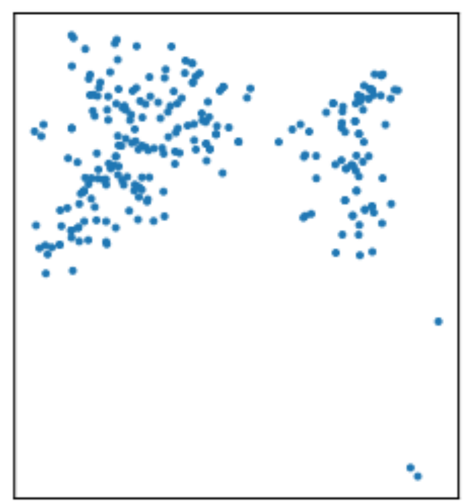
Softmax **can't** handle **multisense**.

Visualization of BERT

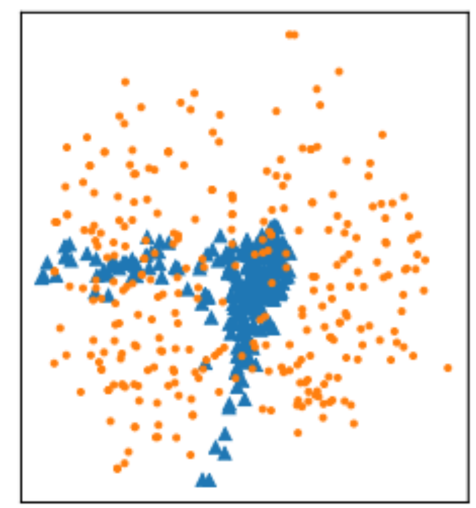
- Multi-Sense & Varying Variances



(a) computer



(b) monitor

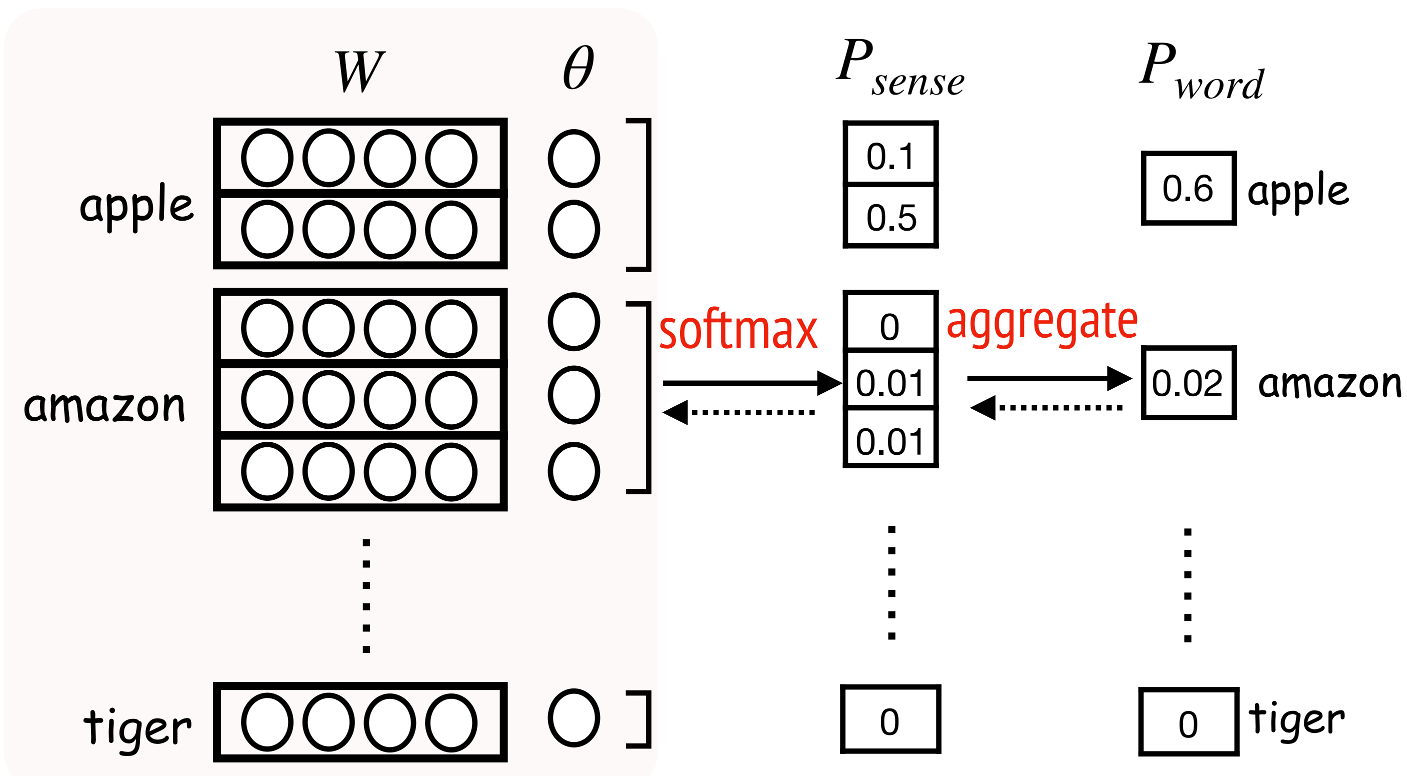


(c) car and vehicle

Softmax **can't** handle **multisense** and varying **variance**s.

KerBS - Multisense

Each word may have several senses. KerBS allocates a vector for each sense.



KerBS - Multisense

After getting the probabilities of each sense, KerBS sums up all sense probabilities of same word.

$$P(x_t = i) = \sum_{j \in 0,1,\ldots,N_i} P(x_t = s_i^j)$$

KerBS - Varying Variances

The distribution of each word's output vectors have different variances.

We use a variable kernel to represent varying variances.

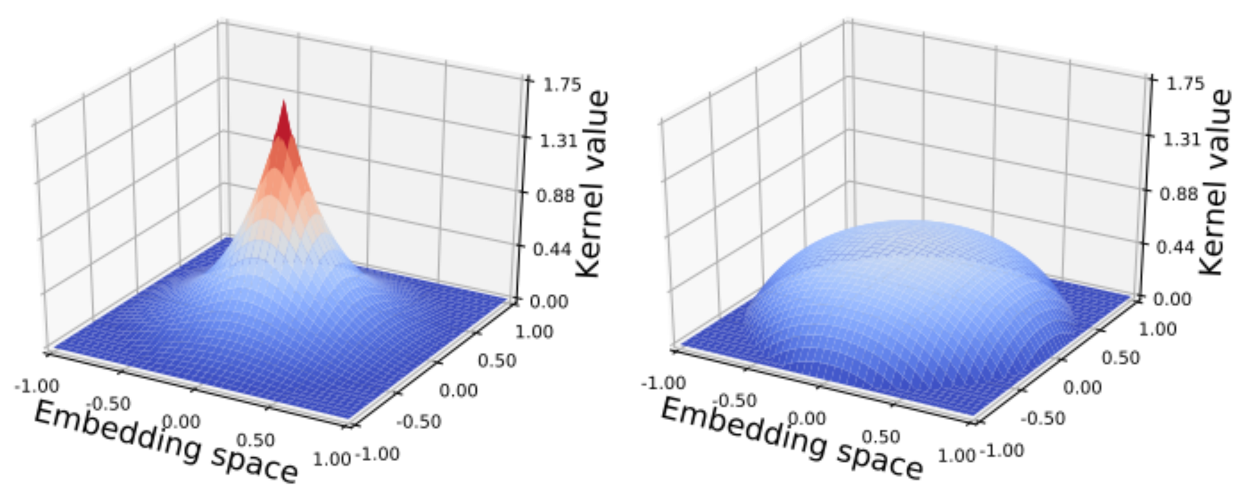
$$P(x_t = s_i^j) = \frac{\exp(\mathcal{K}_{\theta_i^j}(h_t, w_i^j))}{\sum_k \sum_{r \in 0,1,\ldots,N_k} \exp(\mathcal{K}_{\theta_k^r}(h_t, w_k^r))}$$

$$\mathcal{K}_\theta(h, e) = |h||e|(a\exp(-\theta\ cos(h, e)) - a)$$

Note that when $\theta \to 0, \mathcal{K}_\theta(h, e) \to |h||e|\ cos(h, e)$, which is regular Euclidean norm!

KerBS - Varying Variances

The distribution of each word's output vectors have different variances. We use a variable kernel to represent varying variances.



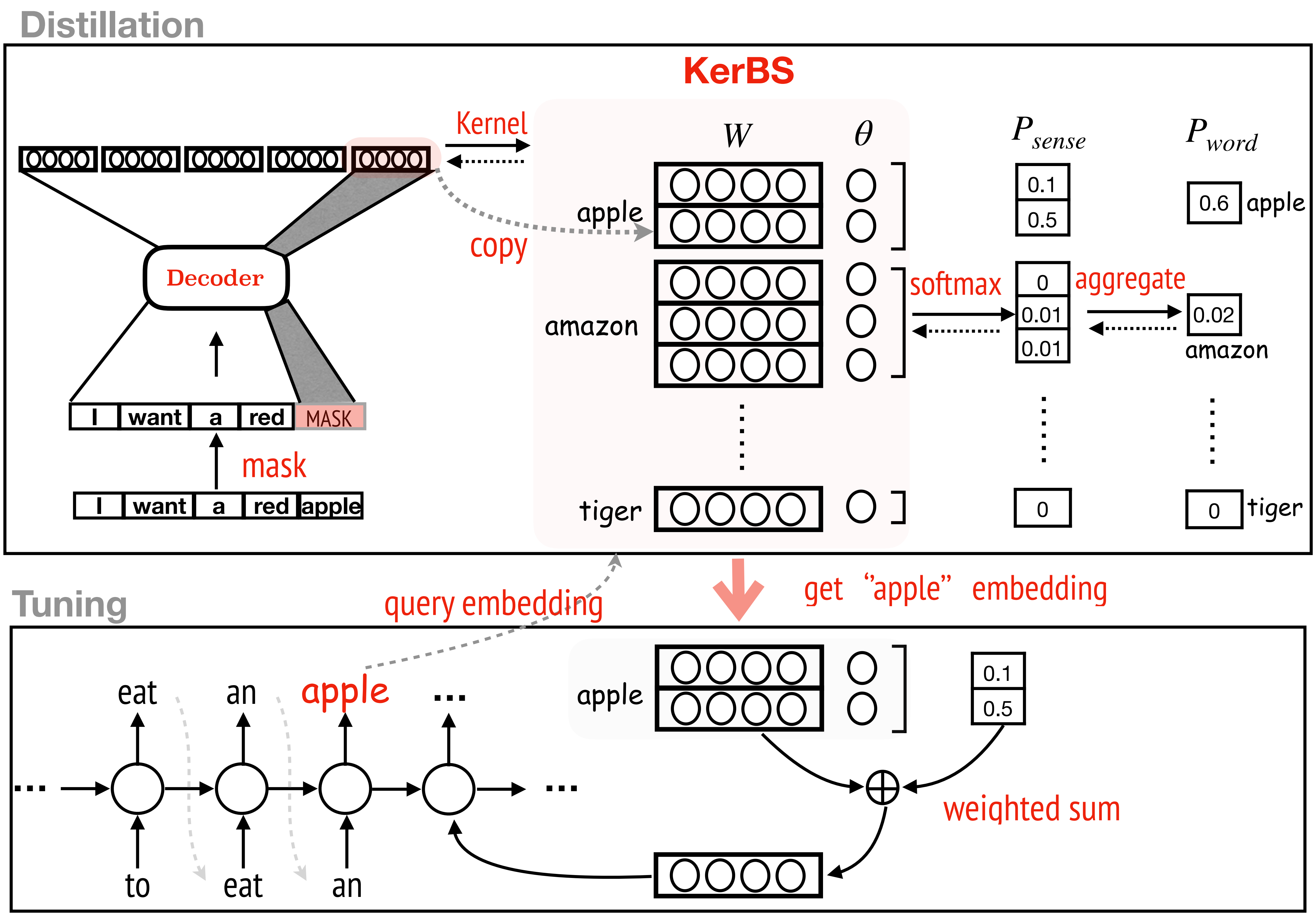
(a) θ = -2 (b) θ = 2

Figure 2: Kernel shapes of different θ.

How to decide the sense number of each word?

Dynamically change each word's sense number while training. Delete senses that are less used. Add senses to words which are not well fitted.

Dynamic Allocation



Theoretical Guarantee

Lemma 1

KerBS has the ability to learn the multi-sense property. If the real distribution of context vectors consists of several disconnected clusters, KerBS will learn to represent as many as these clusters

KerBS can learn the multi-sense property.

Lemma 2

KerBS has the ability to learn model variances. For distributions with larger variances, KerBS learns larger θ.

KerBS can learn varying variances.

Experiments-Setting

We test KerBS on 3 text generation tasks:

1. **Machine Translation (MT)** is conducted on IWSLT'16 De-En, which contains 196k pairs of sentences for training.
2. **Language modeling (LM)** is included. Following previous work, we use a 300k, 10k and 30k subset of One-Billion-Word Corpus for training, validating and testing.
3. **Dialog generation (Dialog)** is also included. We employ the DailyDialog dataset for experiment.

Main Results

Tasks	Metrics	Seq2Seq	Seq2Seq+MoS	Seq2Seq+KerBS
MT	BLEU-4	25.91	26.45	**27.28**
LM	PPL	103.12	102.72	**102.17**
Dialog	BLEU-1	16.56	13.73	**17.85**
	Human Eval.	1.24	1.04	**1.40**

Case on Dialog

Source	what do you mean ?
Seq2Seq	i mean, what s up with the and iron bars on your windows.
Seq2Seq+MoS	well, how can i put this? france is a terrible team.
Seq2Seq+KerB	well, i mean. we always do the same thing. there s no variety in

Source	now , what seems to be the trouble ?
Seq2Seq	trouble is trouble.
Seq2Seq+MoS	yeah. and he was.
Seq2Seq+KerB	not bad. but i have a bad cold today.

Source	now , what seems to be the trouble ?
Seq2Seq	trouble is trouble.
Seq2Seq+MoS	yeah. and he was.
Seq2Seq+KerB	not bad. but i have a bad cold today.

Related Research

1. Inductive Bias:

   Better Structure for Parameterization
2. Non-AutoRegressive Generation
3. Pre-trained Learning by Generation
4. Application

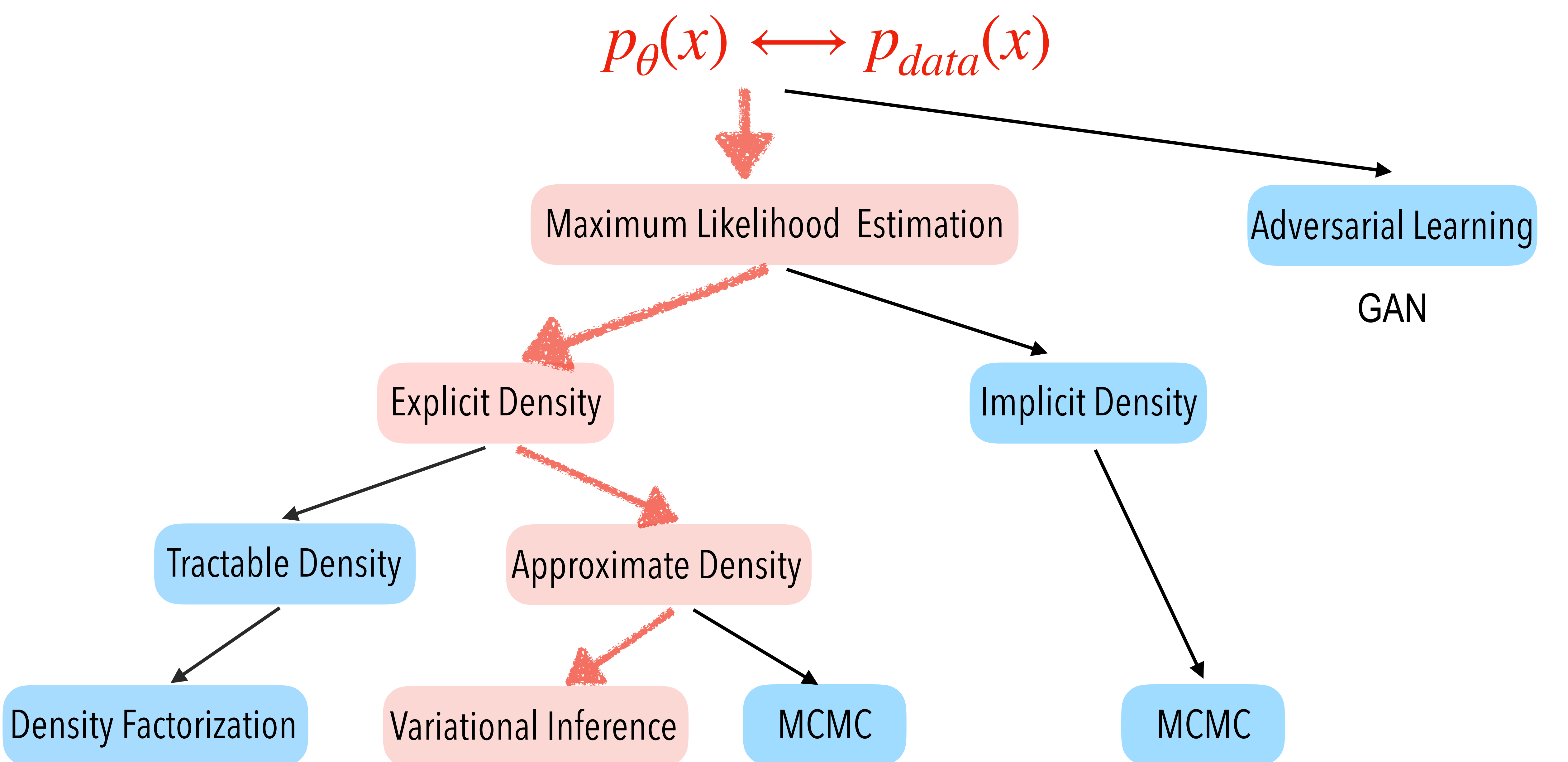
   Story telling， machine translation， Summarization， Dialog， Question Answering， etc.

