Non-Autoregressive Decoding

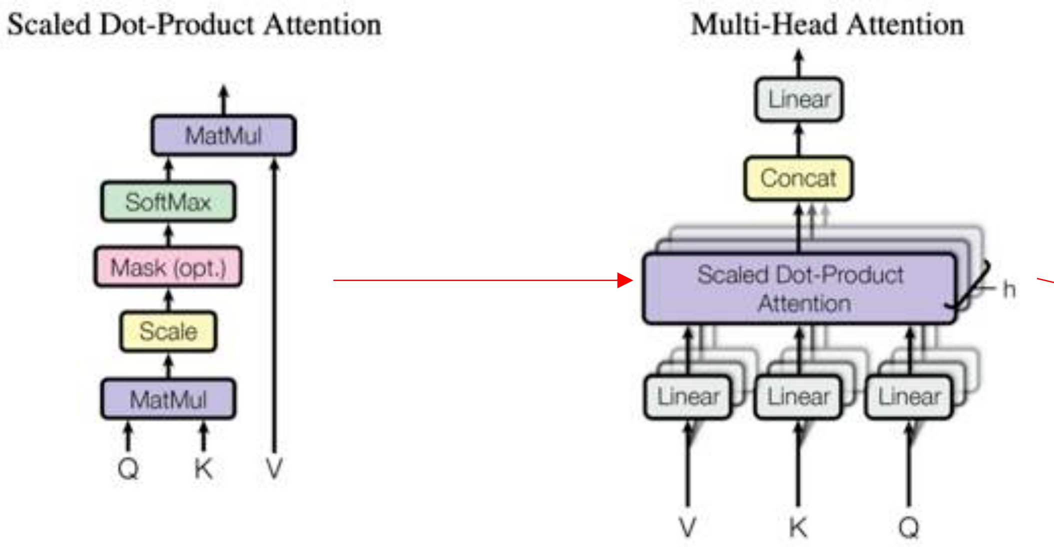
Xiachong Feng

Outline

- Transformer
- The Importance of Generation Order in Language Modeling *EMNLP18*
- Insertion Transformer:
  Flexible Sequence Generation via Insertion Operations *ICML19*
- Non-Monotonic Sequential Text Generation *ICML19*
- Insertion-based Decoding with automatically Inferred Generation Order
- Levenshtein Transformer
- Conclusion
- Paper List
- Reference

