A sequence doesn't have to be generated sequentially nor in a monotonic order

Kyunghyun Cho
New York University
Facebook AI Research
CIFAR Associate Fellow

Non-Monotonic Generation

Recursive generation of a sequence

- Define a one-step generator $\pi : (V \cup \{<\text{empty}>\})^L \times \mathcal{X} \rightarrow [1:L] \times (V \cup \{<\text{terminate}>\})$
  - Input: $Y^{t-1} = (y_1, <\text{empty}>, y_3, \ldots, <\text{empty}>)$ and X
  - Output: $a^t = (<\text{loc}>^t, <\text{symbol}>^t)$
- Recursively apply the generator until it terminates

$$
\begin{aligned}
&1: Y = (<\text{empty}>, \ldots, <\text{empty}>) \\
&2: \text{while true} \\
&3: \qquad l, v \sim \pi(<\text{loc}>, <\text{symbol}>, Y|X) \\
&4: \qquad \text{if } v = <\text{terminate}> \text{ then break} \\
&5: \qquad Y[l] = v
\end{aligned}
$$

- A generic principle behind learning to generate a structured object

Another perspective: probabilistic modeling

- The goal is to estimate $p(y_1, y_2, \ldots, y_L|X)$
  - And, subsequently to find $\arg\max_Y \log p(y_1, y_2, \ldots, y_L|X)$
- From recursive generation to probabilistic sequence learning
  - the sum of the probabilities of generating the target Y via all possible trajectories
  - i.e. marginalizing out all possible generation orders.

$$p(y_1, \ldots, y_L|X) = \sum_{(<\text{loc}>_1, \ldots, <\text{loc}>_L)} \prod_{l=1}^{L} \pi(<\text{loc}>_l, Y[<\text{loc}>_l]|Y[<\text{loc}>_1], \ldots, Y[<\text{loc}>_{l-1}], X)$$

- Exact marginalization is intractable, but we will stick to this formalism for now

A (monotonic) autoregressive model is a special case

- The location of a new token advances by one in a monotonic order

$$\pi(<\text{loc}>_l, Y[l] | Y[<\text{loc}>_{<l}], X) = \pi(<\text{loc}>_l | Y[<\text{loc}>_{<l}], X)\pi(Y[l] | <\text{loc}>_l, Y[<\text{loc}>_{<l}], X),$$

where

$$\pi(<\text{loc}>_l | Y[<\text{loc}>_{<l}], X) = \begin{cases} 1, & \text{if } <\text{loc}>_l = <\text{loc}>_{l-1} + 1 \\ 0, & \text{otherwise} \end{cases}$$

- This greatly simplifies learning due to trivial marginalization: no combinatorial search

$$p(y_1, \ldots, y_L | X) = \sum_{(<\text{loc}>_1, \ldots, <\text{loc}>_L)} \prod_{l=1}^{L} \pi(<\text{loc}>_l, Y[<\text{loc}>_l] | Y[<\text{loc}>_1], \ldots, Y[<\text{loc}>_{l-1}], X)$$

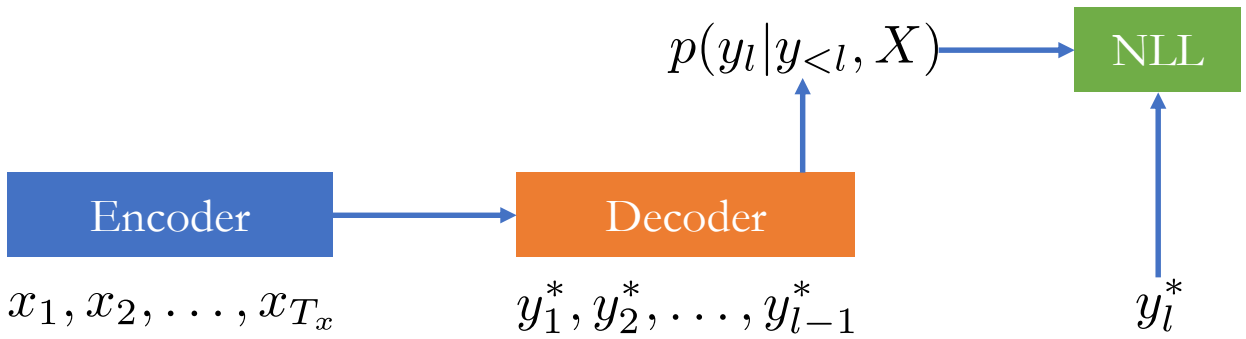
$$= \prod_{l=1}^{L} \pi(Y[l] | Y[< l], X)$$

An autoregressive model is a special case – (2)

- Autoregressive modeling = a sequence of classification
  - Input: prefix and source
  - Target: a correct next symbol

$$\log p(y_1^*, \ldots, y_L^* | X) = \sum_{l=1}^{L} \log p(y_l^* | y_{<l}^*, X)$$

- Graphical illustration



Works really well!

- Machine translation
  - Any commercial machine translation system uses a monotonic autoregressive sequence model.
  - A few of of my favorite recent achievements are

We introduce our efforts towards building a universal neural machine translation (NMT) system capable of translating between any language pair. We set a milestone towards this goal by building a single massively multilingual NMT model handling 103 languages trained on over 25 billion examples. Our system demonstrates effective transfer learning ability, significantly improving translation quality of low-resource languages, while keeping high-resource language translation quality on-par with competitive bilingual baselines. We provide in-depth analysis of various as
building that are cru

Arivazhagan, Bapna, Firat et al. [2019]

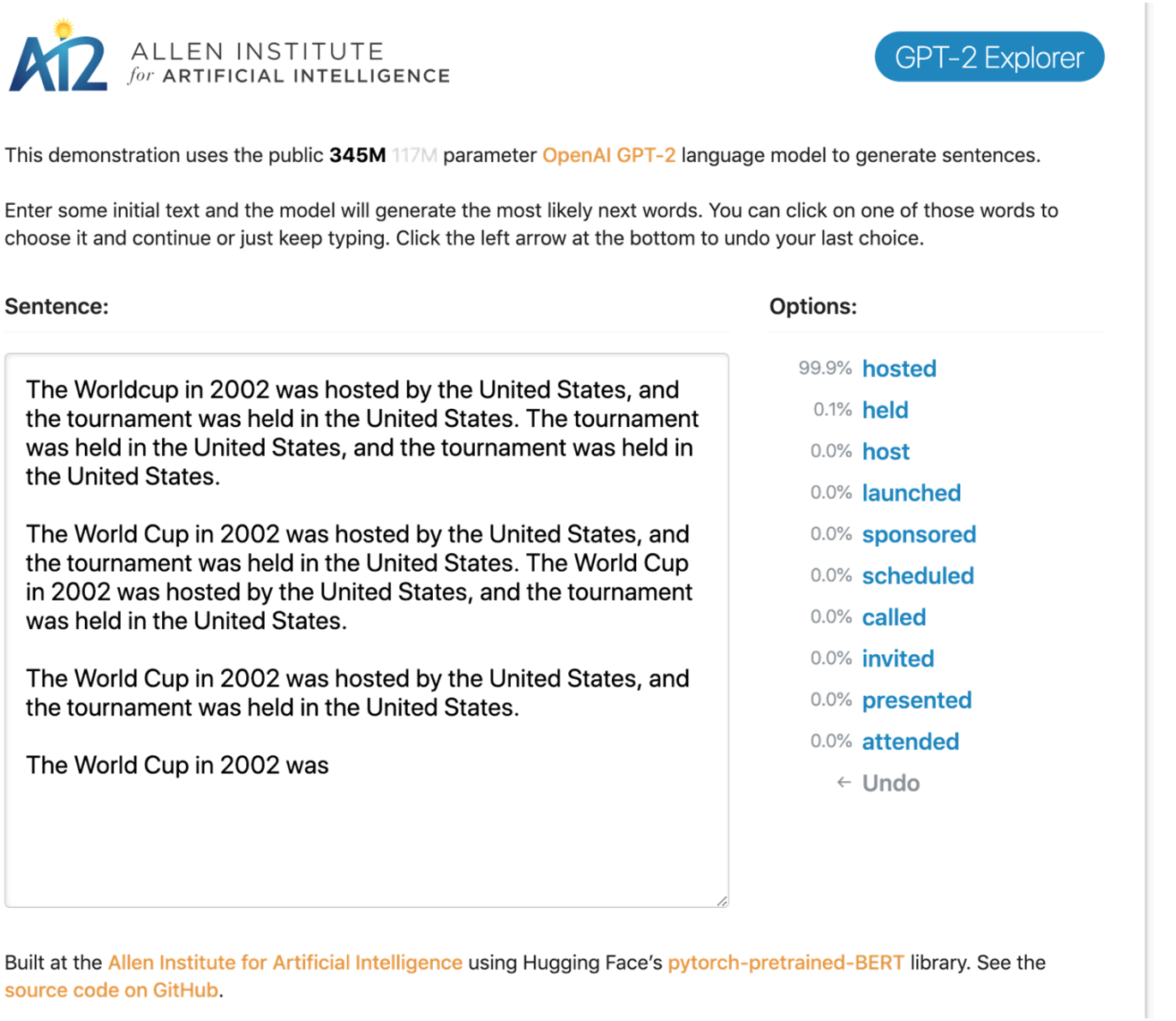
This paper describes Facebook AI's submission to the WAT 2019 Myanmar-English translation task (Nakazawa et al., 2019). Our baseline systems are BPE-based transformer models. We explore methods to leverage monolingual data to improve generalization, including self-training, back-translation and their combination. We further improve results by using noisy channel re-ranking and ensembling. We demonstrate that these techniques can significantly improve not only a system trained with additional monolingual data, but even the baseline system trained exclusively on the provided small parallel dataset. Our system ranks first in both directions according to human evaluation and BLEU, with a gain of over 8 BLEU points above the second best system.