Part 4

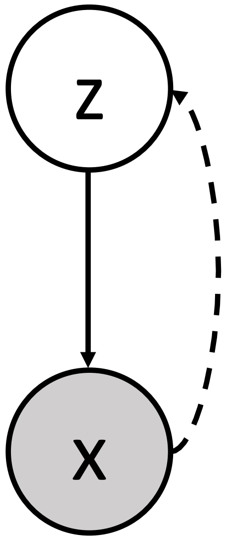
Text Generation by Variational Auto-Encoders

Approximate Density with Variational Inference

Taxonomy of DGM



Variational Auto-Encoders



Introducing Latent Variable:

$$p_\theta(x) = \int_z p(x|z)p(z)$$

Variational Lower Bound

Introducing Latent Variable:

$$p_\theta(x) = \int_z p(x\,|\,z)p(z)$$

Hard to optimize due to the exponential z 

Variational Lower Bound

Introducing Latent Variable:

$$p_\theta(x) = \int_z p(x \mid z)p(z)$$

Hard to optimize due to the exponential z 

Optimizing the Variational Lower Bound

$$J = \mathop{\mathbb{E}}_{z \sim q(z|x)} [-\log p(x \mid z)] + \mathrm{KL}(q(z \mid x) \| p(z))$$

Motivation of VAE

① Why Including Latent Variables ?

② Why Variational Inference ?

Why Including Latent Variables ?

- Data may have latent structures!

MNIST HandWriting

Why Including Latent Variables ?

- Data may have latent structures!

N = 8

Why Including Latent Variables ?

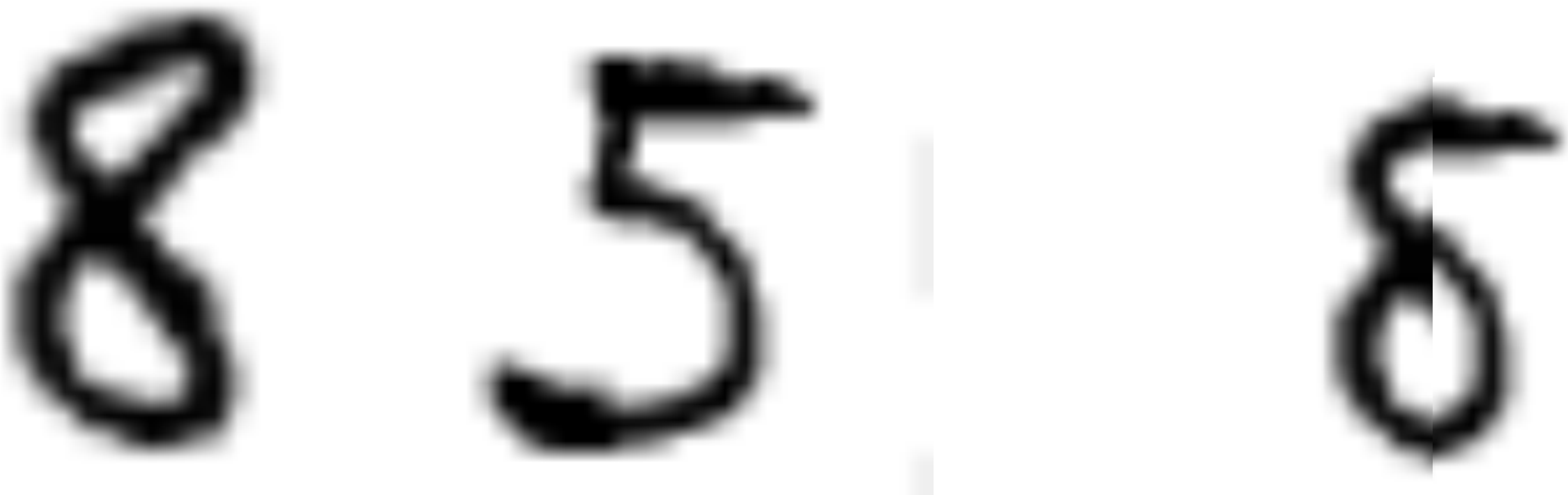
- Data may have latent structures!

N = 8

N = 5

Why Including Latent Variables ?

- Data may have latent structures!



N = 8 N = 5

Why Including Latent Variables ?

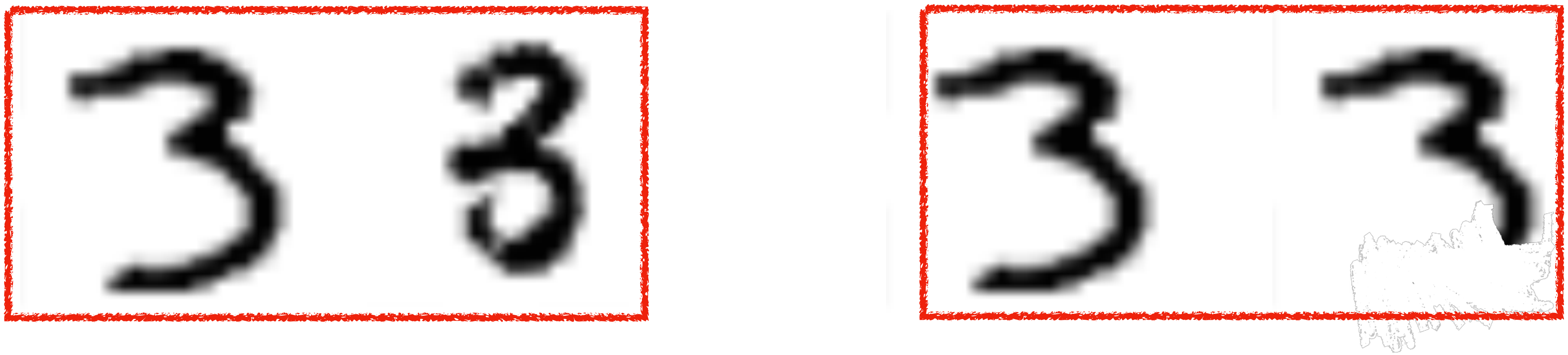
- Data may have latent structures!

N = 8 N = 5 Left of 8 and right of 5

We can avoid the last case
with latent variable?

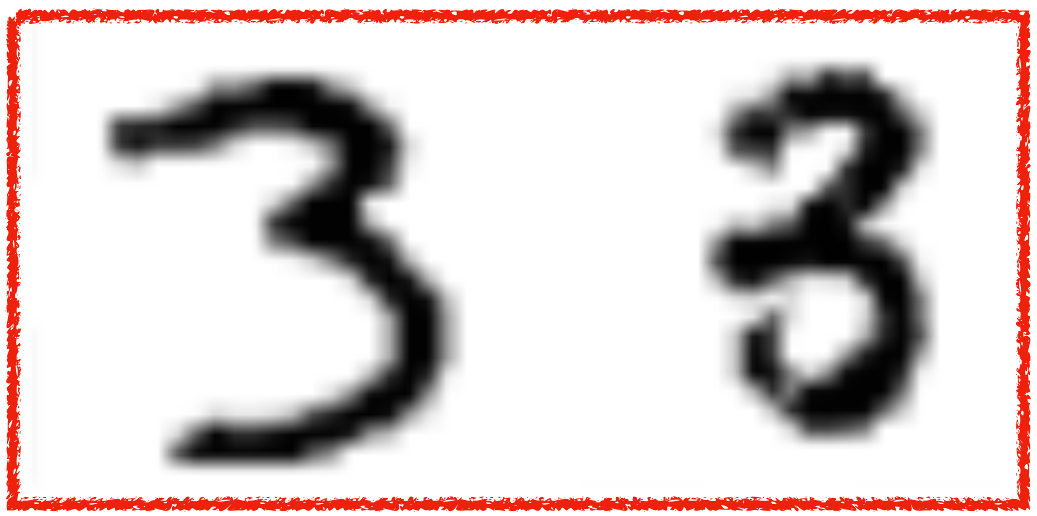
Loss between Instances

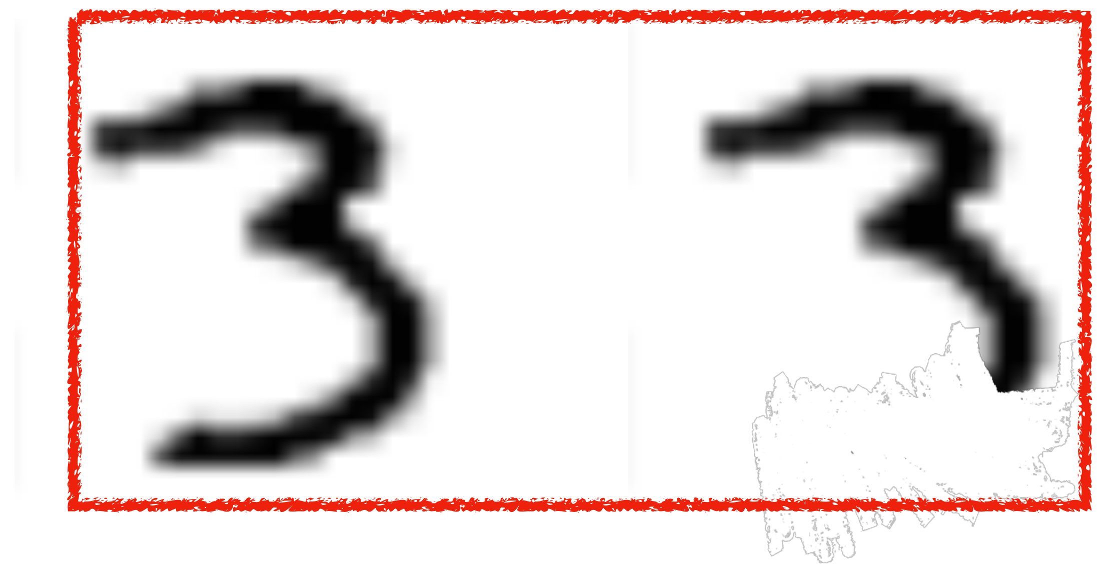


Question1: Which pair is more similar?

Question2: Which pair has lower loss?

Loss between Instances



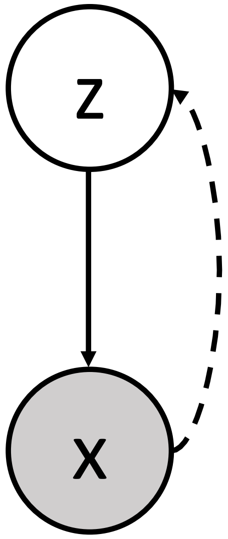


Including Latent Variable may be good for generalization !

Why Including Latent Variables ?

Model capacity can be further improved

$$p_\theta(x) = \int_z p_\theta(x \mid z) p(z)$$



With latent variable, we can present a very complex $p_\theta(x)$ using relatively simple $p_\theta(x \mid z)$!

e.g. mixture of Gaussians can present distributions which are not Gaussians.

Why Variational Inference ?

$$p_\theta(x) = \int_z p_\theta(x \mid z) p(z)$$

How to deal with the integral?

The expectation is intractable, we can use naive Monte-Carlo to estimate ?

Can be estimated by sample average

$$\sum_{all\ possible\ z} p_\theta(x, z) = |Z| \left(\sum_z p_\theta(x, z) \frac{1}{|Z|} \right) = |Z| \mathbb{E}_{z \sim Uniform(Z)}[p_\theta(x, z)]$$

Why Variational Inference ?

However, the naive Monte Carlo works in theory but not in practice!

$\boxed{\text{To most z, } p_\theta(x, z) \text{ is very small, we may also never hit z with large } p_\theta(x, z).}$

We need a more clever way to select z to reduce the variance of the estimator.

Variational Inference-Importance Sampling Perspective

$$p_\theta(x) = \int_z p_\theta(x \mid z)p(z)$$

$$= \int_z Q(z)p_\theta(x \mid z)p(z)/Q(z)$$

$$= \mathbb{E}_{z \sim Q(z)} p_\theta(x, z)/Q(z)$$

Introducing Q as proposal in importance sampling, obtaining a less-variance estimation of $p_\theta(x)$.

Note that optimal $Q = p_\theta(z \mid x)$, with 0 variance.