Transformer

Transformer



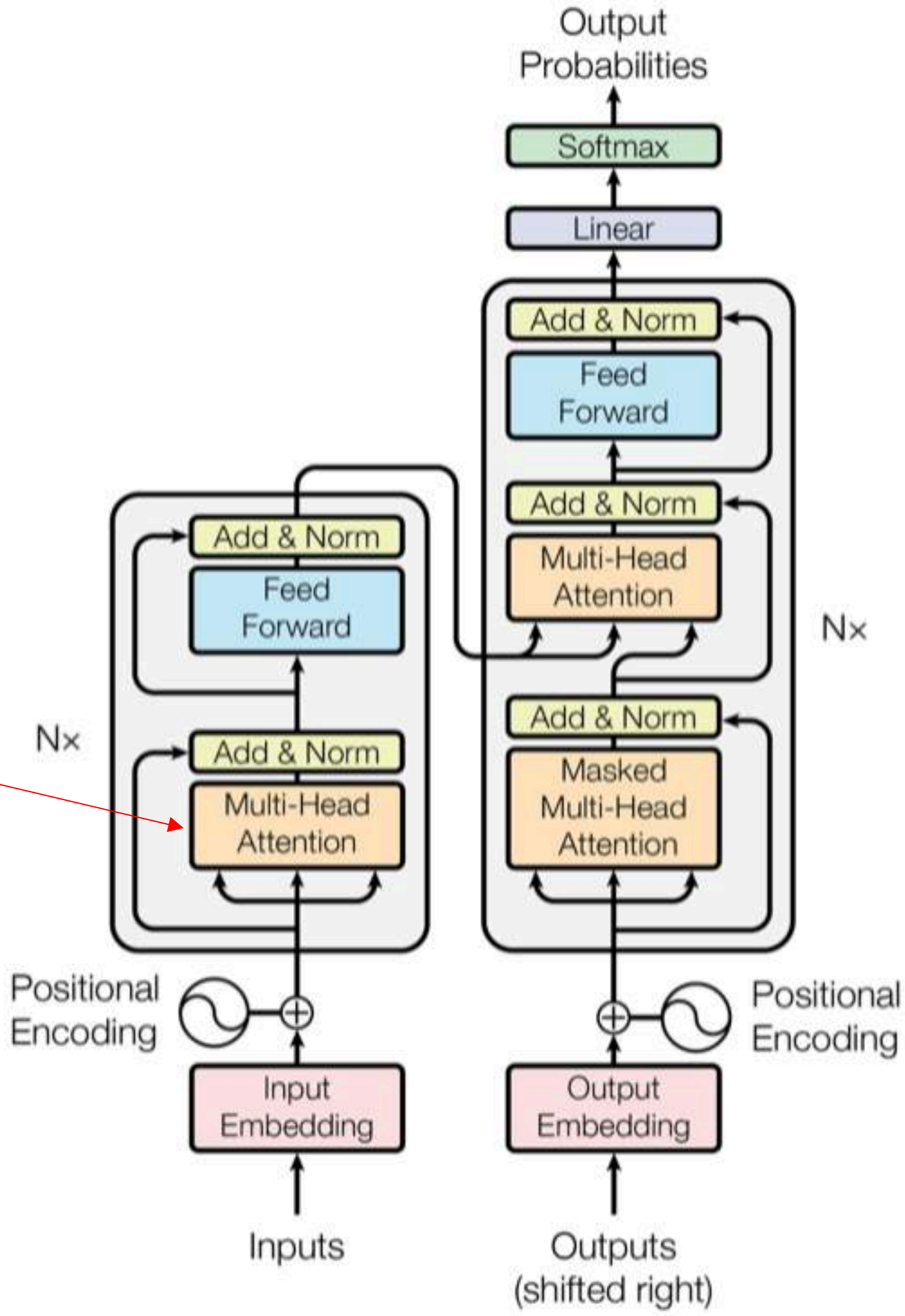
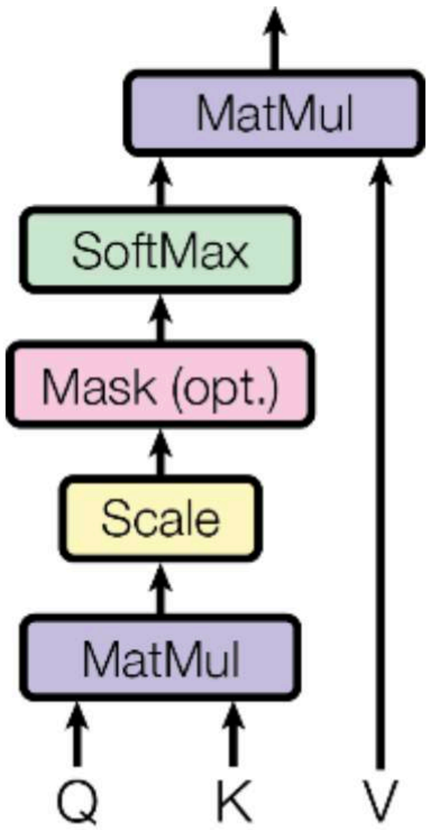


Figure 1: The Transformer - model architecture.