Chen, Shen et al. [2019]

African languages are numerous, complex and low-resourced. The datasets required for machine translation are difficult to discover, and existing research is hard to reproduce. Minimal attention has been given to machine translation for African languages so there is scant research regarding the problems that arise when using machine translation techniques. To begin addressing these problems, we trained models to translate English to five of the official South African languages (Afrikaans, isiZulu, Northern Sotho, Setswana, Xitsonga), making use of modern neural machine translation techniques. The results obtained show the promise of using neural machine translation techniques for African languages. By providing reproducible publicly-available data, code and results, this research aims to provide a starting point for other researchers in African machine translation to compare to and build upon.

Martinus & Abbott [2019]

What's the biggest failure mode?



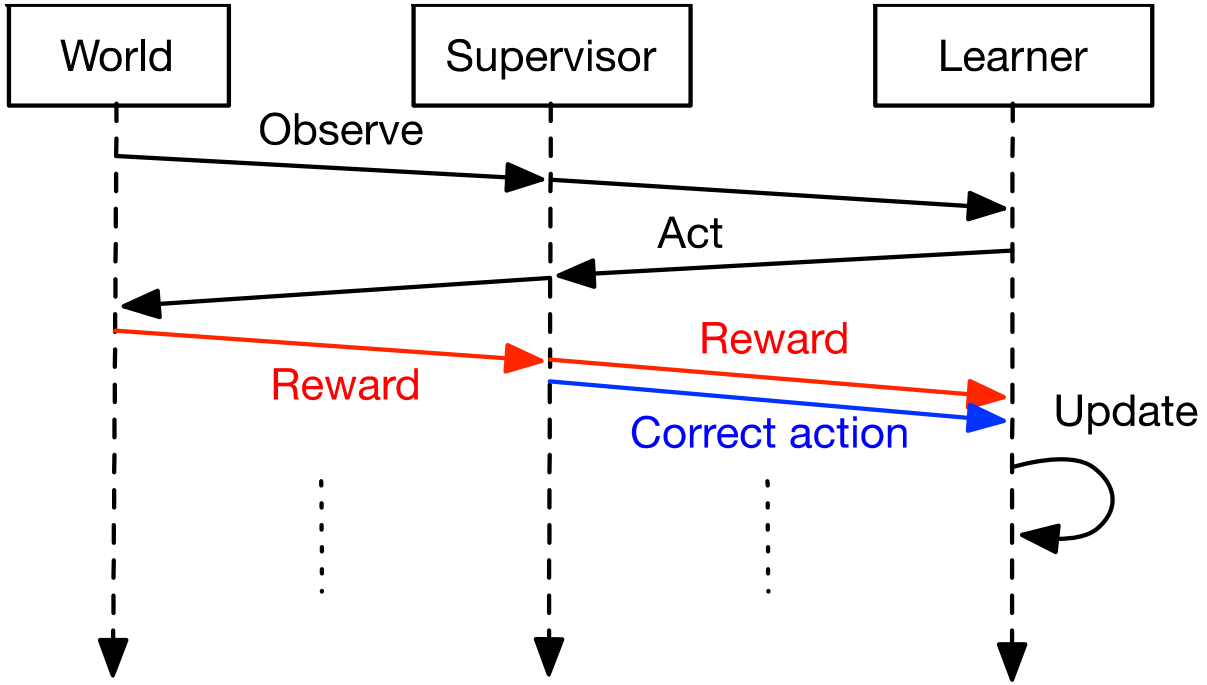
"Exposure bias" a problem since 1995

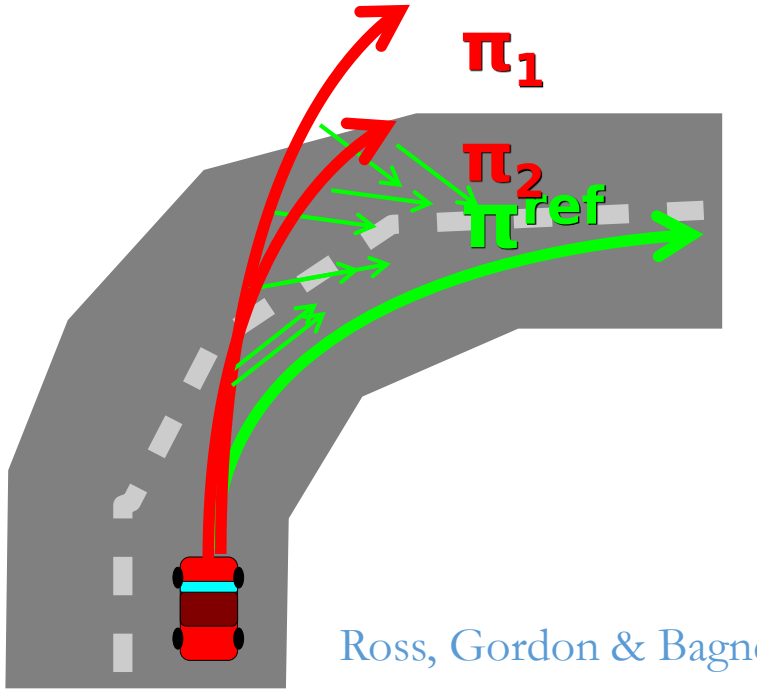
ALVINN:
AN AUTONOMOUS LAND VEHICLE IN A NEURAL NETWORK

Dean A. Pomerleau

There are difficulties involved with training "on-the-fly" with real images. If the network is not presented with sufficient variability in its training exemplars to cover the conditions it is likely to encounter when it takes over driving from the human operator, it will not develop a sufficiently robust representation and will perform poorly. In addition, the network must not solely be shown examples of accurate driving, but also how to recover (i.e. return to the road center) once a mistake has been made. Partial initial training on a variety of simulated road images should help eliminate these difficulties and facilitate better performance.

We want to learning from an expert
(not from their demonstrations)





Ross, Gordon & Bagnell, 2011

Imitation learning in recursive generation

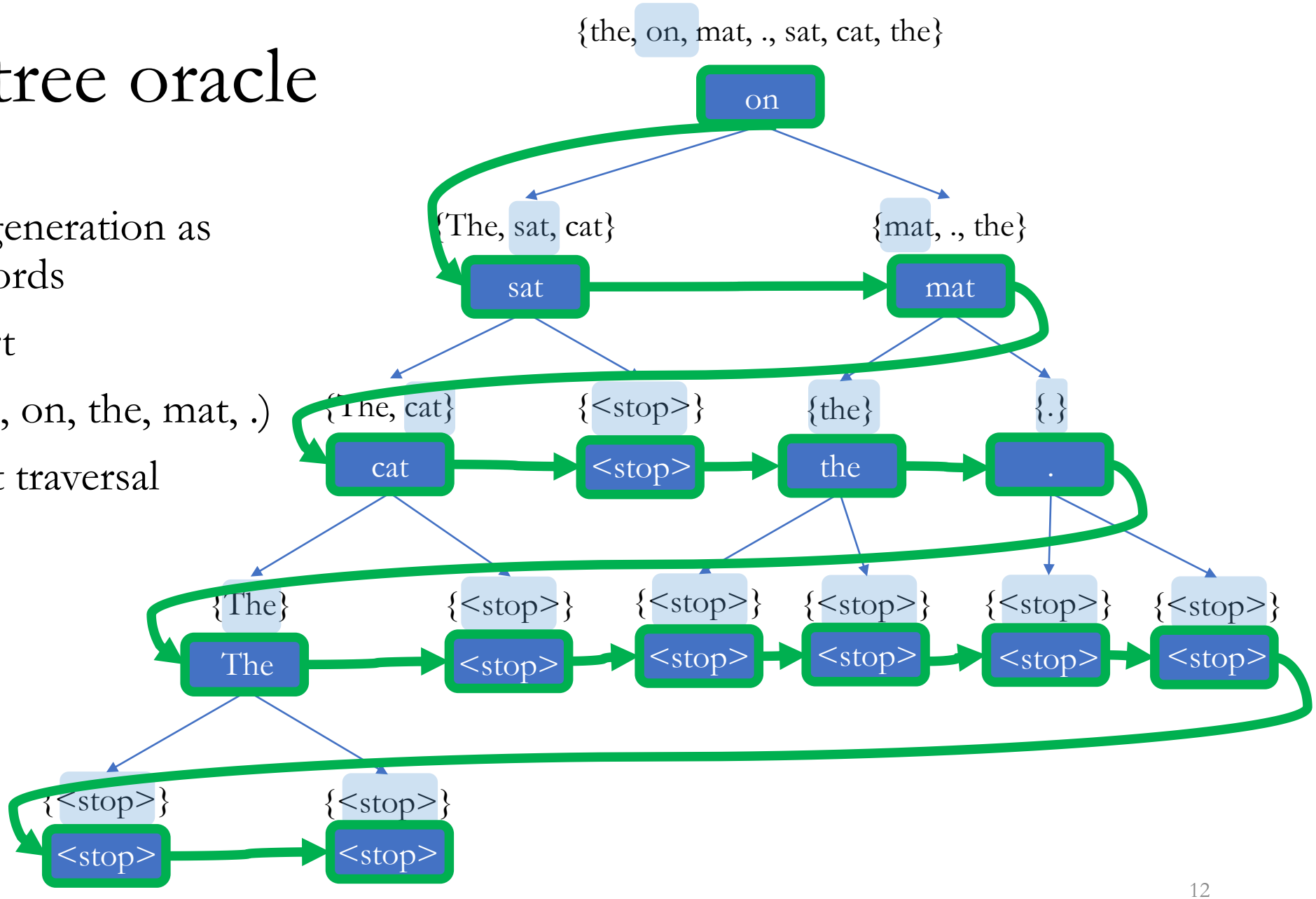
- If I knew the ground-truth target sequence Y^*, what would I have done?