Derivation of ELBO

We are actually interested in $logp_{\theta}(x)$ during MLE

$$logp_{\theta}(x) = logE_{z \sim Q}[p_{\theta}(x, z)/Q(z)]$$

Jensen

Inequality

$$logp_{\theta}(x) \geq E_{z \sim Q}log[p_{\theta}(x, z)/Q(z)]$$

Bayes Rule

$$logp_{\theta}(x) \geq E_{z \sim Q}[logp_{\theta}(x \mid z) + \mathbb{KL}[Q(z) \mid\mid P(z)]]$$

Replacing with

amortization

$$logp_{\theta}(x) \geq E_{z \sim Q}[logp_{\theta}(x \mid z) + \mathbb{KL}[Q(z \mid x) \mid\mid P(z)]]$$

Another Derivation of ELBO

$$\mathbb{KL}[Q(z)||P(z|x)] = E_{z\sim Q}[logQ(z) - logP(z|x)]$$

Bayes Rule

$$\mathbb{KL}[Q(z)||P(z|x)] = E_{z\sim Q}[logQ(z) - logP(x|z) - logP(z)] + logP(x)$$

Transposition

$$logP(x) - \mathbb{KL}[Q(z)||P(z|X)] = E_{z\sim Q}[logP(X|z) - \mathbb{KL}[Q(z)||P(z)]$$

Replacing

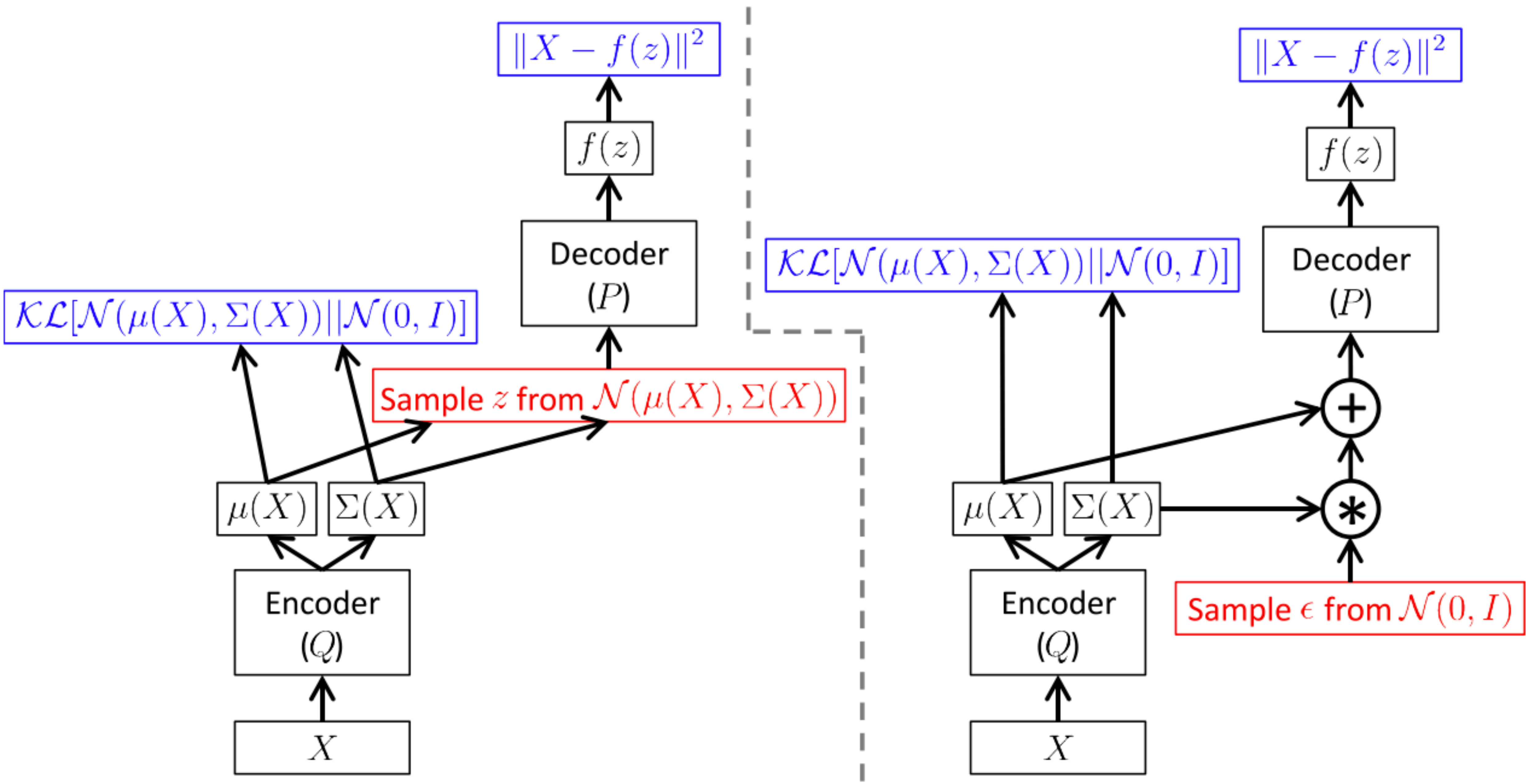
$$logP(x) - \mathbb{KL}[Q(z|X)||P(z|X)] = E_{z\sim Q}[logP(X|z) - \mathbb{KL}[Q(z|X)||P(z)]$$

“Auto-Encoder”

$$logP(x) - \mathbb{KL}[Q(z|X)||P(z|X)] = \underbrace{E_{z\sim Q}[logP(X|z)]}_{\text{Decoder}} - \mathbb{KL}[\underbrace{Q(z|X)}_{\text{Encoder}}||P(z)]$$

This is why such variational Bayes model is so called “Variational Auto-Encoder”

Training of VAE

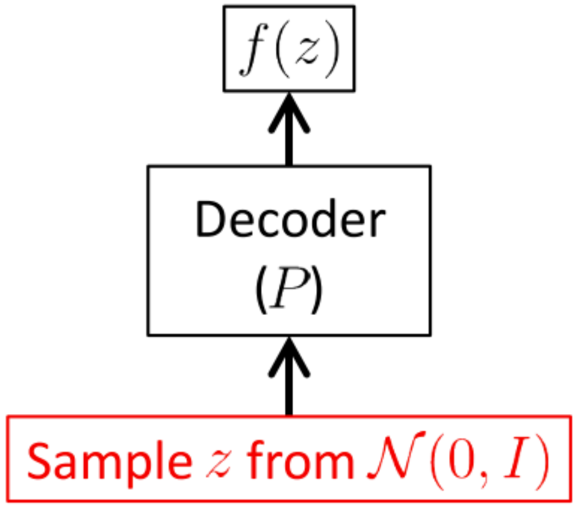


Training

Reparameterization

Carl Doersch, Tutorial on Variational Autoencoders, in arXiv:1606.05908.

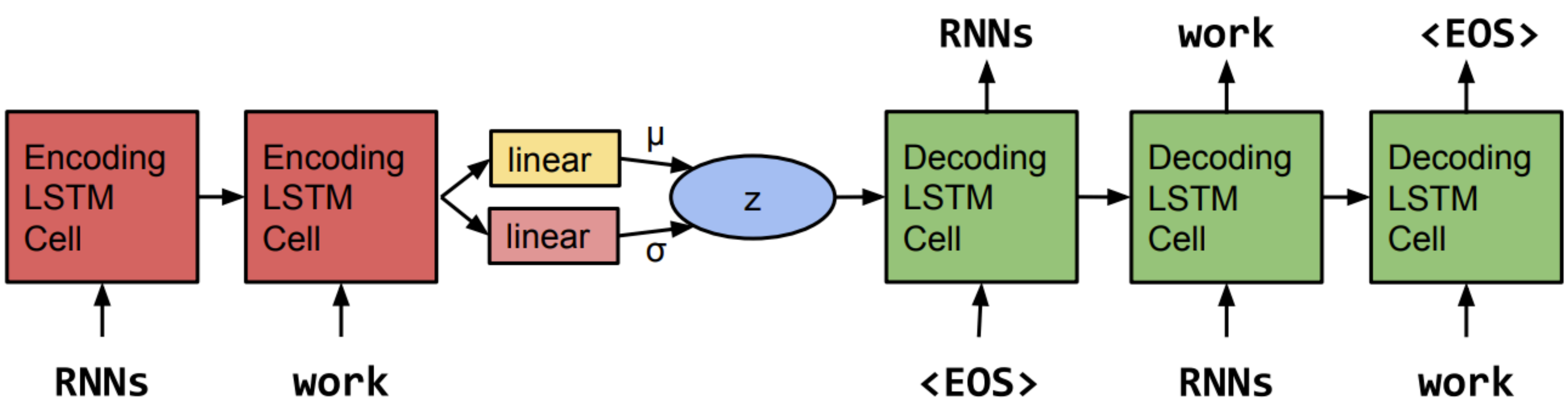
Decoding



Back to the Motivation

- For text generation, the density is always decomposed （Auto-Regressive）.
- What's the benefits of VAEs?

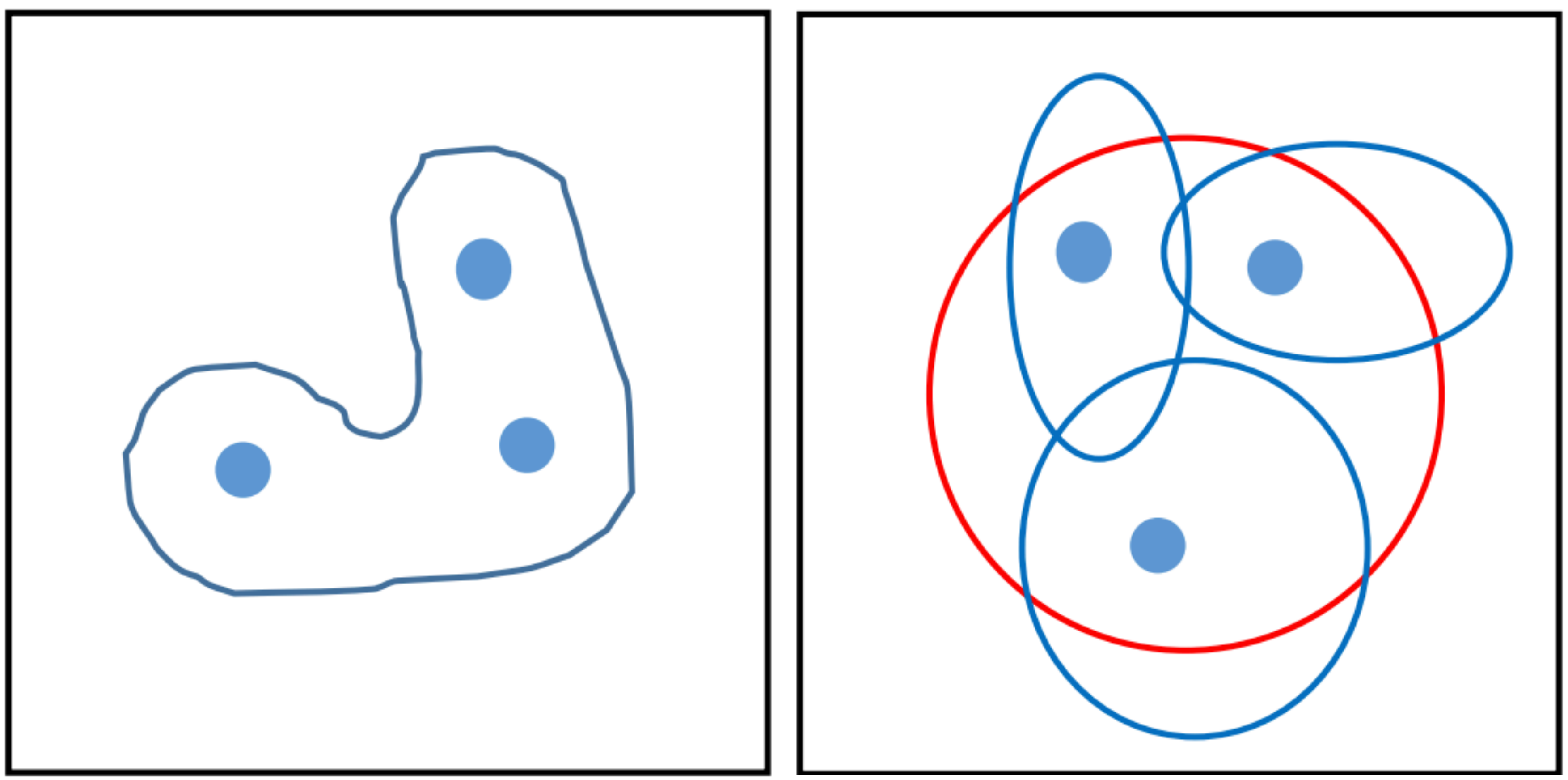
VAE for Text Generation



Bowman et al., Generating Sentences from a Continuous Space, in CoNLL, 2016.