Scaled Dot-Product Attention

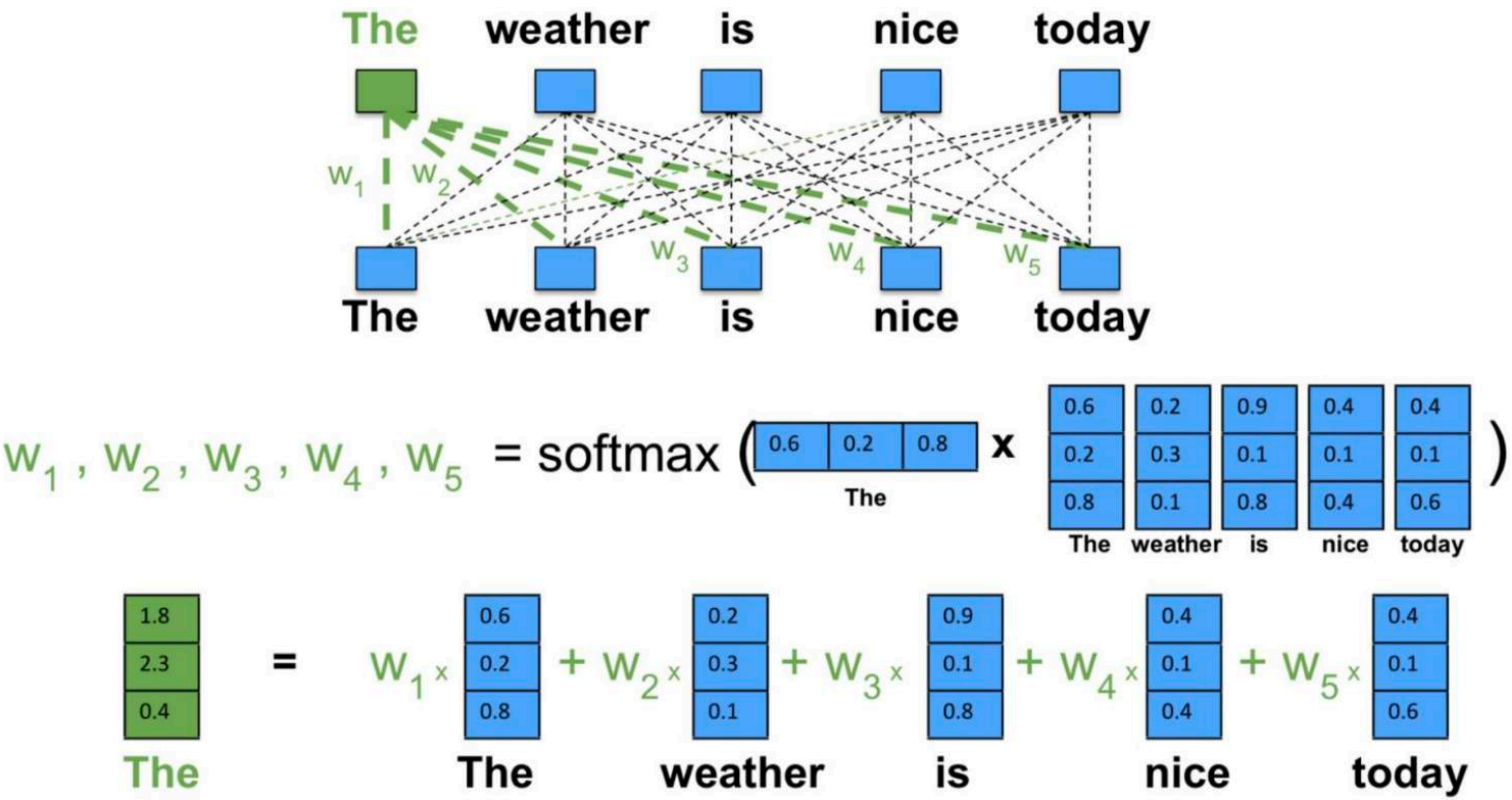


$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}}\right)V$$

For large values of d_k, the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients. To counteract this effect, the dot products are scaled by $\frac{1}{\sqrt{d_k}}$.

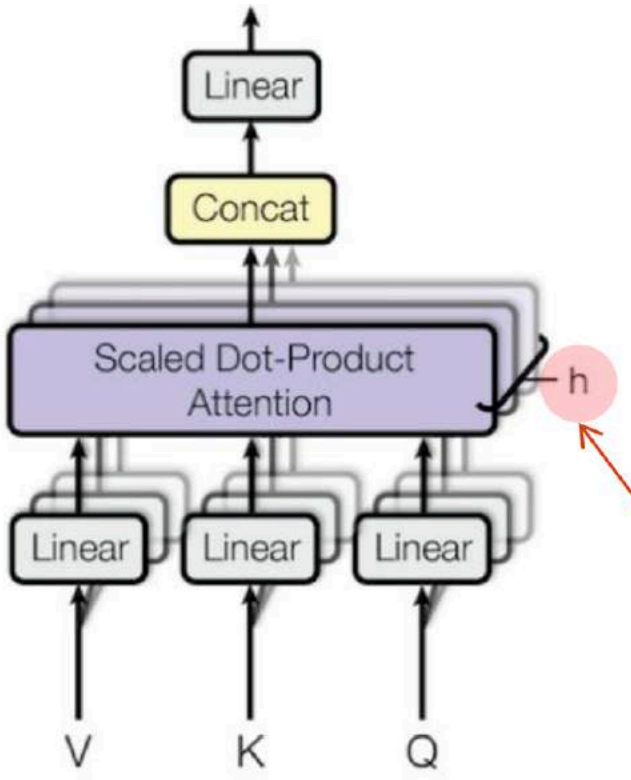
https://cips-upload.bj.bcebos.com/ssatt2019/CIPS_SSATT_2019_问答系统_唐都钰_段楠.pdf

Example



https://cips-upload.bj.bcebos.com/ssatt2019/CIPS_SSATT_2019_问答系统_唐都钰_段楠.pdf

Multi-Head Attention



Compute attentions from different representation spaces.

$$head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, \ldots, head_h)W^O$$

https://cips-upload.bj.bcebos.com/ssatt2019/CIPS_SSATT_2019_问答系统_唐都钰_段楠.pdf

Transformer

Layer Normalization adjusts parameters in each layer to a more stable distribution and accelerates the training of the network.