$1 : Y = (<\text{empty}>, \ldots, <\text{empty}>)$
$2 : \text{while true}$
$3 : \quad l, v \sim \pi(<\text{loc}>, <\text{symbol}>, Y|X)$? $l, v \sim \pi^*(<\text{loc}>, <\text{symbol}>, Y|X)$
$4 : \quad \text{if } v = <\text{terminate}> \text{ then break}$
$5 : \quad Y[l] = v$

- The whole problem is reduced to a series of supervised learning
- We will consider two types of oracles here: binary and n-ary tree oracles

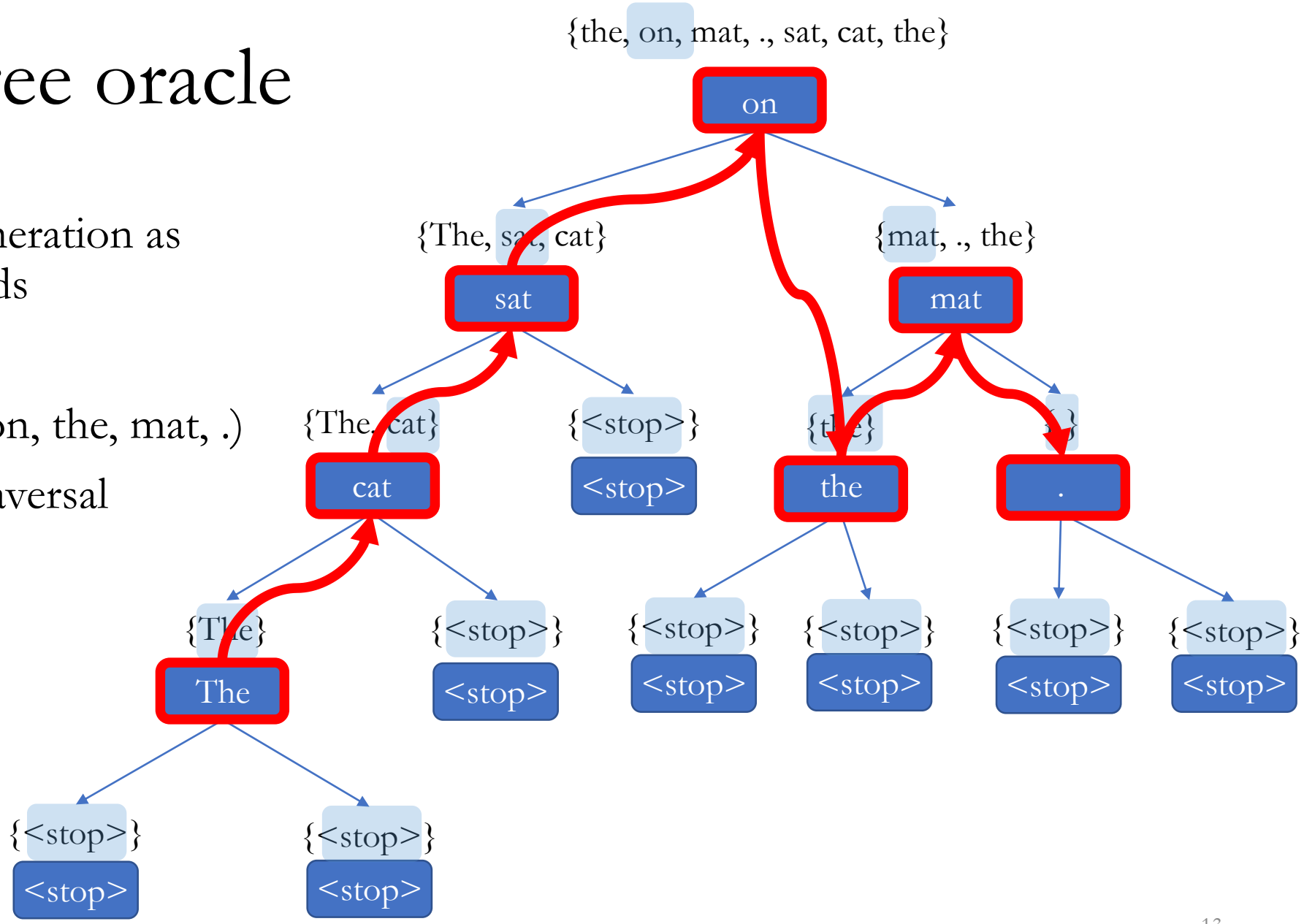
A binary tree oracle

- Consider sequence generation as ordering a bag of words
- Inspired by quicksort
- Target: (The, cat, sat, on, the, mat, .)
- Generation: level-set traversal

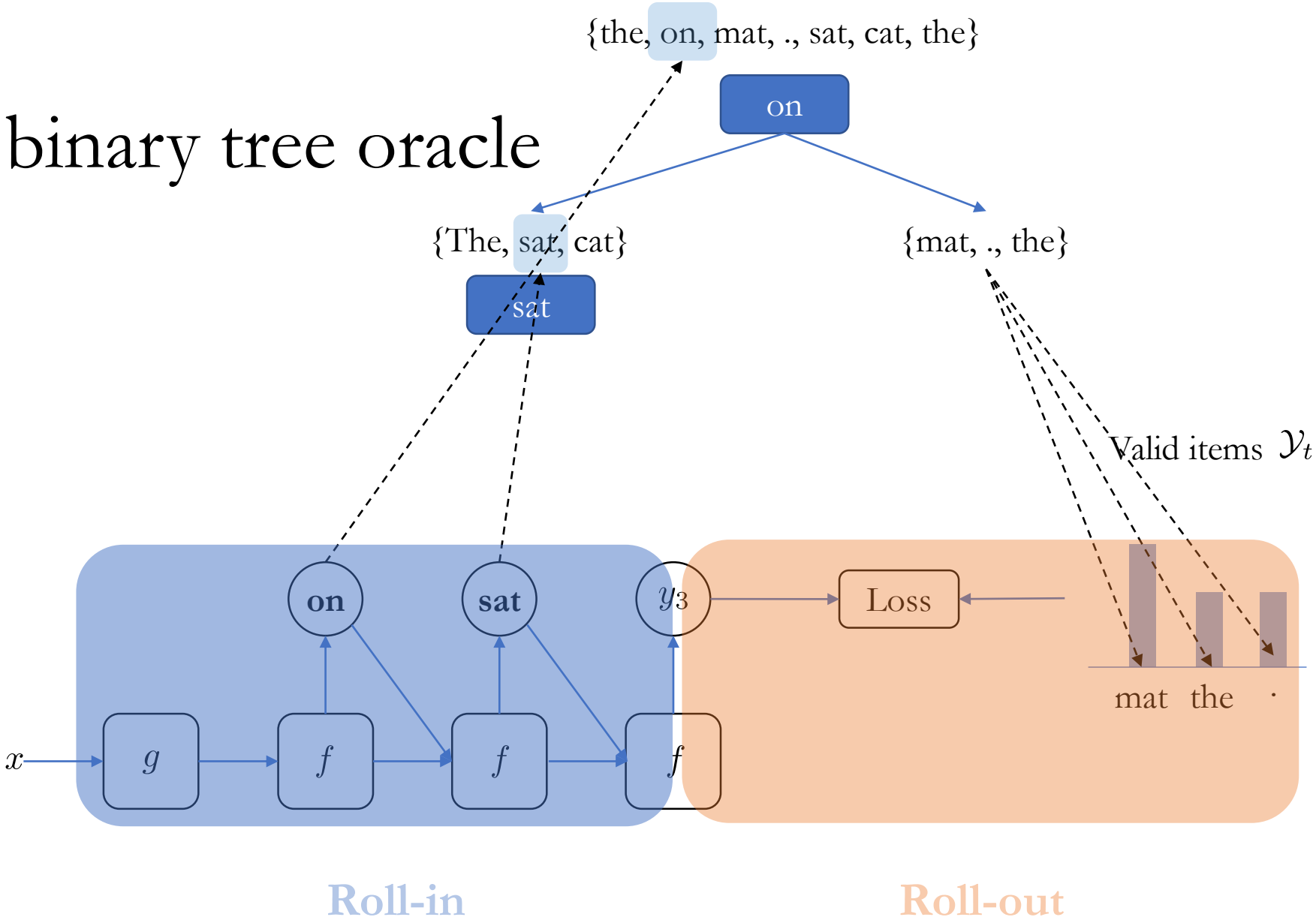


A binary tree oracle

- Consider sequence generation as ordering a bag of words
- Inspired by quicksort
- Target: (The, cat, sat, on, the, mat, .)
- Collection: in-order traversal



A binary tree oracle



An oracle to a reference policy – (1)

- An oracle tell us a set of valid actions (tokens)
  - (<empty1>, “on”, <empty2>, “sat”, <empty3>)→(<empty3>, {mat, ., the})
- There are many possible corresponding reference policies that satify
  - Positive probabilities assigned to all the valid actions
  - Zero probabilities assigned to all the invalid actions

$$\pi^*(a) = \begin{cases} p_a, & \text{if } a \in \mathcal{A}_{\text{valid}} \\ 0, & \text{otherwise} \end{cases}$$

- We impose our *prior* preference by designing an appropriate reference policy

An oracle to a reference policy – (2)

- Uniform reference policy

$$\pi^*_{\text{uniform}}(a) = \begin{cases} 1/|\mathcal{A}_{\text{valid}}|, & \text{if } a \in \mathcal{A}_{\text{valid}} \\ 0, & \text{otherwise} \end{cases}$$

- Coaching reference policy [He et al., 2012]

$$\pi^*_{\text{coaching}}(a) \propto \pi(a) \cdot \pi^*_{\text{uniform}}(a)$$

- Annealed coaching reference policy [Welleck et al., 2019; Emelianenko et al., 2019]

$$\pi^*_{\text{annealed}}(a) = \beta\pi^*_{\text{uniform}}(a) + (1-\beta)(\pi(a) \cdot \pi^*_{\text{uniform}}(a)/Z)$$