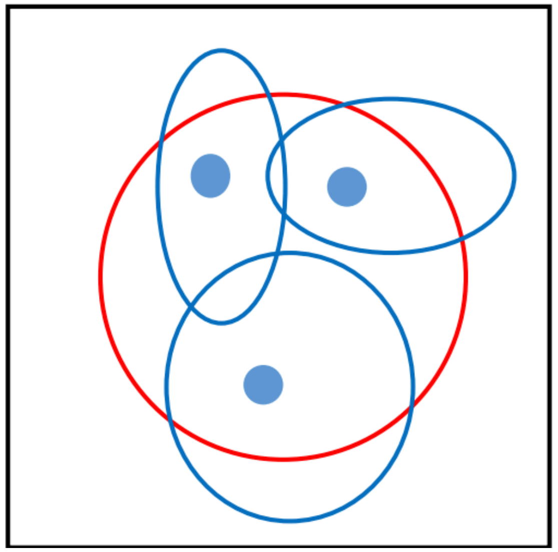
Latent Spaces of VAE

AE VAE

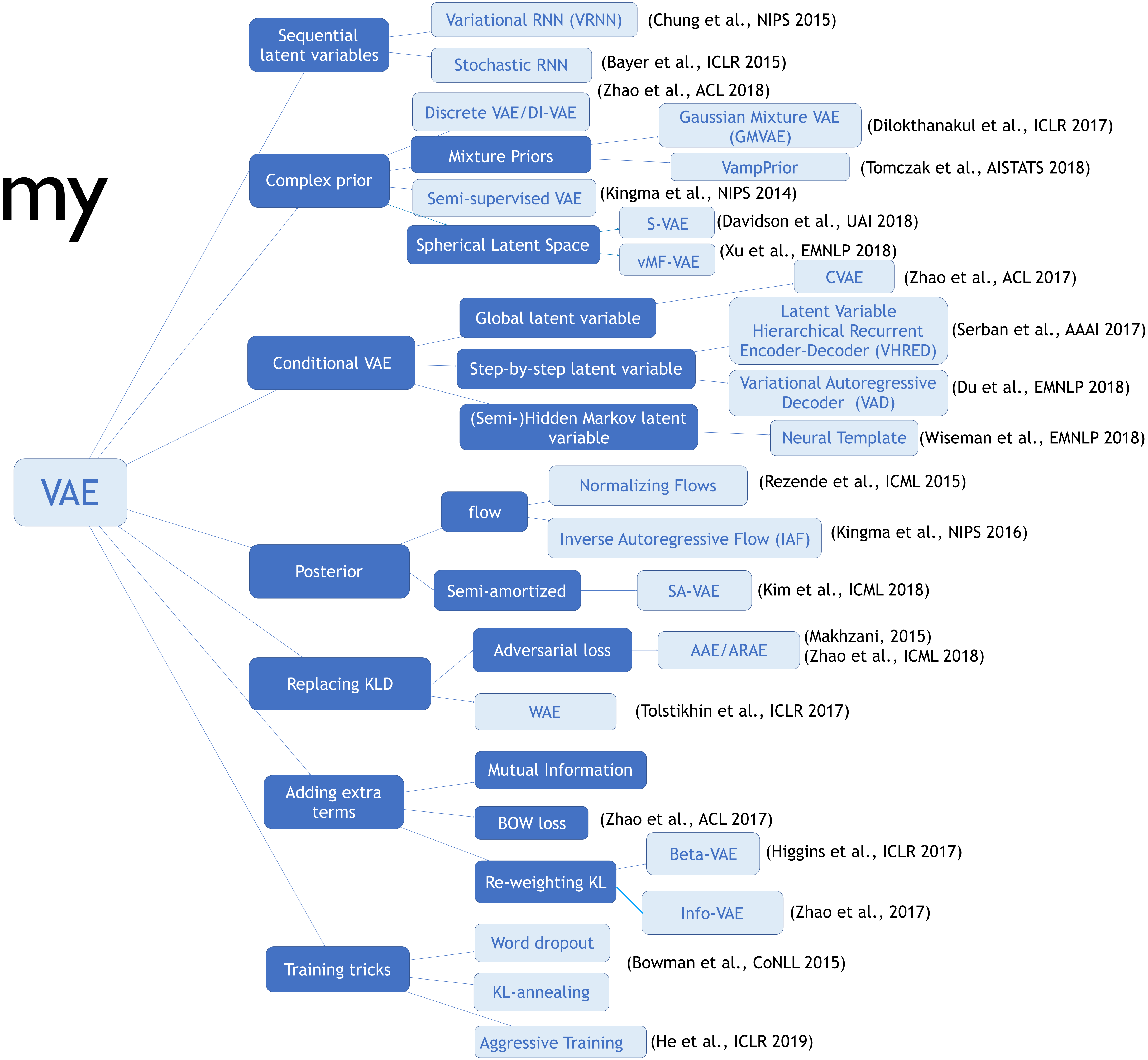


Benefits of VAE

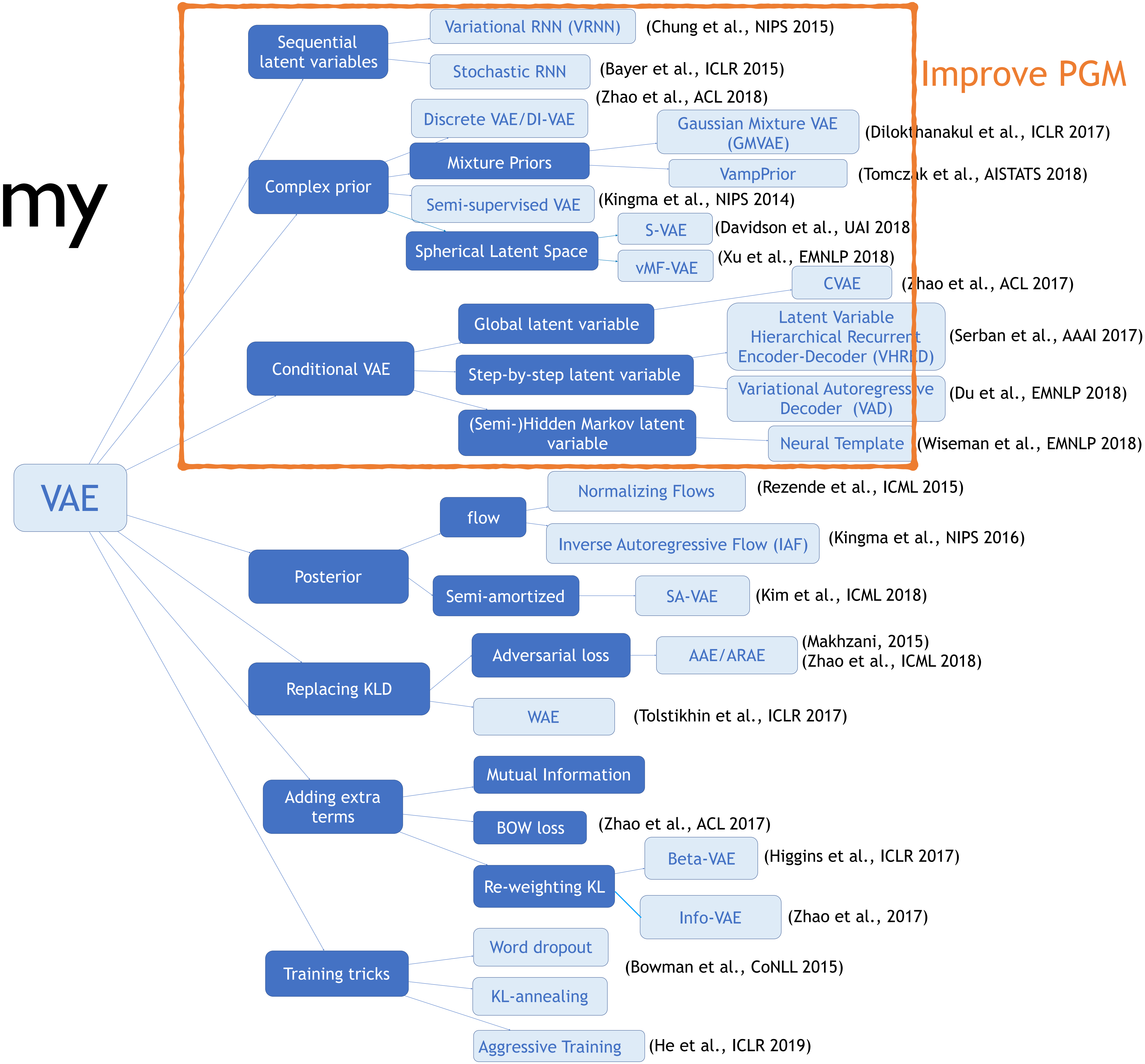
- Regularized Latent Variables:
  1. Sampling
  2. Manipulating



VAE Taxonomy



VAE Taxonomy



Reference

[1] Chung J, Kastner K, Dinh L, et al. A recurrent latent variable model for sequential data[C]//Advances in neural information processing systems. 2015: 2980-2988.
[2] Bayer J, Osendorfer C. Learning stochastic recurrent networks[J]. arXiv preprint arXiv:1411.7610, 2014.
[3] Zhao T, Lee K, Eskenazi M. Unsupervised Discrete Sentence Representation Learning for Interpretable Neural Dialog Generation[C]//Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2018: 1098-1107.
[4] Dilokthanakul N, Mediano P A M, Garnelo M, et al. DEEP UNSUPERVISED CLUSTERING WITH GAUSSIAN MIXTURE VARIATIONAL AUTOENCODERS[J].
[5] Tomczak J, Welling M. VAE with a VampPrior[C]//International Conference on Artificial Intelligence and Statistics. 2018: 1214-1223.
[6] Kingma D P, Mohamed S, Rezende D J, et al. Semi-supervised learning with deep generative models[C]//Advances in neural information processing systems. 2014: 3581-3589.
[7] Zhao T, Zhao R, Eskenazi M. Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders[C]//Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2017: 654-664.
[8] Serban I V, Sordoni A, Lowe R, et al. A hierarchical latent variable encoder-decoder model for generating dialogues[C]//Thirty-First AAAI Conference on Artificial Intelligence. 2017.
[9] Du J, Li W, He Y, et al. Variational Autoregressive Decoder for Neural Response Generation[C]//Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 2018: 3154-3163.
[10] Wiseman S, Shieber S, Rush A. Learning Neural Templates for Text Generation[C]//Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 2018: 3174-3187.
[11] Rezende D, Mohamed S. Variational Inference with Normalizing Flows[C]//International Conference on Machine Learning. 2015: 1530-1538.
[12] Kingma D P, Salimans T, Jozefowicz R, et al. Improved variational inference with inverse autoregressive flow[C]//Advances in neural information processing systems. 2016: 4743-4751.
[13] Kim Y, Wiseman S, Miller A, et al. Semi-Amortized Variational Autoencoders[C]//International Conference on Machine Learning. 2018: 2683-2692.
[14] Makhzani A, Shlens J, Jaitly N, et al. Adversarial autoencoders[J]. arXiv preprint arXiv:1511.05644, 2015.
[15] Zhao J, Kim Y, Zhang K, et al. Adversarially Regularized Autoencoders[C]//International Conference on Machine Learning. 2018: 5897-5906.
[16] Tolstikhin I, Bousquet O, Gelly S, et al. Wasserstein auto-encoders[J]. arXiv preprint arXiv:1711.01558, 2017.
[17] Zhao S, Song J, Ermon S. Infovae: Information maximizing variational autoencoders[J]. arXiv preprint arXiv:1706.02262, 2017.
[18] Zhao T, Zhao R, Eskenazi M. Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders[C]//Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2017: 654-664.
[19] Higgins I, Matthey L, Pal A, et al. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework[J]. ICLR, 2017, 2(5): 6.
[20] Bowman S, Vilnis L, Vinyals O, et al. Generating Sentences from a Continuous Space[C]//Proceedings of the Twentieth Conference on Computational Natural Language Learning (CoNLL). 2016.
[21] He J, Spokoyny D, Neubig G, et al. Lagging Inference Networks and Posterior Collapse in Variational Autoencoders[J]. 2018.
[22] Xu J, Durrett G. Spherical Latent Spaces for Stable Variational Autoencoders[C]//Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 2018: 4503-4513.
[23] Davidson T R, Falorsi L, De Cao N, et al. Hyperspherical Variational Auto-Encoders[J].

Variational Auto-Encoders

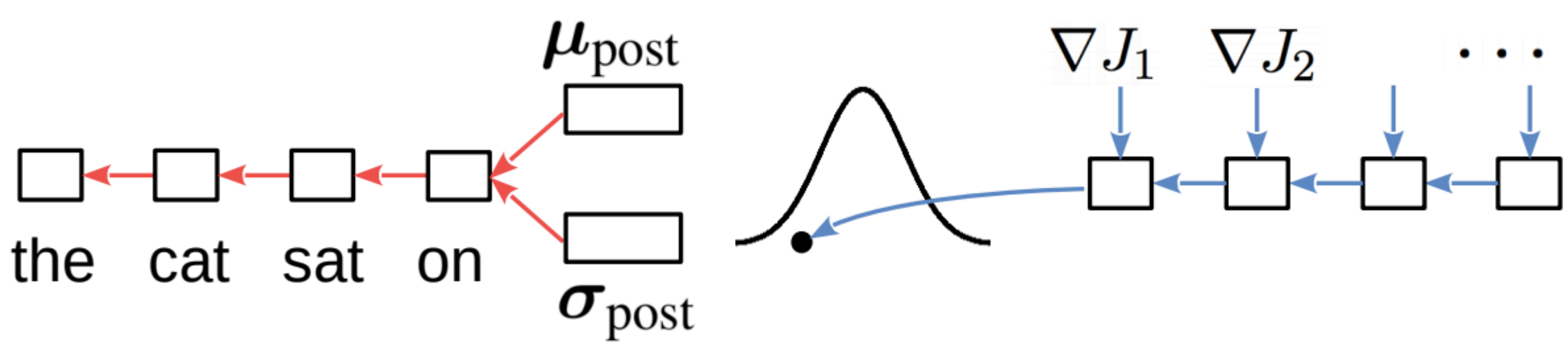
$$p_{model}(x) = \int_z p(x|z)p(z)$$

- VAE: Treating *z* as a random variable
  - Imposing prior $p(z) = \mathcal{N}(\mathbf{0}, \mathbf{I})$
  - Variational posterior $q(z|x) = \mathcal{N}(\boldsymbol{\mu}_{\mathrm{NN}}, \operatorname{diag} \boldsymbol{\sigma}^2_{\mathrm{NN}})$
  - Optimizing the variational lower bound

$$J = \mathop{\mathbb{E}}_{z \sim q(z|x)} [-\log p(x|z)] + \mathrm{KL}(q(z|x) \| p(z))$$

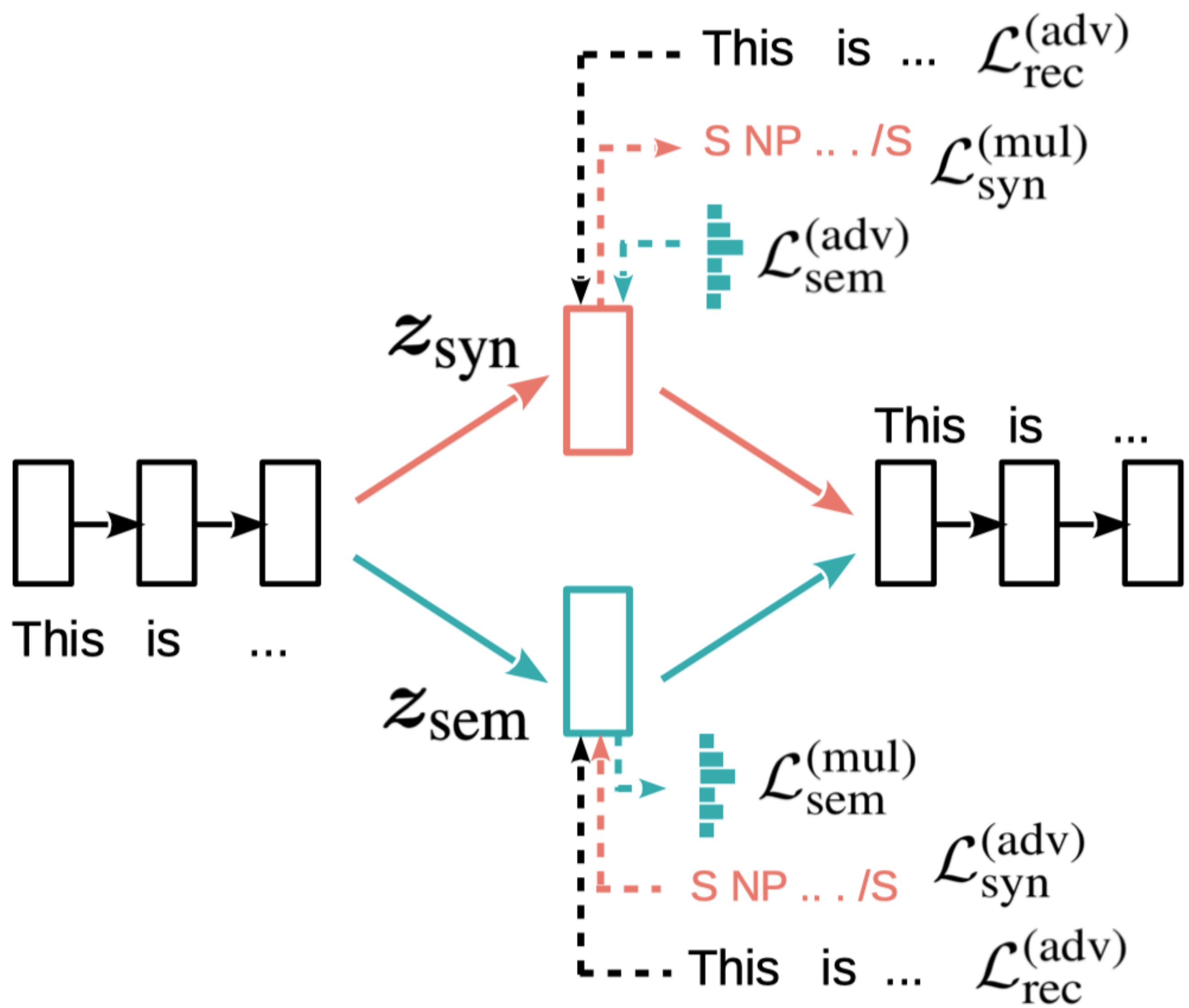
Variational Auto-Encoders

$$p_{model}(x) = \int_z p(x \mid z)p(z)$$



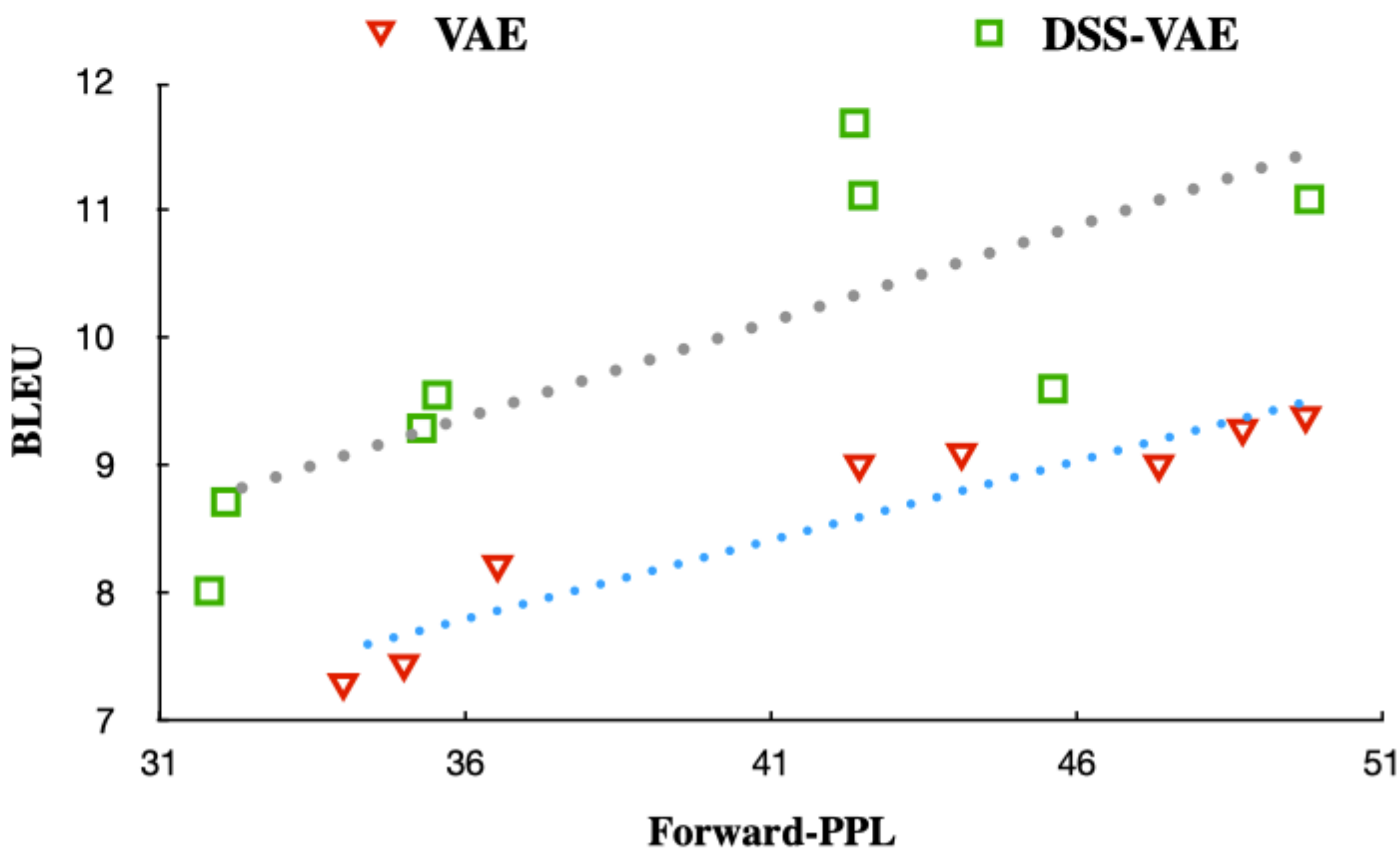
Stochastic encoder **Autoregressive decoder**

Disentangling Syntax and Semantics in Latent Space



Bao et al., Generating Sentences from Disentangled Syntactic and Semantic Spaces, in ACL, 2019.

BLEU *VS.* PPL



Bao et al., Generating Sentences from Disentangled Syntactic and Semantic Spaces, in ACL, 2019.

BLEU *VS.* PPL

Model	Reverse PPL$^{\downarrow}$
Real data	70.76
LSTM-LM	132.46
PRPN-LM	116.67
VAE	125.86
DSS-VAE	**116.23**

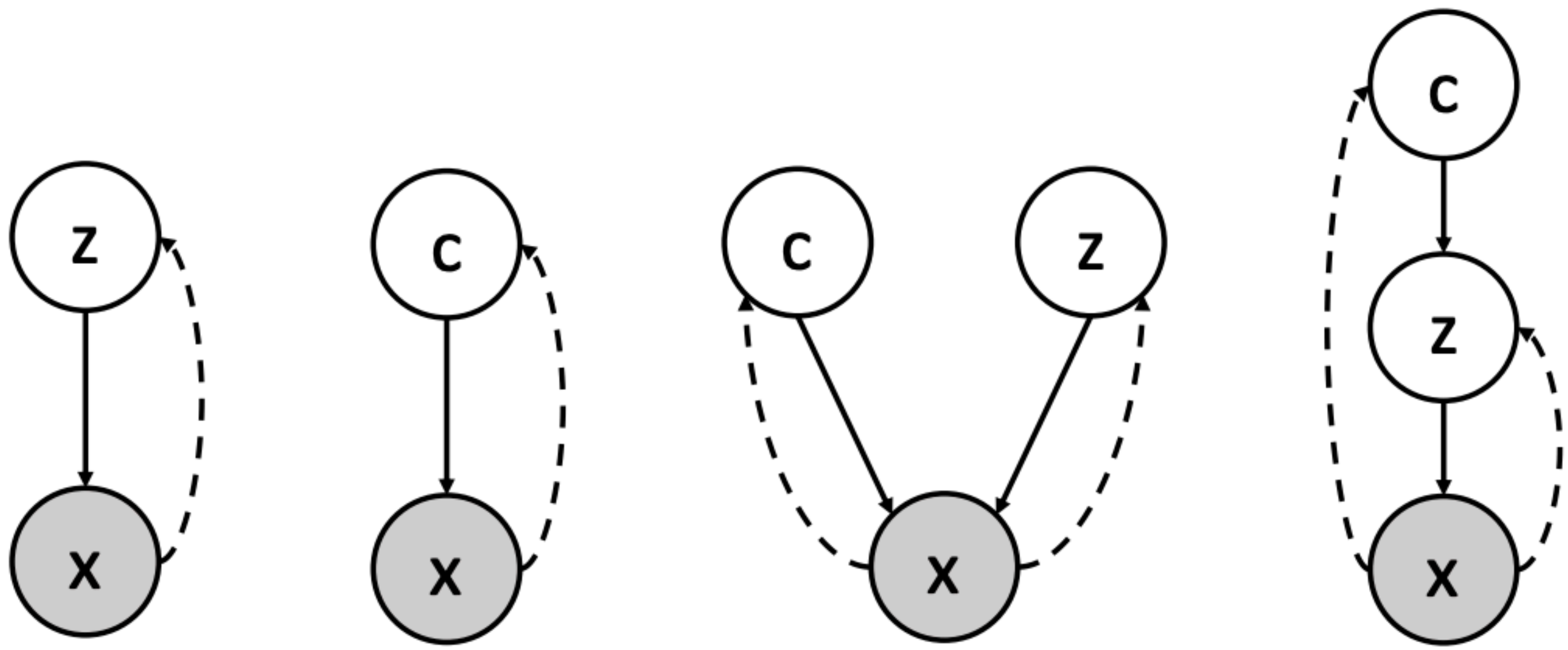
Table 2: Reverse PPL reflect the diversity and fluency of sampling data, the lower$^{\downarrow}$, the better. Training on the model sampled and evaluated on the real test set. We set the same KL weight for DSS-VAE and VAE here.(KL weight=1.0)

Model	BLEU-ref$^{\uparrow}$	BLEU-ori$^{\downarrow}$
Origin Sentence†	30.49	100
VAE-SVG-eq (supervised)‡	22.90	–
VAE (unsupervised)†	9.25	27.23
CGMH†	18.85	50.18
DSS-VAE	**20.54**	52.77

Table 3: Performance of paraphrase generation. The larger$^{\uparrow}$ (or lower$^{\downarrow}$), the better. Some results are quoted from †Miao et al. (2019) and ‡Gupta et al. (2018).

Bao et al., Generating Sentences from Disentangled Syntactic and Semantic Spaces, in ACL, 2019.

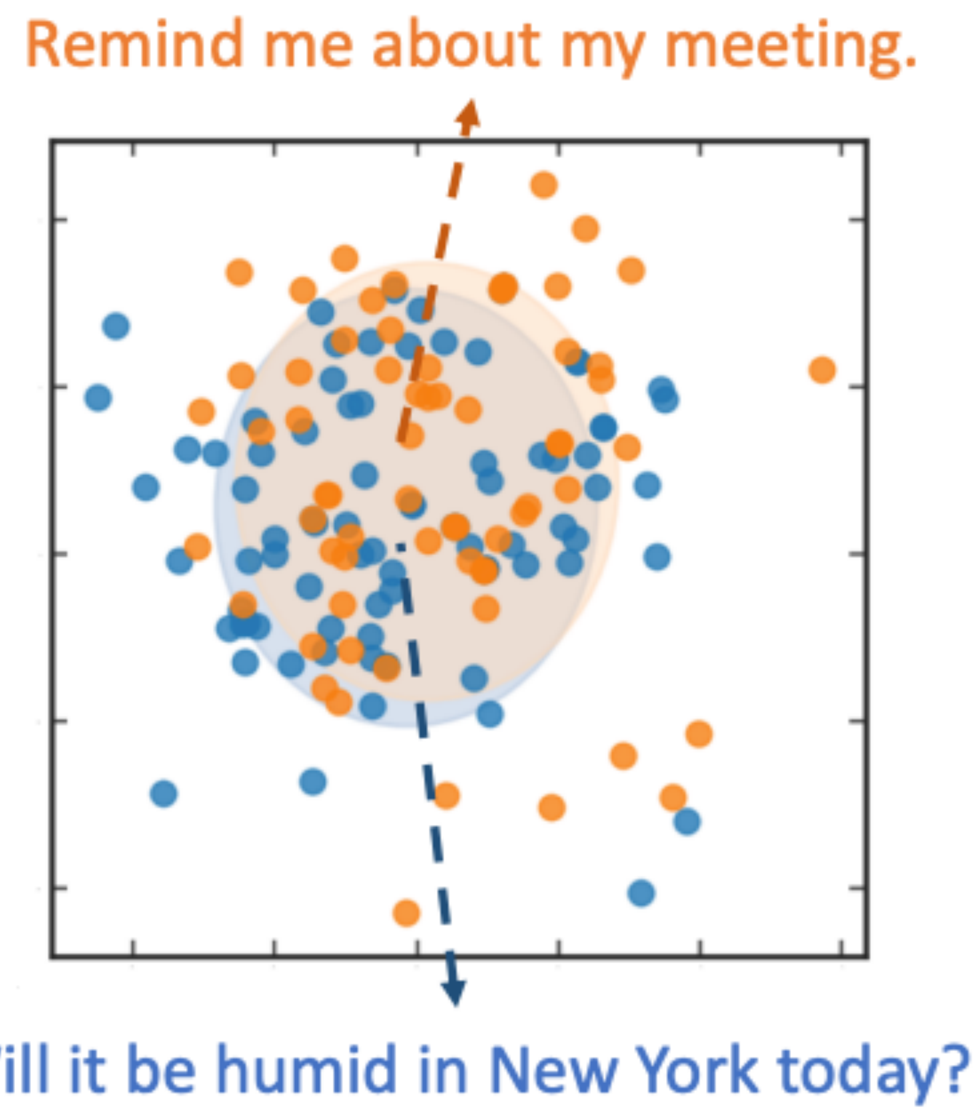
Gaussian Mixture VAE



(a) VAE (b) DI-VAE (c) semi-VAE (d) GMVAE

Shi et al., Fixing Gaussian Mixture VAEs for Interpretable Text Generation, in Arxiv, 1906.06719.

Mode-Collapse



(a) GMVAE

Shi et al., Fixing Gaussian Mixture VAEs for Interpretable Text Generation, in arXiv, 1906.06719.

Theoretical Analysis

Theorem 1. *Maximizing the $\mathcal{R}_c$ pushes a close upper bound of Var$_M$, S_{μ_ϕ}, to decrease. Here $S_{\mu_\phi} = \sum_k (\mu_\phi - \mu_k)^T(\mu_\phi - \mu_k)$ is the squared sum of distance between μ_k and μ_ϕ.*

Theorem 2. *$\mathcal{R}_z$ contains a negative regularization term of Var$_{q_\phi(c|x)}\, \mu_c$.*

$\mathcal{R}_z$ could be re-written as

$$\mathbb{E}_{q_\phi(z|x)} \sum_c q_\phi(c|x) \log \frac{p(z|c)}{q_\phi(z|x)} = -\,\mathbf{KL}(q_\phi(z|x)||\hat{p}(z|x))) - \frac{1}{2\sigma^2}\,\mathbf{Var}_{q_\phi(c|x)}\, \mu_c,$$

Shi et al., Fixing Gaussian Mixture VAEs for Interpretable Text Generation, in arXiv, 1906.06719.

DGMVAE

Theorem 1. *Maximizing the $\mathcal{R}_c$ pushes a close upper bound of Var$_M$, S_{μ_ϕ}, to decrease. Here $S_{\mu_\phi} = \sum_k (\mu_\phi - \mu_k)^T(\mu_\phi - \mu_k)$ is the squared sum of distance between μ_k and μ_ϕ.*

Theorem 2. *$\mathcal{R}_z$ contains a negative regularization term of Var$_{q_\phi(c|x)}\, \mu_c$.*

$\mathcal{R}_z$ could be re-written as

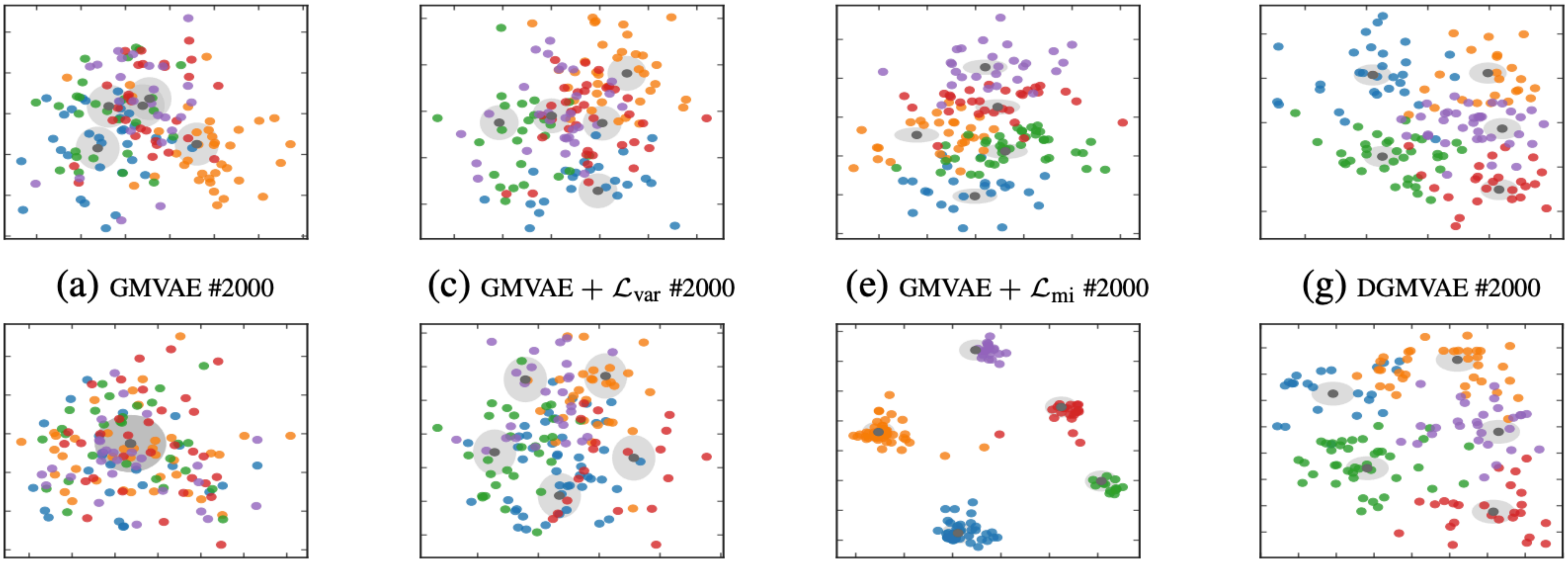
$$\mathbb{E}_{q_\phi(z|x)} \sum_c q_\phi(c|x) \log \frac{p(z|c)}{q_\phi(z|x)} = -\,\mathbf{KL}(q_\phi(z|x)||\hat{p}(z|x))) - \frac{1}{2\sigma^2}\, \mathbf{Var}_{q_\phi(c|x)}\, \mu_c,$$

Our Solution:

$$\mathbb{E}_x\, \mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z) - \mathbf{KL}(q_\phi(c)||p(c)) - \mathbb{E}_x[\mathbf{KL}(q_\phi(z|x)||\hat{p}(z|x)))] - \beta'\, \mathbb{E}_x\, \mathbf{Var}_{q_\phi(c|x)}\, \mu_c$$

Shi et al., Fixing Gaussian Mixture VAEs for Interpretable Text Generation, in arXiv, 1906.06719.

Visualization



(a) GMVAE #2000

(c) GMVAE + $\mathcal{L}_{\text{var}}$ #2000

(e) GMVAE + $\mathcal{L}_{\text{mi}}$ #2000

(g) DGMVAE #2000

(b) GMVAE #10000

(d) GMVAE + $\mathcal{L}_{\text{var}}$ #10000

(f) GMVAE + $\mathcal{L}_{\text{mi}}$ #10000

(h) DGMVAE #10000

Shi et al., Fixing Gaussian Mixture VAEs for Interpretable Text Generation, in arXiv, 1906.06719.

Results on PTB

	Evaluation Results				Regularization Terms			
Model	**rPPL**$^{\downarrow}$	**BLEU**$^{\uparrow}$	**wKL**$^{\downarrow}$	**PPL**$^{\downarrow}$	**KL(z)**	**KL(c)**	**VM**	**MI**
Test Set	-	100.0	**0.14**	-	-	-	-	-
RNNLM (Mikolov et al., 2010)	-	-	-	117.60	-	-	-	-
AE (Vincent et al., 2010)	730.81	**10.88**	0.58	**31.90**	-	-	-	-
VAE (Kingma and Welling, 2013)	922.71	3.73	0.76	91.95	6.62	-	-	-
DAE	797.17	3.93	0.58	88.55	-	-	-	-
DVAE	453.53	3.61	0.58	100.56	-	1.74	-	1.22
DI-VAE (Zhao et al., 2018b)	425.11	4.19	0.69	93.72	-	0.13	-	1.26
semi-VAE (Kingma et al., 2014)	779.53	3.59	0.79	93.78	6.97	0.02	-	0.019
semi-VAE + $\mathcal{L}_{\text{mi}}$	721.34	4.87	0.73	92.95	0.49	0.14	-	1.34
GMVAE	923.66	4.17	0.80	90.26	7.13	0.02	0.38	0.016
DGMVAE $-$ $\mathcal{L}_{\text{var}}$	331.80	6.34	0.45	61.77	13.03	0.10	9.93	1.30
DGMVAE $-$ $\mathcal{L}_{\text{mi}}$	560.56	5.64	0.62	71.12	3.87	0.31	24.84	0.28
DGMVAE	**244.30**	**8.45**	**0.35**	**49.60**	6.41	0.10	21.42	1.19

Results on Dialog

	DD			
Model	**MI**	**BLEU**$^{\uparrow}$	**act**$^{\uparrow}$	**em**$^{\uparrow}$
DI-VAE	1.20	3.05	0.18	0.09
semi-VAE	0.03	4.06	0.02	0.08
semi-VAE + $\mathcal{L}_{\text{mi}}$	1.21	3.69	0.21	0.14
GMVAE	0.00	2.03	0.08	0.02
DGMVAE $-$ $\mathcal{L}_{\text{var}}$	1.41	2.96	0.19	0.09
DGMVAE $-$ $\mathcal{L}_{\text{mi}}$	0.53	**7.63**	0.11	0.09
DGMVAE	1.32	7.39	**0.23**	**0.16**

Table 2: Results of interpretable language generation on DD. Mutual information (MI), BLEU and homogeneity with actions (act) and emotions (em) are shown. The larger$^{\uparrow}$, the better.