A roll-in policy

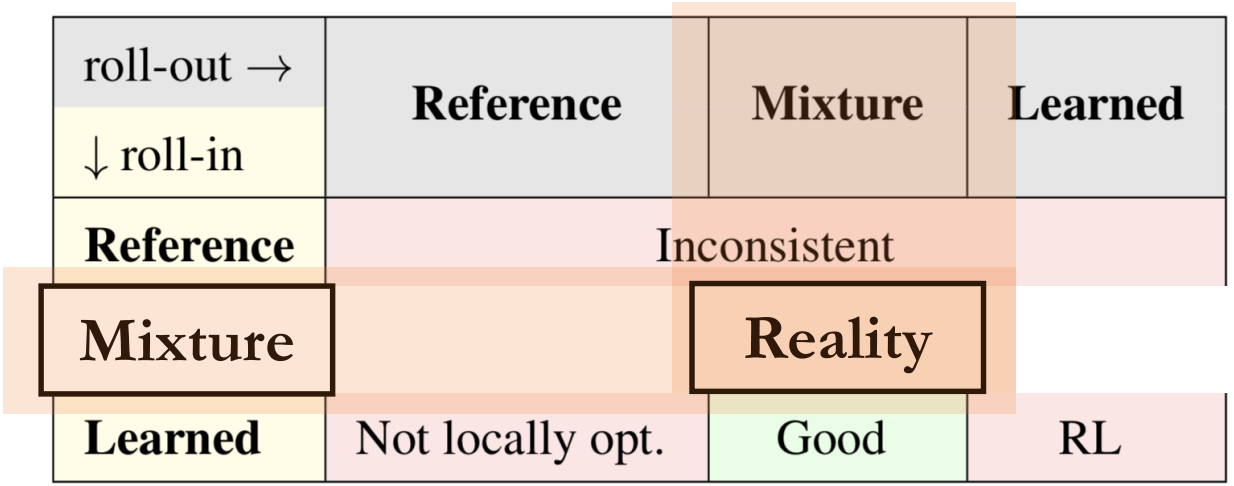
- LOLS [Chang et al., 2015] states that

roll-out → / ↓ roll-in	**Reference**	**Mixture**	**Learned**
Reference	Inconsistent		
Learned	Not locally opt.	Good	RL

- In practice, roll-in with a learned policy easily enters a state space from which recovery is too difficult or inefficient: i.e., not good.

A roll-in policy

- LOLS [Chang et al., 2015] states that



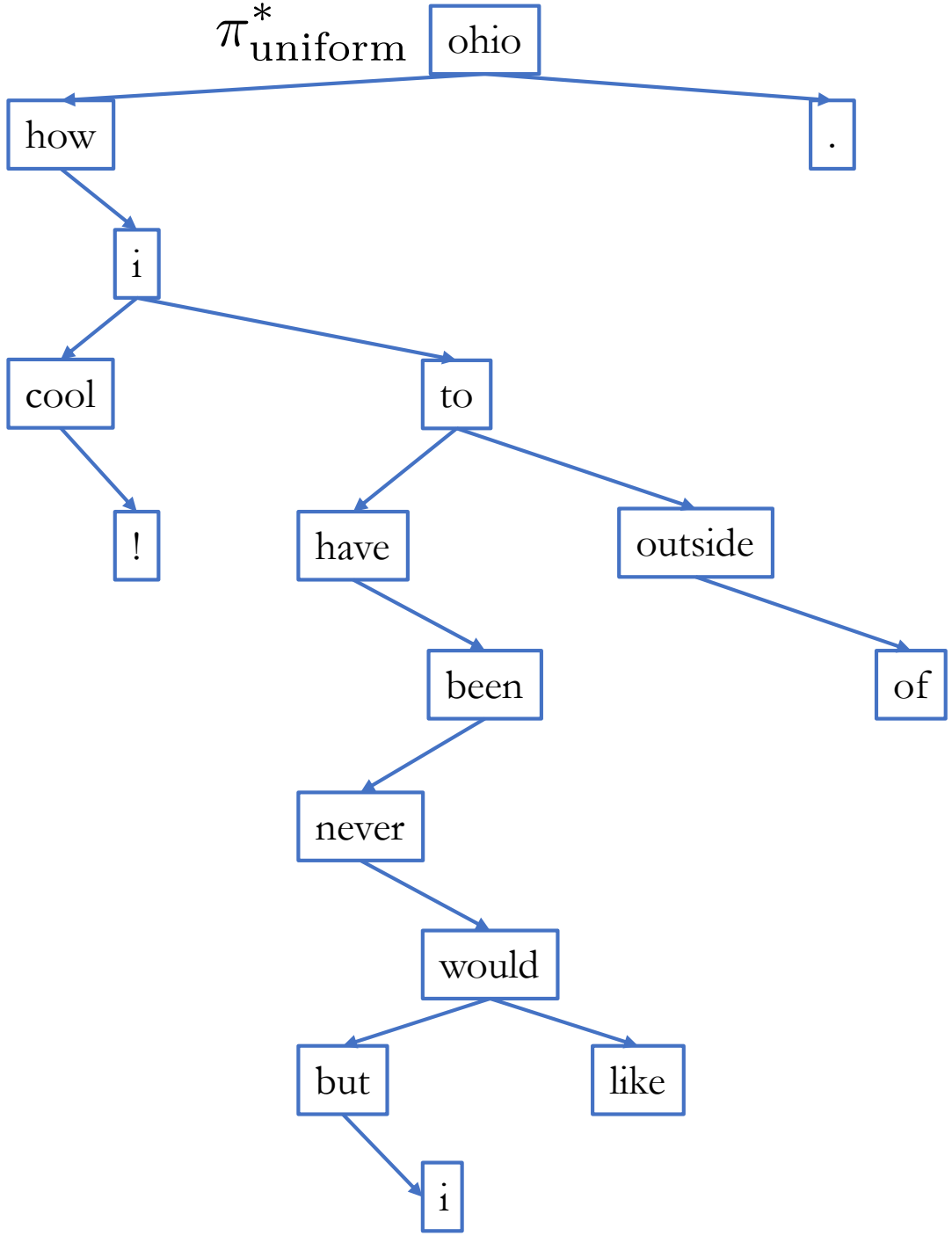
- Using a coaching or annealed coaching strategy for roll-in worked well.
  - It does ignore the possibility of visiting unseen states in the test time
  - Looks less problematic than not being able to learn at all

Learning [Welleck et al., 2019]

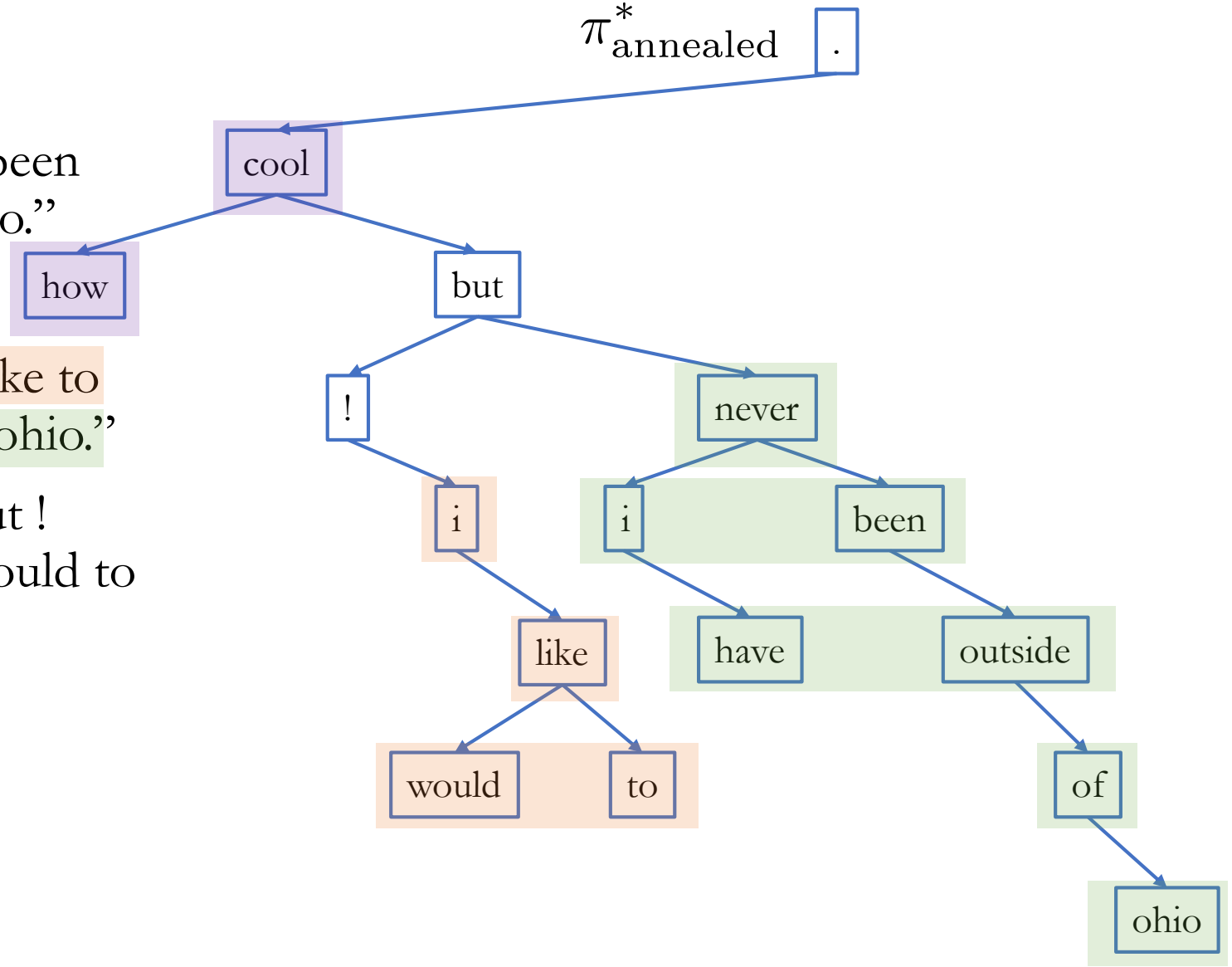
- Given: $Y^* = (y_1^*, y_2^*, \ldots, y_T^*), X,$ and π^*
- Initialize: π_0

1. For $n = 1$ to N
    1. Roll in with π^*_{annealed} to collect $\{(y_1, \ldots, y_t, X)\}_{t=1}^T$
    2. Set $l = 0$
    3. For $t = 1$ to T
        1. $l \leftarrow l + \text{KL}(\pi^*_{\text{annealed}}(\cdot|y_{\leq t}, X) \| \pi_{n-1}(\cdot|y_{\leq t}, X))$
    4. $\pi_n = \pi_0 - \eta \nabla_\pi l(\pi_0)$
2. Return $(\pi_1, \ldots, \underline{\pi_N})$

- **Target**: “how cool! i have never been outside of ohio but I would like to.”
- **Input**: a bag of words
- **Generated**: “how cool! i have never but i would like been to outside of ohio.”
- **Generated order**: “ohio how. i cool to ! have outside been of never would but like i”

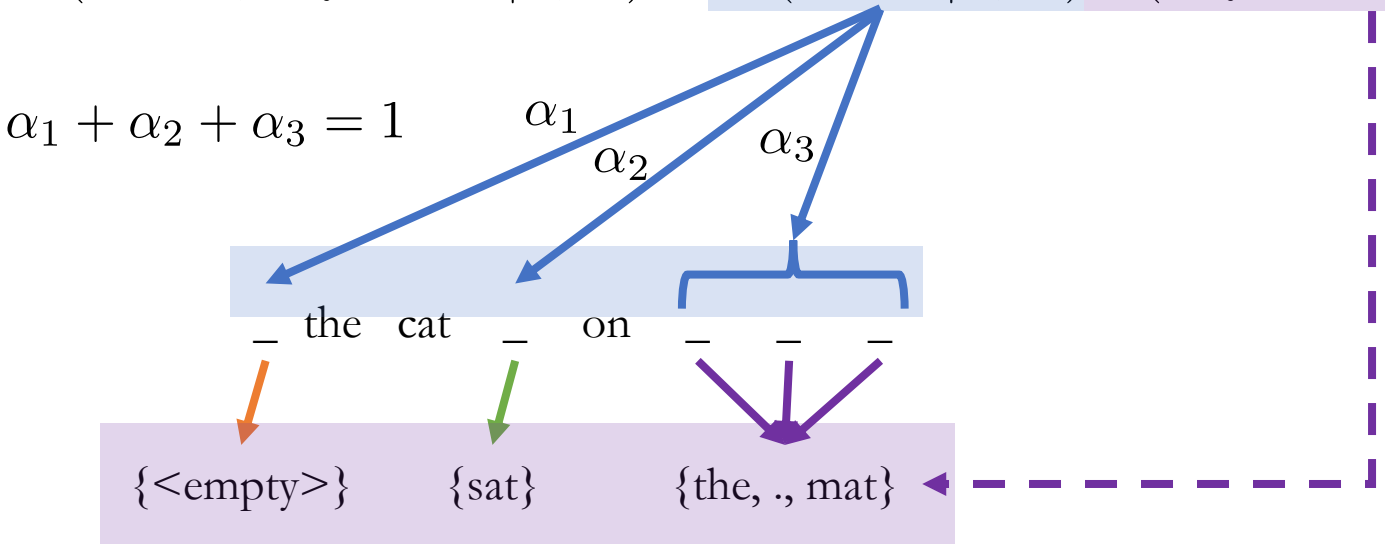


- **Target**: "how cool! i have never been outside of ohio but I would like to."
- **Input**: a bag of words
- **Generated**: "how cool! i would like to but i have never been outside of ohio."
- **Generated order**: ". cool how but ! never i i been like have outside would to of ohio"



More general oracles

- The binary tree oracle considers a subset of all possible generation orders: $O(2^T)$ vs. $O(T!)$
- Instead, consider a full set of generation orders
  - [Stern et al., 2019 ICML] [Gu et al., 2019 NeurIPS] [Emelianenko et al., 2019 NeurIPS]
- A reference policy: $\pi^*(<\text{loc}>, <\text{symbol}>|Y, X) = \pi^*(<\text{loc}>|Y, X)\pi^*(<\text{symbol}>|<\text{loc}>, Y, X)$



More general reference policies

- A balanced tree reference policy [Stern et al., 2019]
  - Prefer a location that is equidistant away from the near-by observed tokens

- An edit-distance reference policy [Gu et al., 2019 NeurIPS]
  - Use dynamic programming to find the next token that minimizes the edit distance to the target sequence

		M	a	n	a	h	a	t	o	n
	0	1	2	3	4	5	6	7	8	9
M	1	0	1	2	3	4	5	6	7	8
a	2	1	0	1	2	3	4	5	6	7
n	3	2	1	0	1	2	3	4	5	6
h	4	3	2	1	1	1	2	3	4	5
a	5	4	3	2	1	2	1	2	3	4
t	6	5	4	3	2	2	2	1	2	3
t	7	6	5	4	3	3	3	2	2	3
a	8	7	6	5	4	4	3	3	3	3
n	9	8	7	6	5	5	4	4	4	3

python-course.edu

The design of a reference policy matters

Input: But on the other side of the state, that is not the impression many people have of their former governor.

Output: Aber auf der anderen Seite des Staates ist das nicht der Eindruck, den viele von ihrem ehemaligen Gouverneur haben.

Parallel decode (binary tree loss):

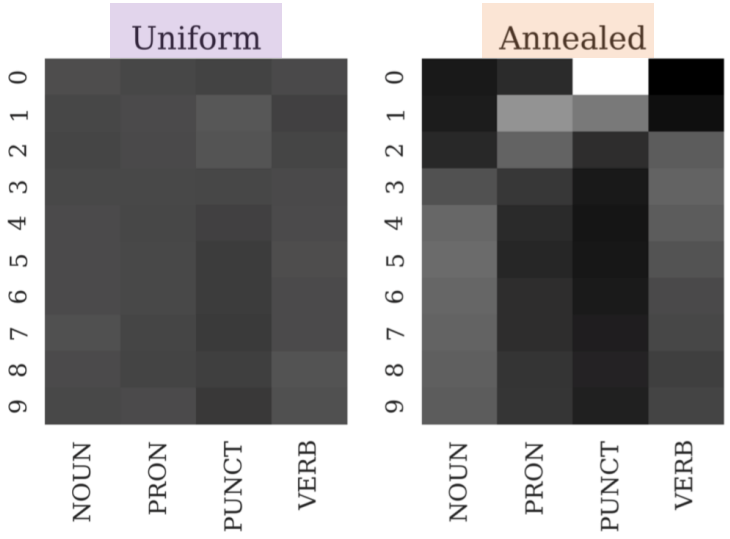
Aber_ auf_ der_ anderen_ Seite_ des_ Staates_ ist_ das_ nicht_ der_ Eindruck_ , _ den_ viele_ von_ ihrem_ ehemaligen_ Gouverneur _ haben_ ._

Aber_ auf_ der_ anderen_ Seite_ des_ Staates_ ist_ das_ nicht_ der_ Eindruck_ , _ den_ viele_ von_ ihrem_ ehemaligen_ Gouverneur _ haben_ ._

Aber_ auf_ der_ anderen_ Seite_ des_ Staates_ ist_ das_ nicht_ der_ Eindruck_ , _ den_ viele_ von_ ihrem_ ehemaligen_ Gouverneur _ haben_ ._

Aber_ auf_ der_ anderen_ Seite_ des_ Staates_ ist_ das_ nicht_ der_ Eindruck_ , _ den_ viele_ von_ ihrem_ ehemaligen_ Gouverneur _ haben_ ._

Aber_ auf_ der_ anderen_ Seite_ des_ Staates_ ist_ das_ nicht_ der_ Eindruck_ , _ den_ viele_ von_ ihrem_ ehemaligen_ Gouverneur _ haben_ ._



[Welleck et al., 2019]

Input: Everyone has the Internet, an iPad and eBooks.

Output: Jeder hat das Internet, ein iPad und eBooks.

Greedy decode (uniform loss):

Jeder_ hat_ das_ Internet_ , _ ein_ i Pad _ und_ eB oo ks_ ._

Jeder_ hat_ das_ Internet_ , _ ein_ i Pad _ und_ eB oo ks_ ._

Jeder_ hat_ das_ Internet_ , _ ein_ i Pad _ und_ eB oo ks_ ._

Jeder_ hat_ das_ Internet_ , _ ein_ i Pad _ und_ eB oo ks_ ._

Jeder_ hat_ das_ Internet_ , _ ein_ i Pad _ und_ eB oo ks_ ._

Jeder_ hat_ das_ Internet_ , _ ein_ i Pad _ und_ eB oo ks_ ._

Jeder_ hat_ das_ Internet_ , _ ein_ i Pad _ und_ eB oo ks_ ._

[Mitchell et al., 2019]

An alternative: a latent variable model

- $p(y_1, \ldots, y_L|X)$, where $y_l \in V$ and $L < \infty$
- From a finite set of data points $D = \{(Y^1, X^1), \ldots, (Y^N, X^N)\}$
- Three ways to go
  1. ~~Build a table containing all possible Y for each X to store probabilities~~
  2. Factorize the table: *what I just talked about today*