	Automatic Metrics			
Model	**BLEU**	**Ave.**	**Ext.**	**Grd.**
DI-VAE	7.06	76.17	43.98	60.92
DGMVAE	**10.16**	**78.93**	**48.14**	**64.87**

	Human Evaluation	
Model	**Quality**	**Consistency**
DI-VAE	2.31	3.08
DGMVAE	**2.45**	**3.35**

Table 3: Dialog evaluation results on SMD. Four automatic metrics: BLEU, average (Ave.), extrema (Ext.) and greedy (Grd.) word embedding based similarity are shown. Response quality and consistency within the same *c* are scored by human.

Cases

Act	Inform-route/address
Utt	There is a Safeway 4 miles away. There are no hospitals within 2 miles. There is Jing Jing and PF Changs.
Act	Request-weather
Utt	What is the weather today? What is the weather like in the city? What's the weather forecast in New York?

Table 4: Example actions (Act) and corresponding utterances (Utt) discovered by DG-MVAE on SMD. The action name is annotated by experts.

Context	*Sys:* Taking you to Chevron.
Predict	(1-1-3, thanks) Thank you car, let's go there! (1-0-2, request-address) What is the address?
Context	*User:* Make an appointment for the doctor.
Predict	(3-2-4, set-reminder) Setting a reminder for your doctor's appointment on the 12th at 3pm. (3-0-4, request-time) What time would you like to be schedule your doctor's appointment?

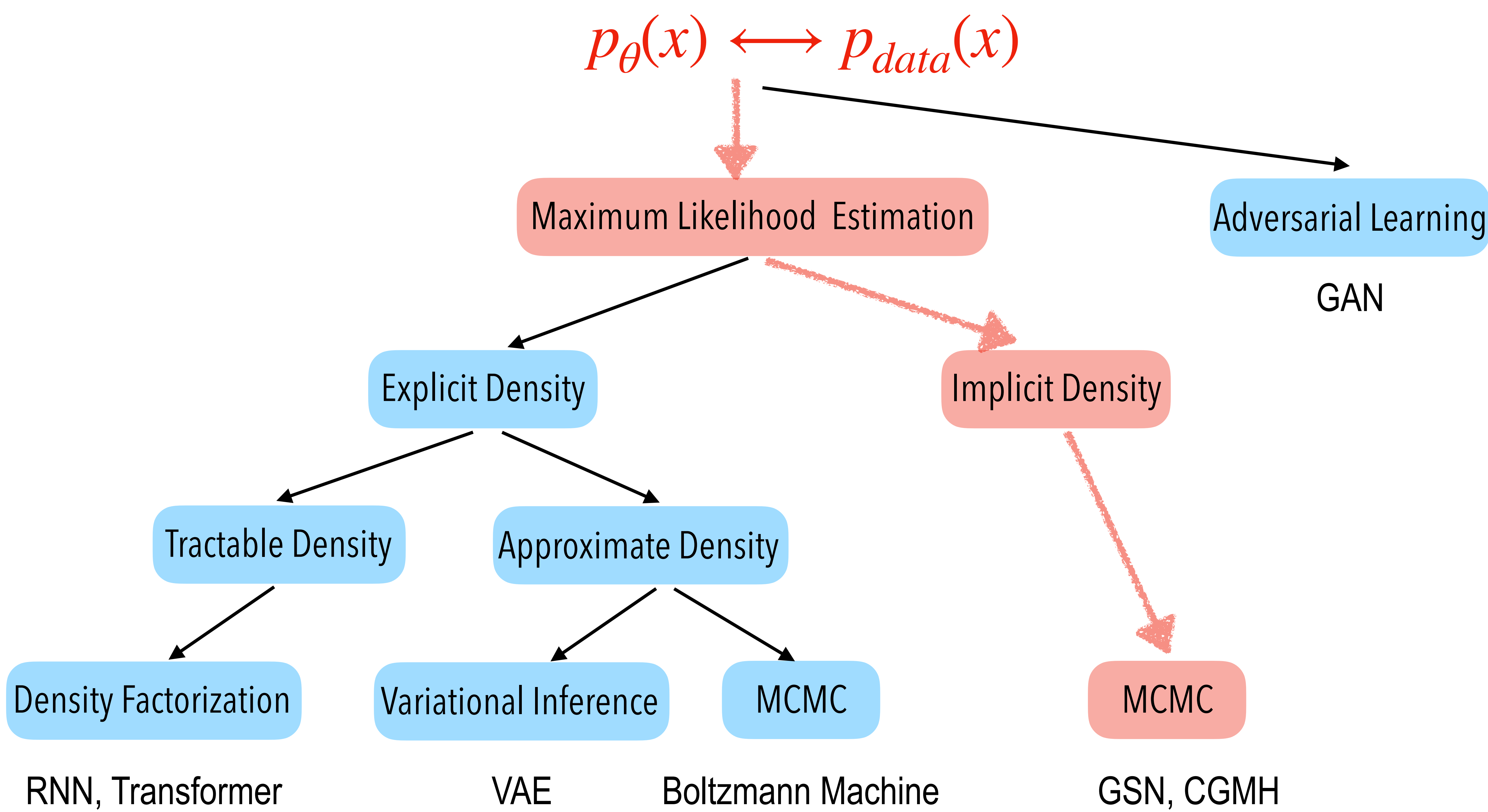
Table 5: Dialog cases on SMD, which are generated by sampling different c from policy network. The label of sampled c are listed in parentheses with the annotated action name.

Part 5

Text Generation by MCMC

Text Generation without Explicit Density and in Arbitrary Order

Taxonomy of DGM



Generation by Sampling

- Could we better exploit sampling in text generation?
- Especially for some special cases!

Sampling has Larger Potentials

Sampling Can Be Faster Than Optimization

Yi-An Ma[a], Yuansi Chen[b], Chi Jin[a], Nicolas Flammarion[a], and Michael I. Jordan*[a, b]

[a]Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720
[b]Department of Statistics, University of California, Berkeley, CA 94720

November 21, 2018

Problem Definition

- Generating sentence satisfying constraints:
  - Hard constrains: Keyword must occur in sentences
    - E.g. Juice -> Brand natural juice, specially made for you
  - Soft constrains: Semantically similar to a given sentence (paraphrase)
    - E.g. The movie is a great success -> It is one of my favorite movies

Advertisement Slogan by Constrained Generation

Keywords from Advertiser

Advertisement Slogan

<u>Rin</u> <u>clothes</u> <u>bright</u>

Challenges

- To generation samples (sentences) from the target distribution

$$\pi(x) = \prod_t P\left(x_t \mid x_{0:t-1}\right) \cdot \prod_{\mathrm{i}} P_C^i(x)$$

language model probability

Indicator(0-1) function for constraints

- $\pi(x)$ is high-dimensional, and no direct sampling method.

Main Idea of CGMH

- Instead of sampling from $\pi(x)$ directly, generate samples iteratively:
  - Starting with initial keywords
  - next sentence based on modification of previous
  - action proposals to modify the sentences
- Metropolis-Hastings Algorithm

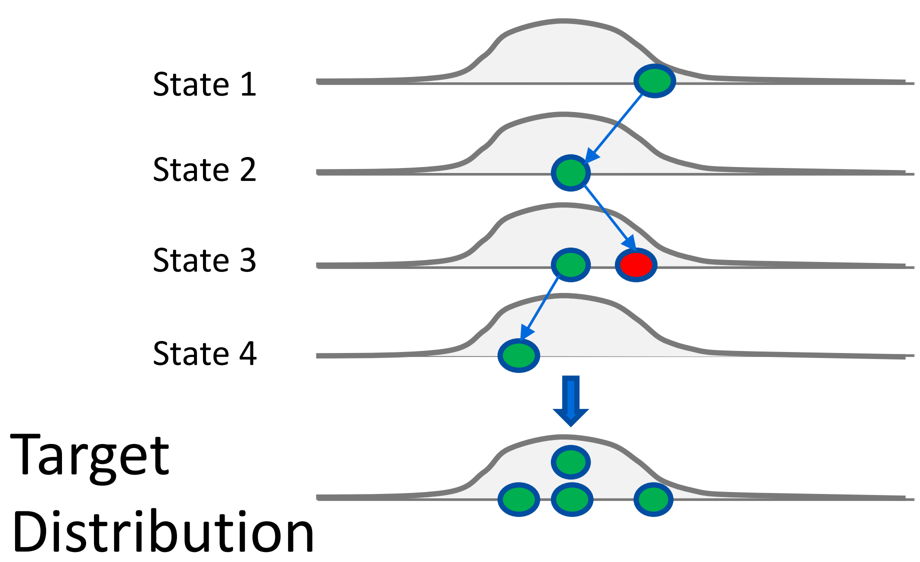
Miao et al., CGMH: Constrained Sentence Generation by Metropolis-Hastings Sampling, in AAAI, 2019.

Metropolis Hastings Sampling

Metropolis-Hastings(MH) perform sampling by first **proposes** a transition, and then **accepts or rejects** the transition.

$$A(x'|x_{t-1}) = \min\left(1, \frac{\pi(x')\cdot g(x_{t-1}|x')}{\pi(x_{t-1})\cdot g(x'|x_{t-1})}\right)$$

g is proposal distribution



Sampling in Sentence Space

CGMH performs Metropolis-Hastings sampling directly in sentence space:

Step	Action	Acc/Rej	Sentences
0	[Input]		BMW sports
1	Insert	Accept	BMW sports car
2	Insert	Accept	BMW the sports car
...	...	...	...
6	Insert	Accept	BMW , the sports car of daily life
7	Replace	Accept	BMW , the sports car of future life
8	Insert	Accept	BMW , the sports car of the future life
9	Delete	Reject	BMW , the sports car of the future life
10	Delete	Accept	BMW , the sports car of the future life
11	[Output]		BMW , the sports car of the future

Miao et al., CGMH: Constrained Sentence Generation by Metropolis-Hastings Sampling, in AAAI, 2019.

Sampling in Sentence Space

CGMH performs Metropolis-Hastings sampling directly in sentence space:

Step	Action	Acc/Rej	Sentences
0	[Input]		BMW sports

Miao et al., CGMH: Constrained Sentence Generation by Metropolis-Hastings Sampling, in AAAI, 2019.

Sampling in Sentence Space

CGMH performs Metropolis-Hastings sampling directly in sentence space:

Step	Action	Acc/Rej	Sentences
0	[Input]		BMW sports
1	Insert	Accept	BMW sports car

Miao et al., CGMH: Constrained Sentence Generation by Metropolis-Hastings Sampling, in AAAI, 2019.

Sampling in Sentence Space

CGMH performs Metropolis-Hastings sampling directly in sentence space:

Step	Action	Acc/Rej	Sentences
0	[Input]		BMW sports
1	Insert	Accept	BMW sports car
2	Insert	Accept	BMW the sports car

Miao et al., CGMH: Constrained Sentence Generation by Metropolis-Hastings Sampling, in AAAI, 2019.

Sampling in Sentence Space

CGMH performs Metropolis-Hastings sampling directly in sentence space:

Step	Action	Acc/Rej	Sentences
0	[Input]		BMW sports
1	Insert	Accept	BMW sports car
2	Insert	Accept	BMW the sports car
...	...	...	...

Miao et al., CGMH: Constrained Sentence Generation by Metropolis-Hastings Sampling, in AAAI, 2019.

Sampling in Sentence Space

CGMH performs Metropolis-Hastings sampling directly in sentence space:

Step	Action	Acc/Rej	Sentences
0	[Input]		BMW sports
1	Insert	Accept	BMW sports car
2	Insert	Accept	BMW the sports car
…	…	…	…
6	Insert	Accept	BMW , the sports car of daily life

Miao et al., CGMH: Constrained Sentence Generation by Metropolis-Hastings Sampling, in AAAI, 2019.

Sampling in Sentence Space

CGMH performs Metropolis-Hastings sampling directly in sentence space:

Step	Action	Acc/Rej	Sentences
0	[Input]		BMW sports
1	Insert	Accept	BMW sports car
2	Insert	Accept	BMW the sports car
...	...	...	...
6	Insert	Accept	BMW , the sports car of daily life
7	Replace	Accept	BMW , the sports car of future life

Miao et al., CGMH: Constrained Sentence Generation by Metropolis-Hastings Sampling, in AAAI, 2019.

Sampling in Sentence Space

CGMH performs Metropolis-Hastings sampling directly in sentence space:

Step	Action	Acc/Rej	Sentences
0	[Input]		BMW sports
1	Insert	Accept	BMW sports car
2	Insert	Accept	BMW the sports car
…	…	…	…
6	Insert	Accept	BMW , the sports car of daily life
7	Replace	Accept	BMW , the sports car of future life
8	Insert	Accept	BMW , the sports car of the future life

Miao et al., CGMH: Constrained Sentence Generation by Metropolis-Hastings Sampling, in AAAI, 2019.

Sampling in Sentence Space

CGMH performs Metropolis-Hastings sampling directly in sentence space:

Step	Action	Acc/Rej	Sentences
0	[Input]		BMW sports
1	Insert	Accept	BMW sports car
2	Insert	Accept	BMW the sports car
…	…	…	…
6	Insert	Accept	BMW , the sports car of daily life
7	Replace	Accept	BMW , the sports car of future life
8	Insert	Accept	BMW , the sports car of the future life
9	Delete	Reject	BMW , the sports car of the future life

Miao et al., CGMH: Constrained Sentence Generation by Metropolis-Hastings Sampling, in AAAI, 2019.

Sampling in Sentence Space

CGMH performs Metropolis-Hastings sampling directly in sentence space:

Step	Action	Acc/Rej	Sentences
0	[Input]		BMW sports
1	Insert	Accept	BMW sports car
2	Insert	Accept	BMW the sports car
…	…	…	…
6	Insert	Accept	BMW , the sports car of daily life
7	Replace	Accept	BMW , the sports car of future life
8	Insert	Accept	BMW , the sports car of the future life
9	Delete	Reject	BMW , the sports car of the future life
10	Delete	Accept	BMW , the sports car of the future life

Miao et al., CGMH: Constrained Sentence Generation by Metropolis-Hastings Sampling, in AAAI, 2019.

Sampling in Sentence Space

CGMH performs Metropolis-Hastings sampling directly in sentence space:

Step	Action	Acc/Rej	Sentences
0	[Input]		BMW sports
1	Insert	Accept	BMW sports car
2	Insert	Accept	BMW the sports car
…	…	…	…
6	Insert	Accept	BMW , the sports car of daily life
7	Replace	Accept	BMW , the sports car of future life
8	Insert	Accept	BMW , the sports car of the future life
9	Delete	Reject	BMW , the sports car of the future life
10	Delete	Accept	BMW , the sports car of the future life
11	[Output]		BMW , the sports car of the future

Miao et al., CGMH: Constrained Sentence Generation by Metropolis-Hastings Sampling, in AAAI, 2019.

CGMH

CGMH performs constrained generation by:

1. Pretrain Language Model prob;
2. Start from a initial sentence;
3. Propose a new action and accept/reject the action.

Pretained LM in Target Distribution

- We set the stationary distribution as:

$$\pi(x) \ = P(x) \cdot P_C(x)$$

  - $P\ (x) = \prod_t P(x_t|x_{0:t-1})$ is the probability of sentence in a general-purpose language model.
  - $P_C(x)$= $\prod_{\mathrm{i}} P_C^i(x)$ is the indicator function showing whether constraints are satisfied.