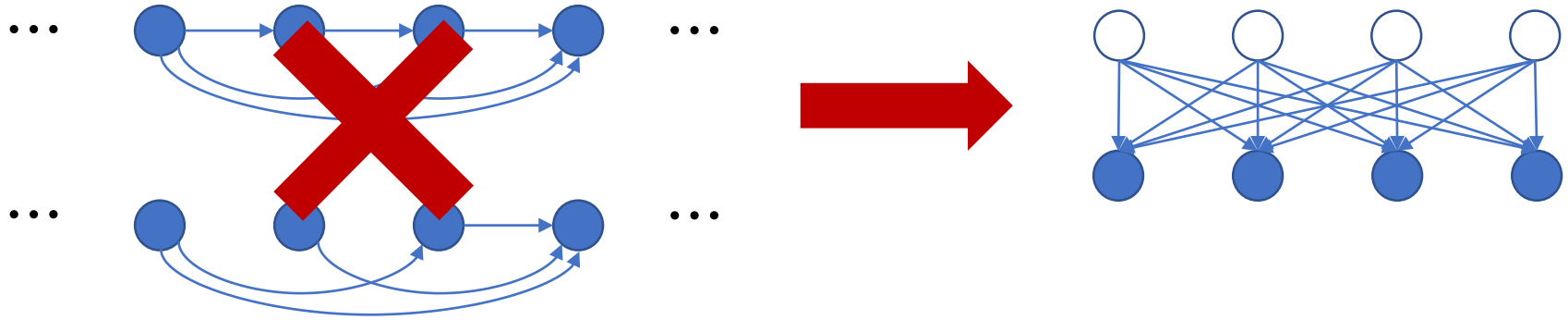
$$p(y_1, \ldots, y_L|X) = \sum_{(<\text{loc}>_1, \ldots, <\text{loc}>_L)} \prod_{l=1}^{L} \pi(<\text{loc}>_l, Y[<\text{loc}>_l]|Y[<\text{loc}>_1], \ldots, Y[<\text{loc}>_{l-1}], X)$$

3. Smooth the table: latent variable models

$$p(y_1, \ldots, y_L|X) = \int_{\mathcal{Z}} \prod_{l=1}^{L} \pi(y_l|X, Z)\mathrm{d}Z$$

An alternative: a latent variable model

- Combinatorial search → iterative optimization



- Search over a discrete space is difficult, but optimization in a continuous space is easier

Latent variable models

- Notoriously difficult to fit a latent variable model
  - Principal component analysis [Pearson, 1901], Boltzmann machines [Ackley et al., 1985], Helmholtz free energy [Hinton, 1994], wake-sleep algorithm [Hinton et al., 1995], independent component analysis [Comon, 1994; Bell&Sejnowski, 1995; Hyvarinen et al., 2000], contrastive divergence [Hinton, 2002], deep belief nets [Hinton et al., 2006], denoising autoencoders [Vincent et al., 2009], … and my entire PhD dissertation…
- Because, it's difficult to marginalize the latent variables

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \log \int_{z} p(x^n | z, C^n) p(z, C^n) \mathrm{d}z$$

Quick recap: variational autoencoders

- Learning by maximizing the lowerbound with an approximate posterior q

$$\log \int_{\mathcal{Z}} p(Y|Z,X)p(Z|C)\mathrm{d}Z \geq \mathbb{E}_{Z\sim q(Z|Y,X)}\left[\log p(Y|Z,X)\right] - \mathrm{KL}(q(Z|Y,X)\|p(Z|X))$$

  - Reparametrization trick to estimate the gradient w.r.t. q with a minimal variance
  - Proposed by Kingma & Welling [2013] and Rezende et al. [2014]
  - Reasonably standardized to train a VAE nowadays
- Generation by maximizing the lowerbound

$$\hat{Y} = \arg\max_{Y} \mathbb{E}_{Z\sim q(Z|Y,X)}\left[\log p(Y|Z,X)\right] - \mathrm{KL}(q(Z|Y,X)\|p(Z|X))$$

  - A difficult problem to solve deterministically
  - A standard practice is not good enough: $\arg\max_{Y} \log p(Y|\mathbb{E}_{p(Z|X)}[Z],X)$

Deterministic, iterative generation

- Delta posterior

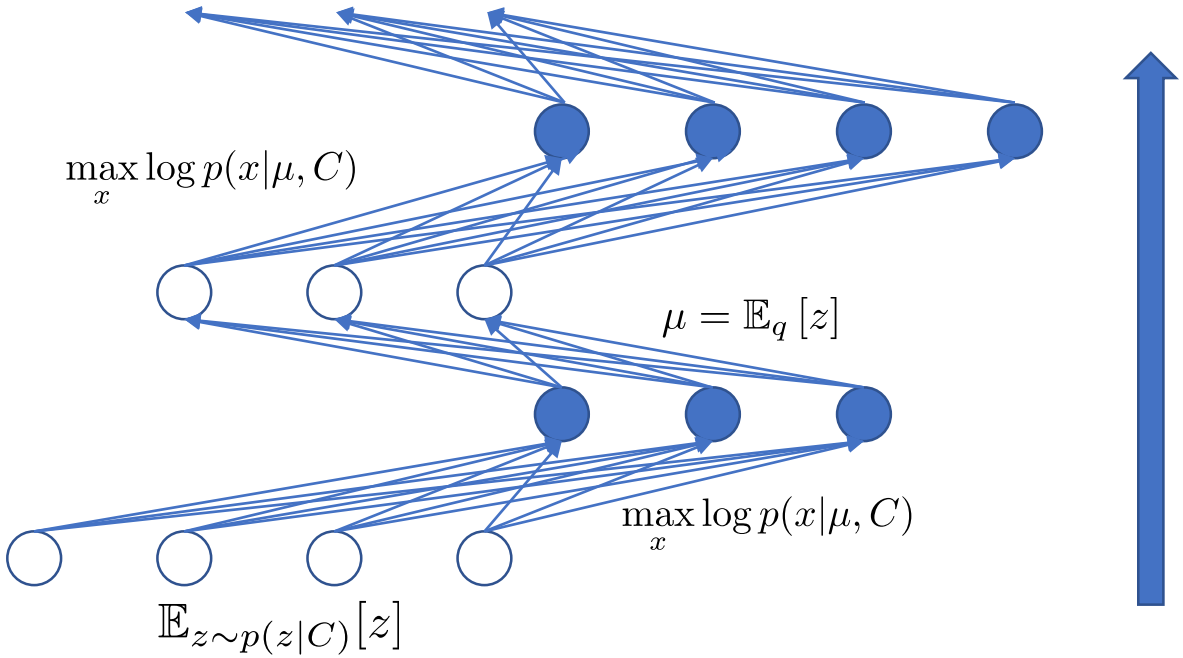
$$q'(z|x,C) = \begin{cases} 1, & \text{if } z = \mu \\ 0, & \text{otherwise} \end{cases}$$

- Alternating between two steps until convergence
  - Fit the delta posterior

$$\min_{\mu} \mathrm{KL}(q'\|q) \iff \mu = \mathbb{E}_q[z]$$

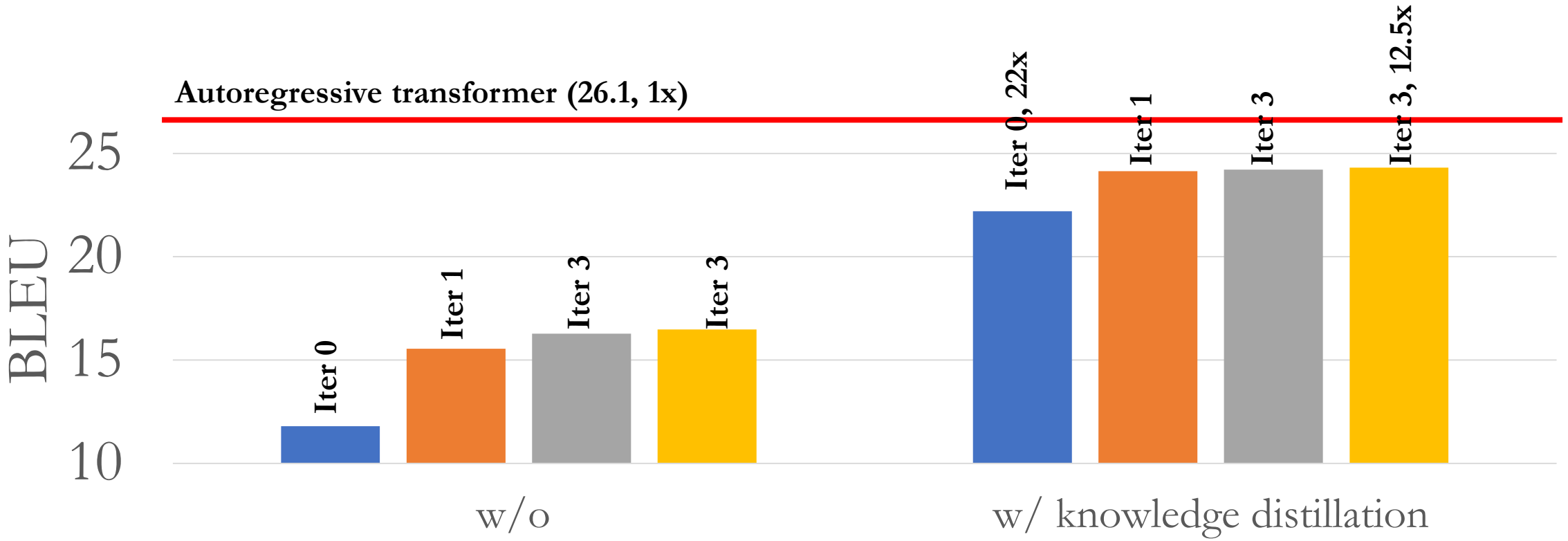
  - Maximize the lowerbound using q'