CGMH: Action Proposal

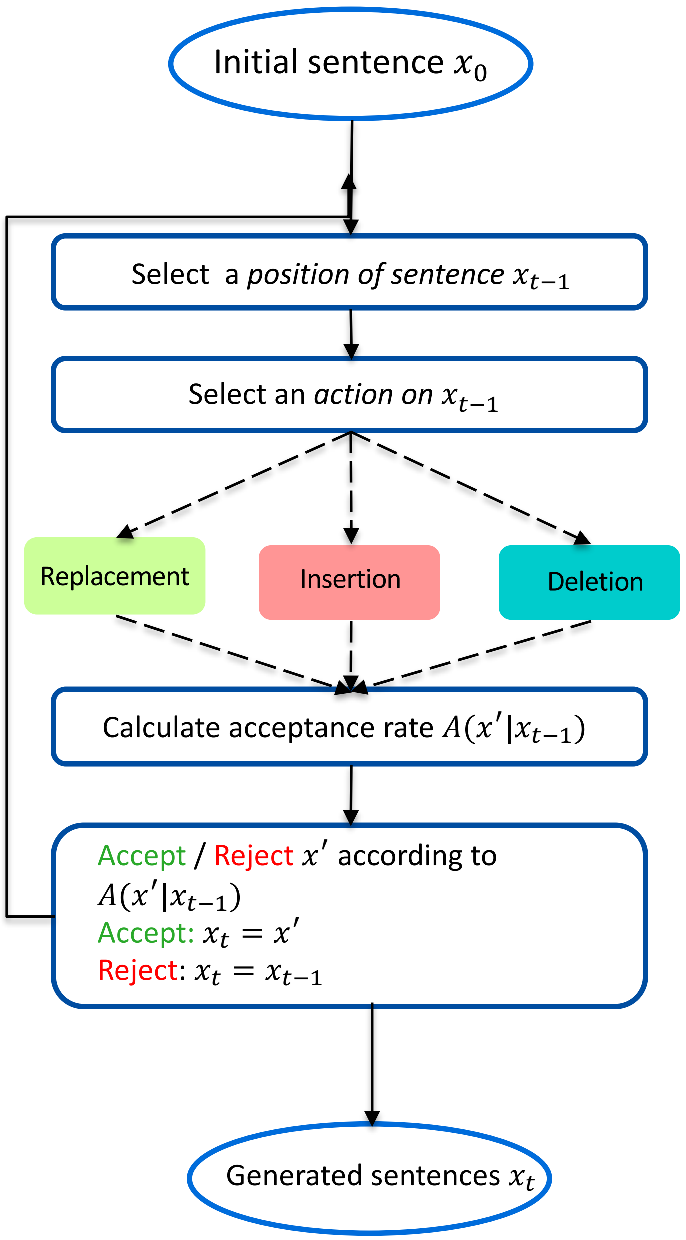
- We use MH algorithm to sample from $\pi(x)$
  - From a sentence x_{t-1}, we propose an action on one word of x_{t-1}.
  - Actions include:
    1. Replacement: change a word to another one
    2. Insertion: add a word
    3. Deletion: remove a word

CGMH: Acceptance Ratio

- Calculate the acceptance rate:

$$A(x'|x_{t-1}) = \min(1, \frac{\pi(x')\cdot g(x_{t-1}|x')}{\pi(x_{t-1})\cdot g(x'|x_{t-1})})$$

- Accept x' with probability $A(x'|x_{t-1})$

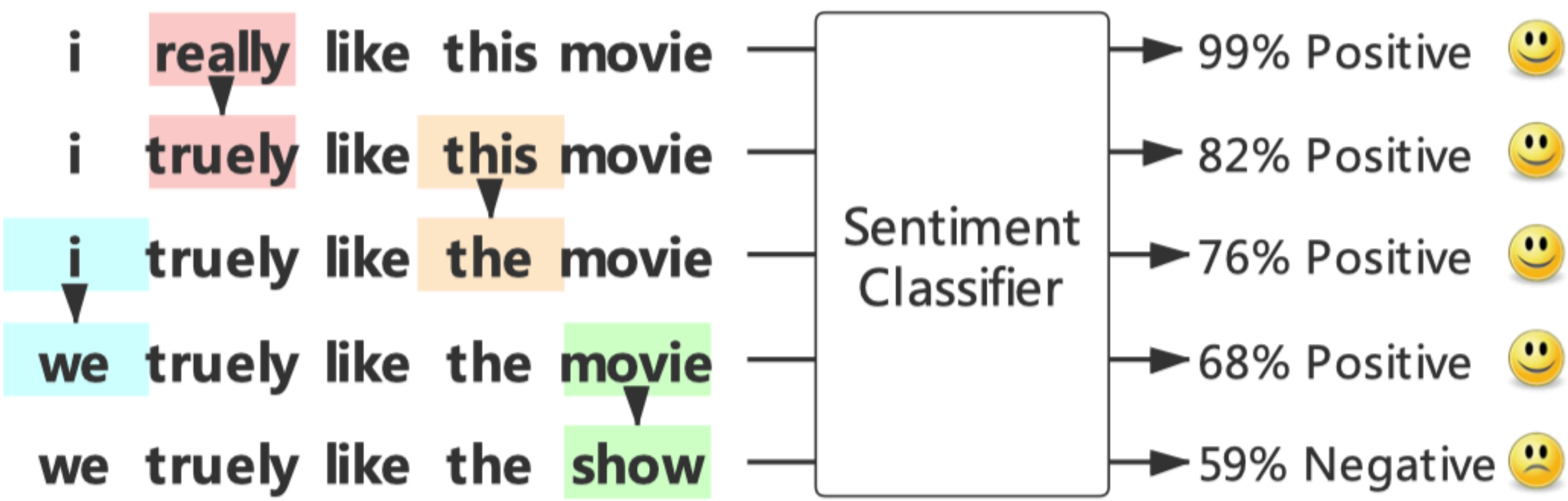


Adversarial Example for Text

Generating adversarial example for text is hard! Because the text space is discrete, which is non-trivial to apply adversarial gradients!

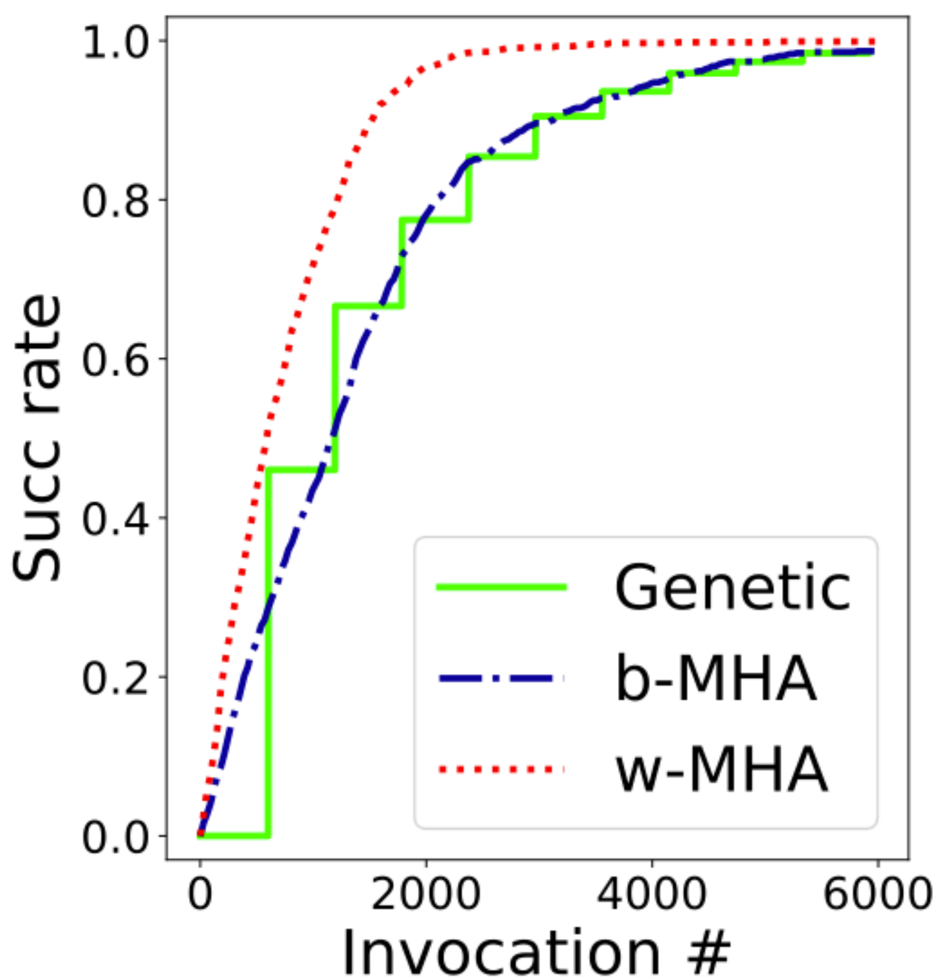
Zhang et al., Generating Fluent Adversarial Examples for Natural Languages, in ACL, 2019, short paper.

CGMH for Generating Fluent Adversarial Examples

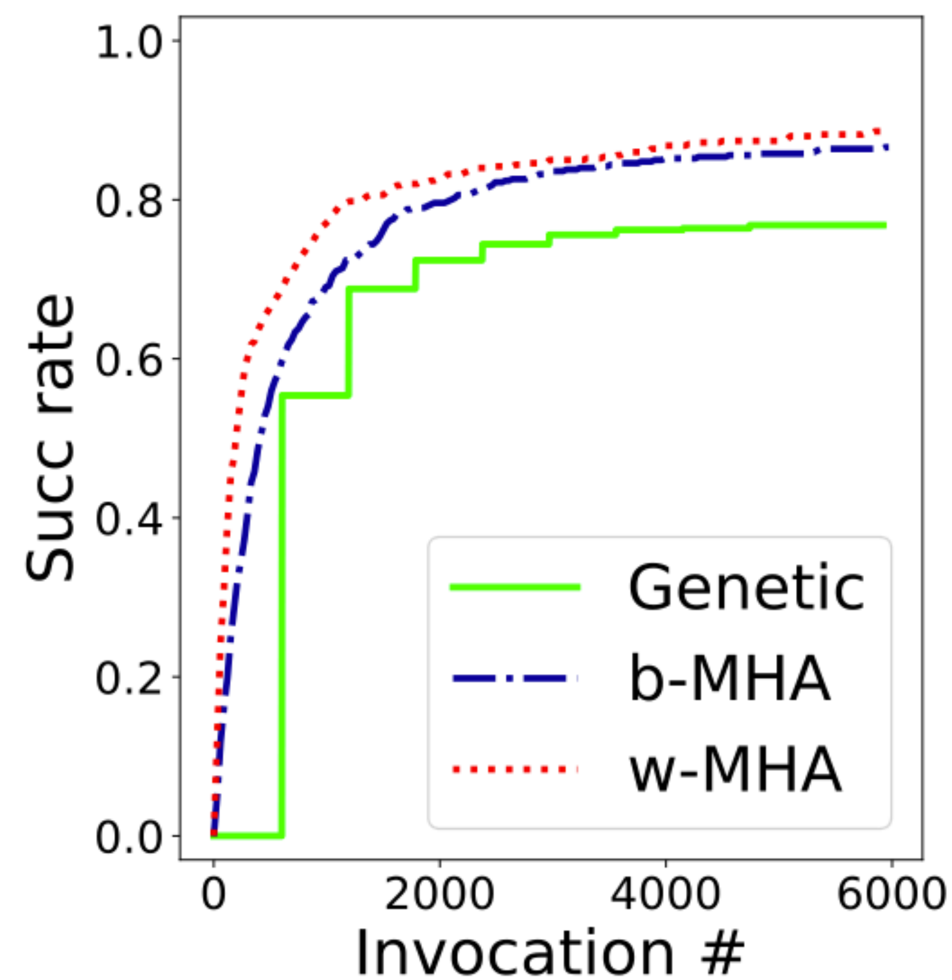


Zhang et al., Generating Fluent Adversarial Examples for Natural Languages, in ACL, 2019, short paper.

CGMH for Generating Fluent Adversarial Examples



(a) IMDB



(b) SNLI

Zhang et al., Generating Fluent Adversarial Examples for Natural Languages, in ACL, 2019, short paper.

CGMH for Generating Fluent Adversarial Examples

Task	Approach	Succ(%)	Invok#	PPL	α(%)
IMDB	Genetic	98.7	1427.5	421.1	–
	b-MHA	98.7	1372.1	385.6	17.9
	w-MHA	**99.9**	**748.2**	**375.3**	34.4
SNLI	Genetic	76.8	971.9	834.1	–
	b-MHA	86.6	681.7	358.8	9.7
	w-MHA	**88.6**	**525.0**	**332.4**	13.3

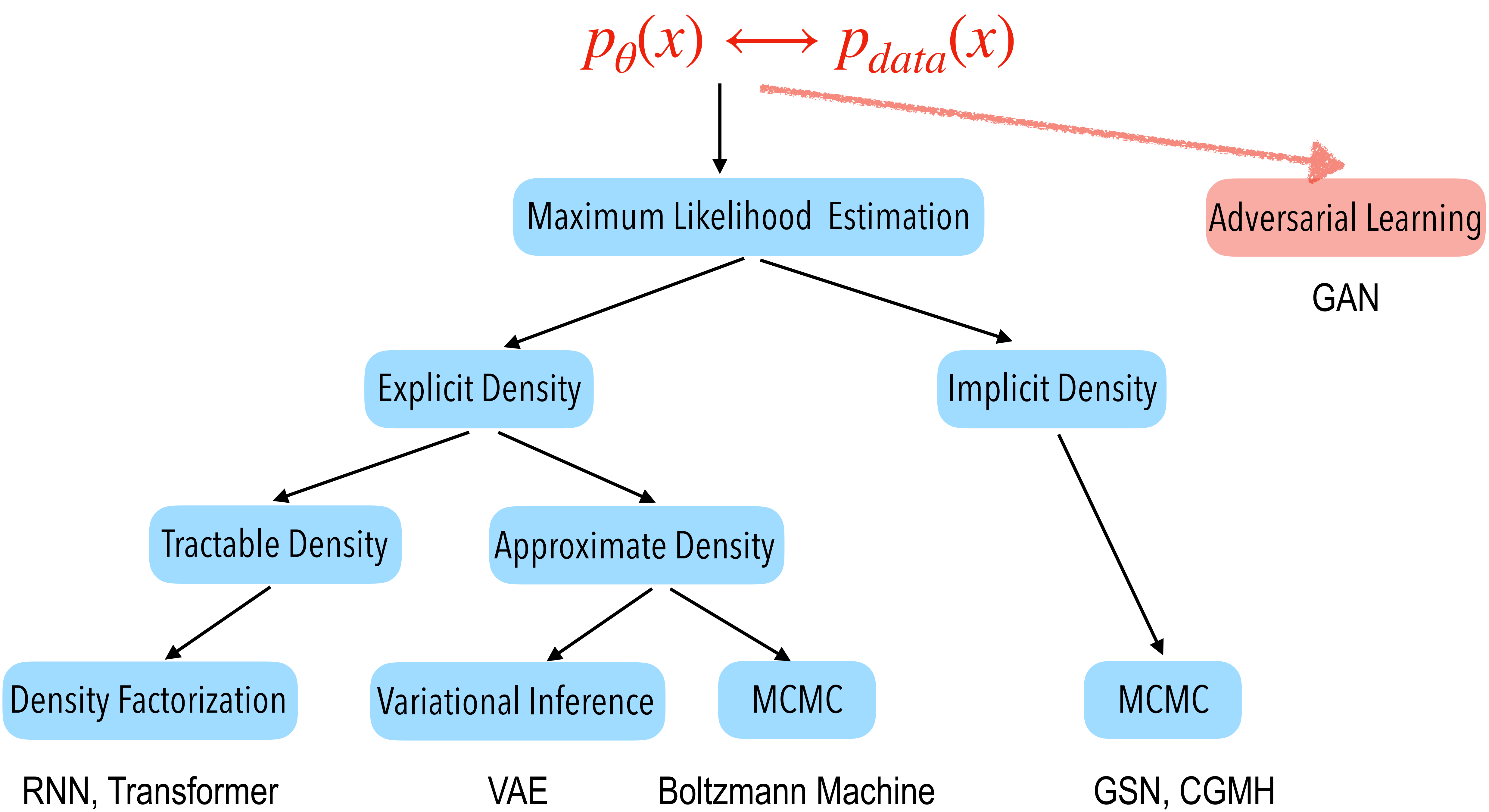
Zhang et al., Generating Fluent Adversarial Examples for Natural Languages, in ACL, 2019, short paper.