$$\max_{x} \mathbb{E}_{q'}[\log p(x|z,C)] = \max_{x} \log p(x|\mu,C)$$



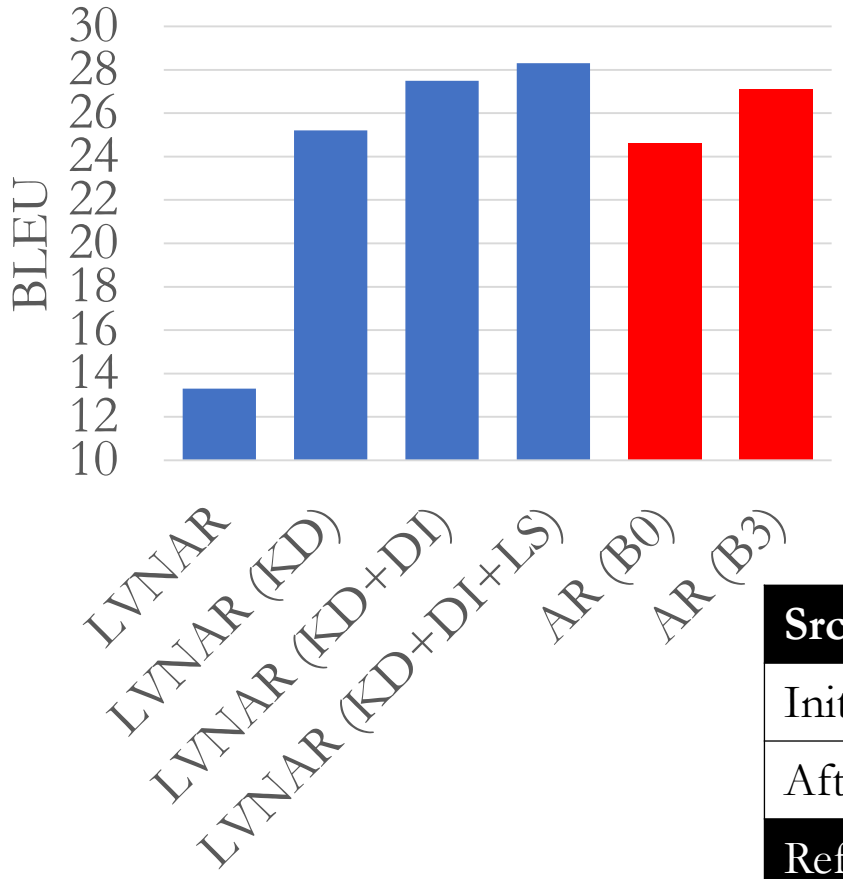
[Shu, Lee & Cho, 2020]

Deterministic, iterative generation



[Shu, Lee & Cho, 2020]

Deterministic, iterative generation



- Ja-En ASPEC [Nakazawa et al., 2016]
  - Limited to scientific domains
  - A narrower set of sentence styles

Src	標記 減 衰 量 標準 の 確立 を 試み た 。
Initial	an attempt was to establish the damping attenuation standard ...
After	an attempt **tries** to establish the damping **quantity** standard ...
Ref	an attempt was made establish establish damping attenuation standard ...

[Shu, Lee & Cho, 2020]

Wrap-up

- Structured prediction can be done in various ways
  - Fixed order, sequential generation ← often a default choice
  - Learned order, sequential generation ← Non-monotonic generation with imitation learning
  - Learned order, semi-sequential generation ← possible, but no time to discuss today
  - No order, parallel generation ← Latent-variable non-autoregressive generation
  - And, they all exhibit different properties

Conclusion

- Structured prediction is *not* reinforcement learning
  1. There is no notion of time in structured prediction, unlike in reinforcement learning
  2. There are often reference structures (labels), unlike in reinforcement learning
  3. It's not YOLO, unlike reinforcement learning

- If you are using (pure) reinforcement learning for your problem, try to see your problem as structured prediction and find an easier alternative solution:
  1. Can you run your policy over and over?
  2. Can your structure be generated in an arbitrary order?
  3. Is there a computational algorithm (or expert) that can (approximately) provide an optimal action?

Bonus – BERT Generation

Bonus: separation of location and symbol

- Consider the following decomposition

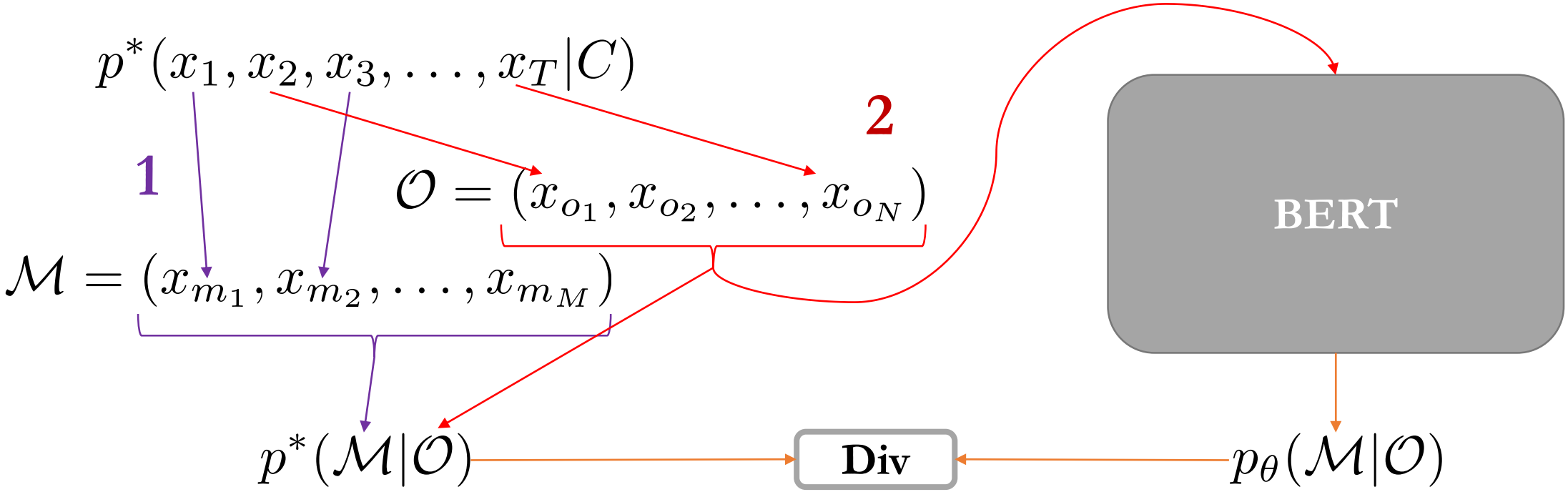
$$\pi(<\text{loc}>, <\text{symbol}>|Y, X) = \pi(<\text{loc}>|Y, X)\pi(<\text{symbol}>|<\text{loc}>, Y, X)$$

- Is it necessary to estimate $\pi(<\text{symbol}>|<\text{loc}>, Y, X)$ from scratch?
  - i.e., can we train $\pi(<\text{symbol}>|<\text{loc}>, Y, X)$ offline?
  - i.e., can we train $\pi(<\text{symbol}>|<\text{loc}>, Y, X)$ to work with any $\pi(<\text{loc}>|Y, X)$?

- BERT Generation [Mansimov et al., 2019; Ghazvininejad et al., 2019; Lawrence et al., 2019]
  - BERT [Devlin et al., 2019]
  - Cross-lingual BERT [Lample&Conneau, 2019]

What was BERT trained to do?

- **BERT learns a distribution** by learning as many conditionals as possible



What was BERT trained to do?

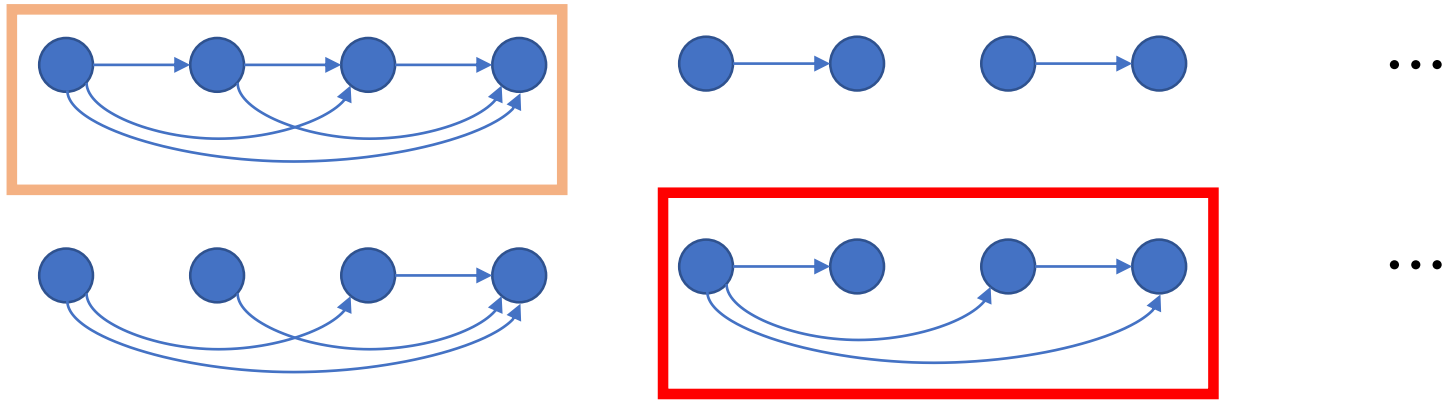
- **BERT** is trained to predict a symbol given a partially completed sequence
  - BERT assumes conditionally independent output, i.e.,
    $$p_{\text{BERT}}(y_i, y_j | y_{\neq i,\neq j}, X) = p_{\text{BERT}}(y_i | y_{\neq i,\neq j}, X) p_{\text{BERT}}(y_j | y_{\neq i,\neq j}, X) \; \forall i \neq j$$
  - BERT masks out multiple tokens, i.e., a few variables are missing:
    $$p_{\text{BERT}}(y_i | y_{k \in \mathcal{O}}, X), \text{ where } \mathcal{O} \subseteq \{1, \ldots, i-1, i+1, \ldots, T\}$$

- In other words, BERT was accidentally trained to be a perfect piece in a seq. generator
  $$\pi(\text{<loc>}, \text{<symbol>} | Y, X) = \pi(\text{<loc>} | Y, X) \pi(\text{<symbol>} | \text{<loc>}, Y, X)$$

- We just need to figure out the location policy

What does $\pi(<\text{loc}>|Y, X)$ do?

- Searches for a **directed**, **acyclic graph*** that covers all the variables

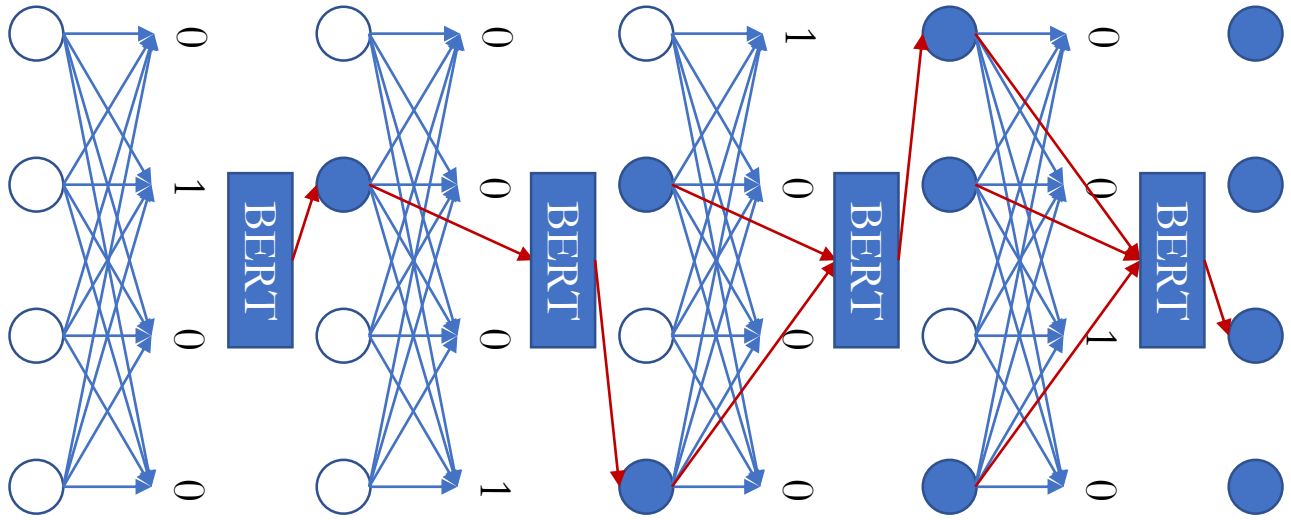


- The optimal **dependency structure** is conditioned on the input X
  - Optimality depends on our evaluation metric.

* it's not necessary to be acyclic in general as long as we constrain the number of visits per node in topological sweep.

BERT Generation

- Alternates between
  1. Determine the next set of variables to replace $\pi(<\text{loc}>|Y, X)$
  2. Sample each selected variable from BERT $\pi_{\text{BERT}}(<\text{symbol}>|<\text{loc}>, Y, X)$



[Manimov, Wang & Cho, 2019; Ghazvininejad et al., 2019; Lawrence, Kotnis & Niepert, 2019]

Confession of a deep learner

- A log-linear model with manually crafted features and coefficients

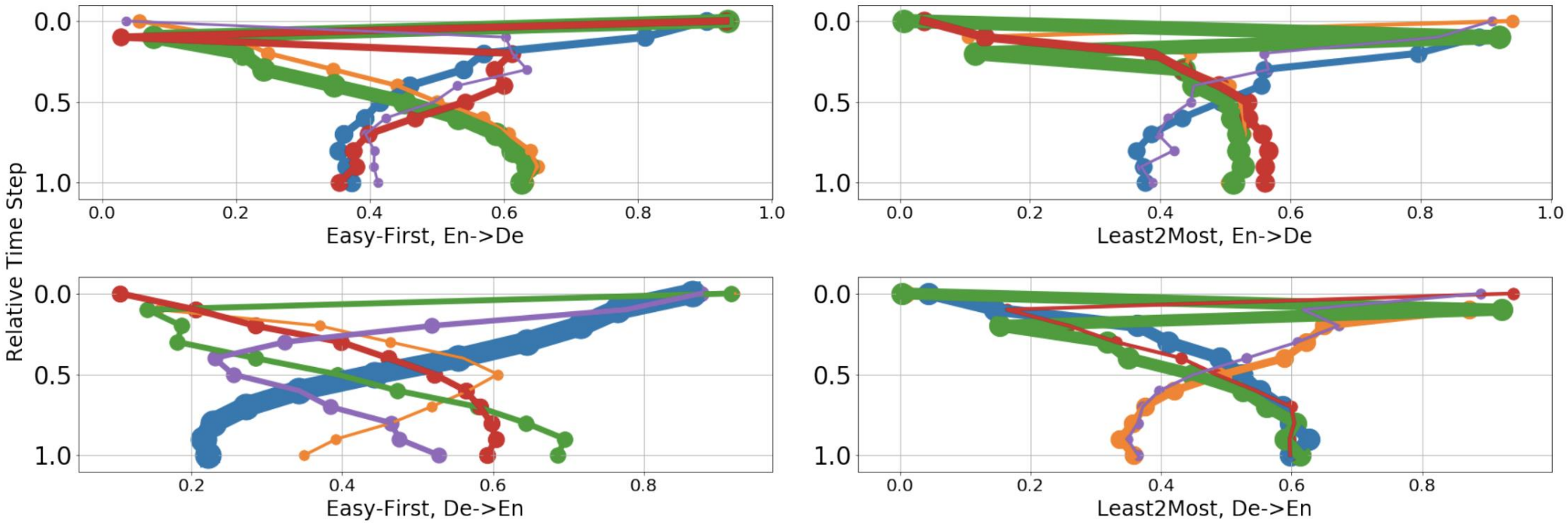
$$p(z_t^l = 1 | Z^{<l}, X^{<l}, C) \propto \exp \left\{ \sum_{k=1}^{K} \alpha_k \log \phi_k(t, l, \mathrm{BERT}(x^l | X^{<l})) \right\}$$

- Features
  1. Monotonicity: $\log \phi_{\mathrm{loc}}(t, l, \mathrm{BERT}(x^l | X^{<l})) = l - t$
  2. Predictability: $\log \phi_{\mathrm{pred}}(t, l, \mathrm{BERT}(x^l | X^{<l})) = \log p_{\mathrm{BERT}}(x_t^l = x_t^{l-1} | X^{<l})$
  3. Uncertainty: $\log \phi_{\mathrm{ent}}(t, l, \mathrm{BERT}(x^l | X^{<l})) = \log \mathcal{H}_{\mathrm{BERT}}(x_t^l | X^{<l})$
- Coefficients
  - Manually set them to induce desired properties
  - Uniform, left-to-right, easy-first, least-to-most, …

Mansimov, Wang & Cho [2019; rejected]

Beam search & adaptive generation order

- Generalizing beam search from monotonic autoregressive modeling
- Different search strategies lead to different order of generation



* though, for experiments, we only beam searched over the choice of words.

Source	**Doch ohne zivilgesellschaftliche Organisationen könne eine Demokratie nicht funktionieren .**
	_ _ _ _ _ _ _ _ _ _ _ _
	_ _ _ _ _ _ _ _ _ _ _ _ .
	_ _ _ _ _ _ _ _ **cannot** _ .
	_ _ _ _ _ _ _ _ **democracy** cannot _ .
	_ **without** _ _ _ _ _ democracy cannot _ .
	_ without _ _ _ _ _ democracy cannot **work** .
	But without _ _ _ _ _ democracy cannot work .
	But without _ **society** _ _ _ democracy cannot work .
	But without civil _ _ **,** _ democracy cannot work .
	But without civil **society** _ , _ democracy cannot work .
	But without civil society **organisations** , _ democracy cannot work .
	But without civil society organisations , **a** democracy cannot work .
Reference	**Yet without civil society organisations , a democracy cannot function .**

Reasonable, but not quite state-of-the-art

	b	T	Baseline Autoregressive	Decoding from an undirected sequence model Uniform	Left2Right	Least2Most	Easy-First
En→De	1	L	25.33	21.01	24.27	23.08	23.73
	4	L	26.84	22.16	25.15	23.81	24.13
	4	L^*	–	22.74	25.66	24.42	24.69
	1	$2L$	–	21.16	24.45	23.32	23.87
	4	$2L$	–	21.99	25.14	23.81	24.14
De→En	1	L	29.83	26.01	28.34	28.85	29.00
	4	L	30.92	27.07	29.52	29.03	29.41
	4	L^*	–	28.07	30.46	29.84	30.32
	1	$2L$	–	26.24	28.64	28.60	29.12
	4	$2L$	–	26.98	29.50	29.02	29.41

But, some interesting implications: Constant-time machine translation

- With enough parallel compute, each iteration is constant w.r.t. length

$$\hat{Z}^l = \arg\operatorname{top}_{Z^l} - \lceil L/T \rceil \sum_{t=1}^{T} \log p(\hat{x}_k | \hat{X}^{<l}, \hat{Z}^{<l}, C)$$

$$\hat{X}^l = \arg\max_{X^l} \log p(\hat{z}_t^l | \hat{Z}^{<l}, \hat{X}^{<l}, C)$$

L	K	Uniform	Left2Right	Least2Most	Easy-First	Hard-First
10	$T \to 1$	22.38	22.38	27.14	22.21	26.66
10	$T \to 1^*$	23.64	23.64	28.63	23.79	28.46
10	$\lceil T/L \rceil$	22.43	21.92	24.69	25.16	23.46
20	$T \to 1$	26.01	26.01	28.54	22.24	28.32
20	$T \to 1^*$	27.28	27.28	30.13	24.55	29.82
20	$\lceil T/L \rceil$	24.69	25.94	27.01	27.49	25.56

Initially put forward by Ghazvininejad et al. [2019]