Part 6

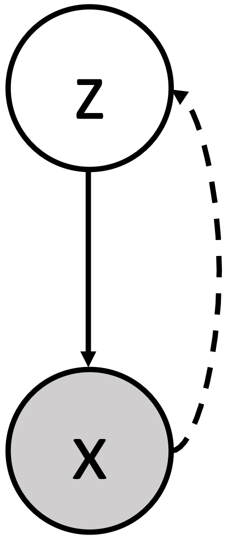
Text Generation by Generative Adversarial Networks

Generation without Maximum Likelihood Estimation

Taxonomy of DGM



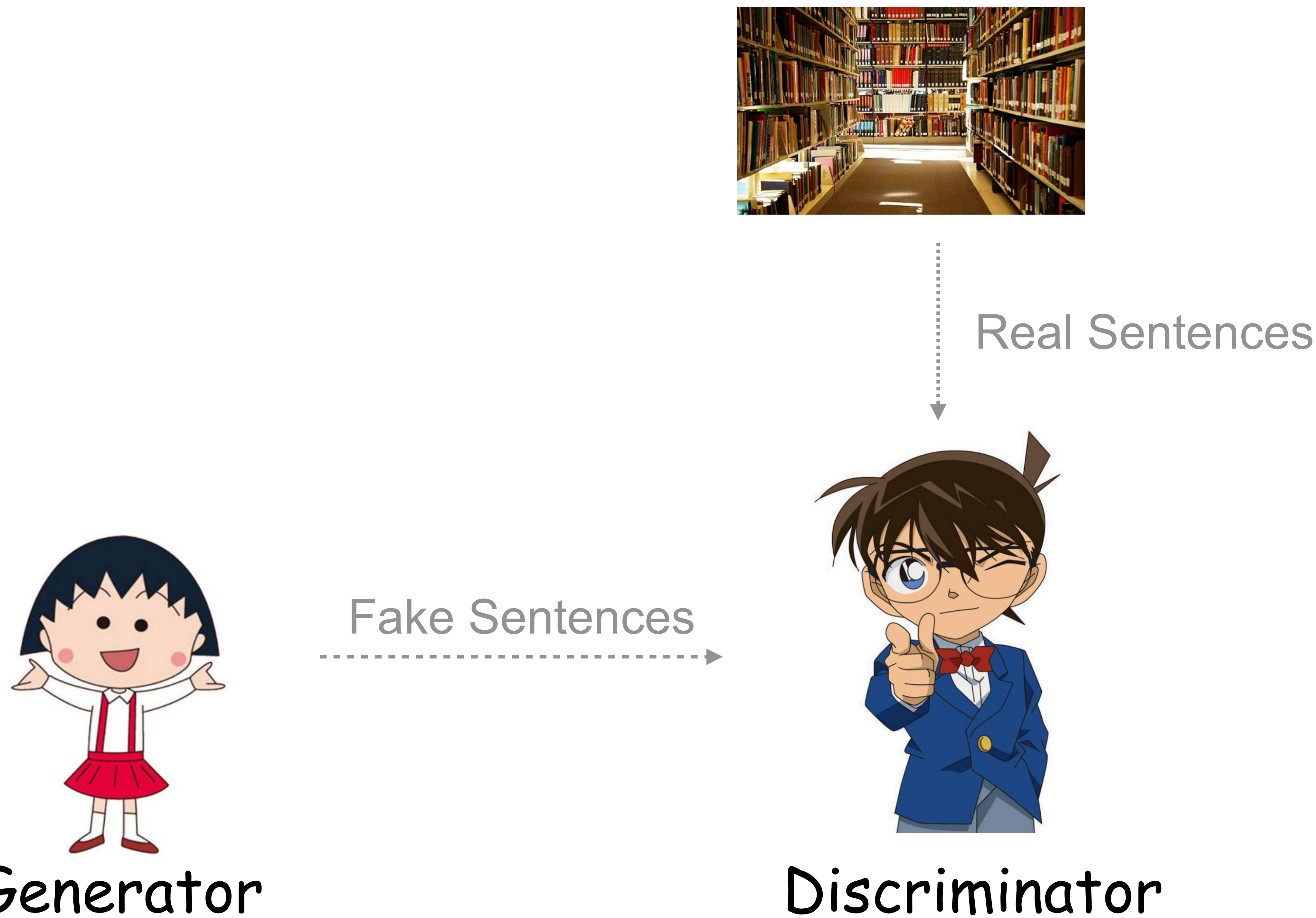
What's GAN ?



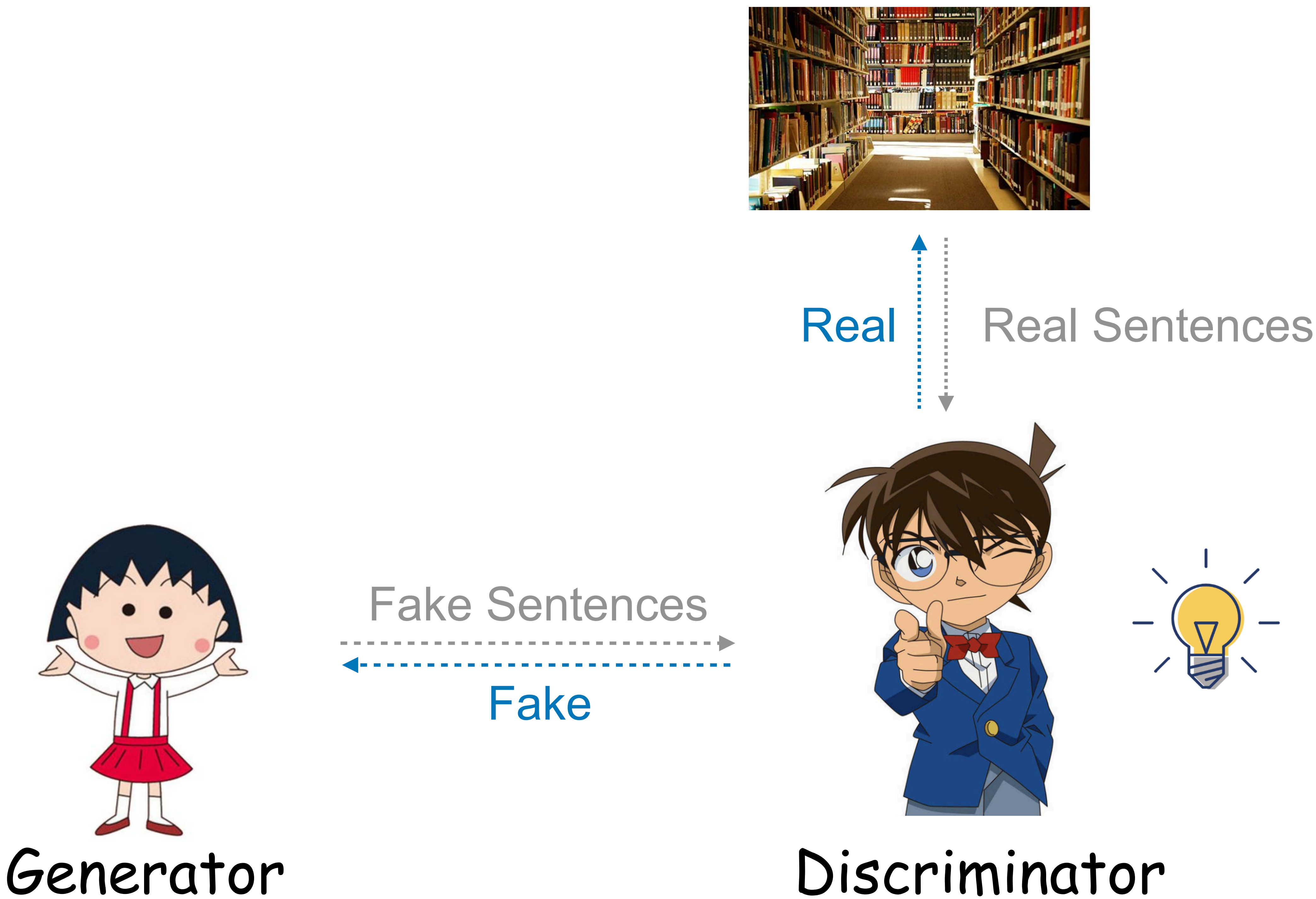
Generative Adversarial Networks:

$$
\begin{aligned}
\min_G \max_D L(D, G) &= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \\
&= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{x \sim p_g(x)}[\log(1 - D(x)]
\end{aligned}
$$

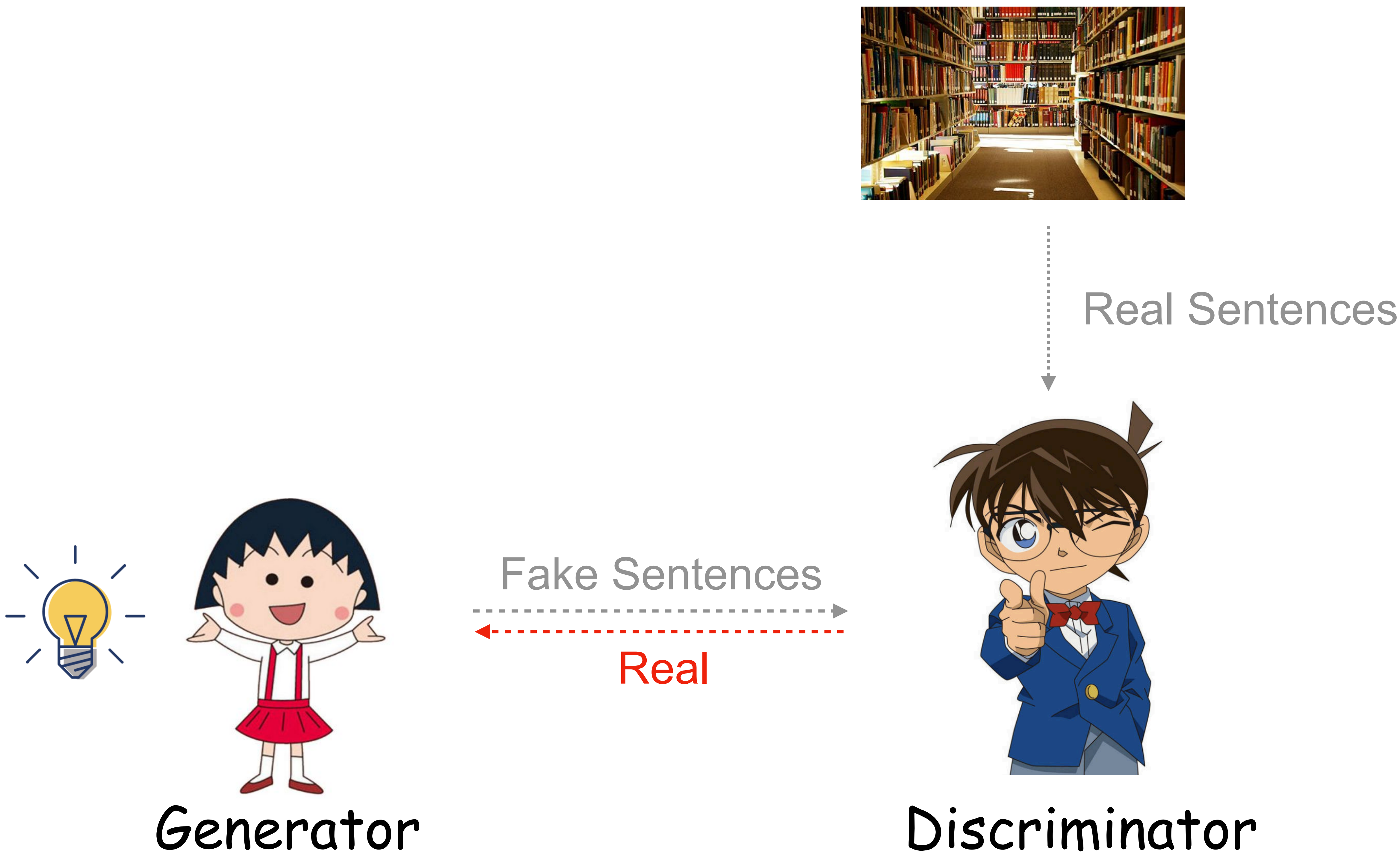
Generator VS. Discriminator



Generator VS. Discriminator



Generator VS. Discriminator



Objective Revisit

Generative Adversarial Networks:

$$\begin{aligned}\min_G \max_D L(D,G) &= \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \\ &= \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{x\sim p_g(x)}[\log(1 - D(x)]\end{aligned}$$

Essence of MLE

MLE = Minimizing KLD

Recall that for continuous distributions P and Q, the KL divergence is

$$KL(P||Q) = \int_x P(x) \log \frac{P(x)}{Q(x)} \, dx$$

In the limit (as $m \to \infty$), samples will appear based on the data distribution P_r, so

$$\begin{aligned}
\lim_{m\to\infty} \max_{\theta\in\mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} \log P_\theta(x^{(i)}) &= \max_{\theta\in\mathbb{R}^d} \int_x P_r(x) \log P_\theta(x) \, dx \\
&= \min_{\theta\in\mathbb{R}^d} - \int_x P_r(x) \log P_\theta(x) \, dx \\
&= \min_{\theta\in\mathbb{R}^d} \int_x P_r(x) \log P_r(x) \, dx - \int_x P_r(x) \log P_\theta(x) \, dx \\
&= \min_{\theta\in\mathbb{R}^d} KL(P_r \| P_\theta)
\end{aligned}$$

Derivations in order: limit of summation turns into integral, flip max to min by negating, add a constant that doesn't depends on θ, and apply definition of KL divergence.

GAN -> JSD

Derivation of GAN -> JSD

$$
\begin{aligned}
L(G, D^*) &= \int_x \left(p_r(x) \log(D^*(x)) + p_g(x) \log(1 - D^*(x)) \right) dx \\
&= \log \frac{1}{2} \int_x p_r(x) dx + \log \frac{1}{2} \int_x p_g(x) dx \\
&= -2 \log 2
\end{aligned}
$$

$$
\begin{aligned}
D_{JS}(p_r \| p_g) =& \frac{1}{2} D_{KL}(p_r || \frac{p_r + p_g}{2}) + \frac{1}{2} D_{KL}(p_g || \frac{p_r + p_g}{2}) \\
=& \frac{1}{2} \left(\log 2 + \int_x p_r(x) \log \frac{p_r(x)}{p_r + p_g(x)} dx \right) + \\
& \frac{1}{2} \left(\log 2 + \int_x p_g(x) \log \frac{p_g(x)}{p_r + p_g(x)} dx \right) \\
=& \frac{1}{2} \left(\log 4 + L(G, D^*) \right)
\end{aligned}
$$

$$
L(G, D^*) = 2 D_{JS}(p_r \| p_g) - 2 \log 2
$$

MLE VS. GAN

Maximum Likelihood Estimation:

$$min\ \mathbb{E}_{x\sim p_{data}}[-log\ p_\theta(x\,|\,y)]$$

$$p_\theta(x\,|\,y) = \prod_{i=1}^{n} \quad p_\theta(x_i\,|\,x_1, x_2, \ldots, x_{i-1}, y) = \prod_{i=1}^{n} p_\theta(x_i\,|\,x_{<i}, y)$$

Generative Adversarial Networks:

$$\begin{aligned}\min_G \max_D L(D,G) &= \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \\ &= \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{x\sim p_g(x)}[\log(1-D(x)]\end{aligned}$$

KLD VS. JSD

Motivation of GAN for Text Generation

- Exposure Bias
  - Discrepancy between training and inference
- Multi-Modal Output
  - GAN may better address the multi-modal output than MLE training

GAN for Text

Text is discrete, hard to propagate gradients from D to G !

BackPropagation Fails

- Sentence is discrete, BP fails in such case
  - Policy Gradient
  - Gumbel Softmax

Observations

GAN tend to generate less diverse sentences than MLE training.

Thank You!