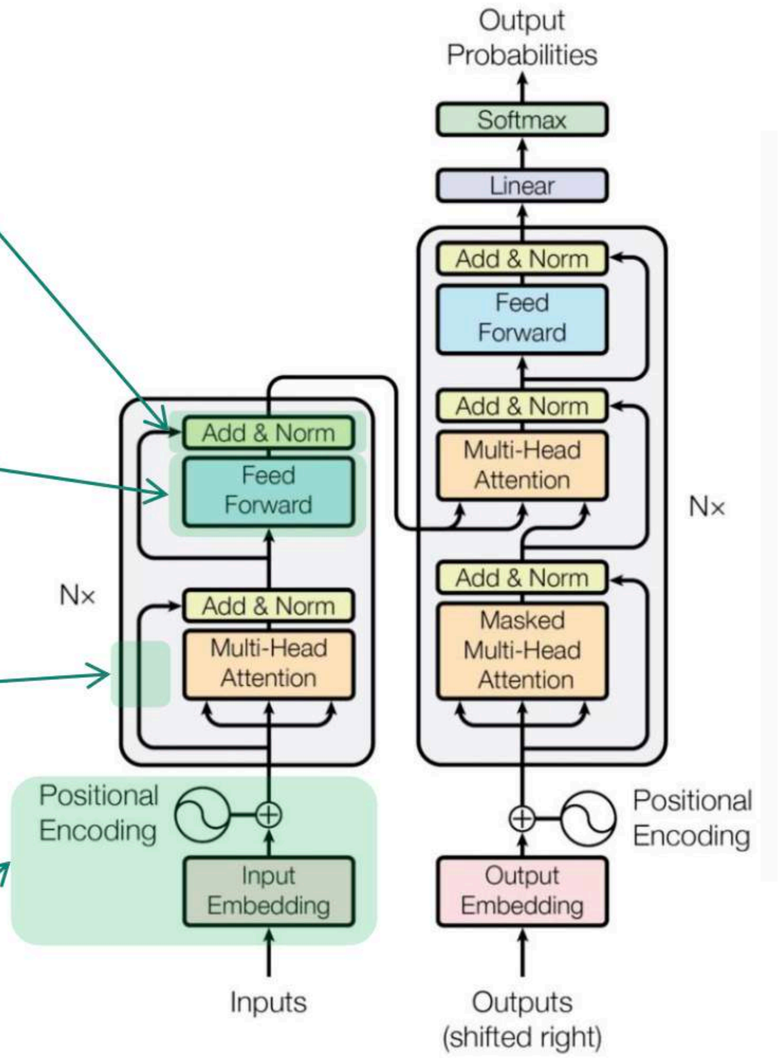
Position-wide Feed-Forward Networks is applied to each position:

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

While the linear transformations are the same across different positions, they use different parameters from layer to layer.

Residual connections allow to have deeper models.

Position encodings make use of the order of the sequence and they have the same dimension as input embeddings. The two can be summed to tell the neural net that there's an underlying sequential structure.



https://cips-upload.bj.bcebos.com/ssatt2019/CIPS_SSATT_2019_问答系统_唐都钰_段楠.pdf

The Importance of Generation Order in Language Modeling

Nicolas Ford, Daniel Duckworth, Mohammad Norouzi, George E. Dahl

Google Brain

EMNLP18

Overview

- Linguistic intuition might suggest that we should first generate some abstract representation of what we want to say and then serialize it.

- The best ordering we tried generates function words first and content words last, which cuts against the idea of committing to the general topic of a sentence first and only then deciding exactly how to phrase it.

Two-pass Language Models

- Produces partially-filled sentence “templates” and then fills in missing tokens
- Partitioning of the vocabulary into a set of *first-pass* and *second-pass* tokens to generate sentences.

sentence	**common first**
” all you need to do if you want the nation ’s press camped on your doorstep is to say you once had a [UNK] in 1947 , ” he noted memorably in his diary . [EOS]	” all you __ to __ if you __ the __ ’s __ __ on __ __ is to __ you __ had a [UNK] in __ , ” he __ __ in his __ . [EOS]

y

$y^{(1)}$ Template:first-pass tokens + a special placeholder token

$y^{(2)}$ Second-pass tokens

Two-pass Language Models

- Two copies of the Transformer model
  - **Neural language model $\boldsymbol{p_1}$** : The first copy just generates the template, so it has no encoder.
  - **Conditional translation model $\boldsymbol{p_2}$** : The second copy is a sequence-to-sequence model that translates the template into the complete sentence.

$$p(\mathbf{y}) \; = \; p_1(\mathbf{y}^{(1)}) \, p_2(\mathbf{y}^{(2)} \mid \mathbf{y}^{(1)}) \; .$$

Sentence→template   template→final

no encoder   Seq2Seq

Two-pass Language Models

template

sentence	common first	rare first	function first	content first	odd first
” all you need to do if you want the nation ’s press camped on your doorstep is to say you once had a [UNK] in 1947 , ” he noted memorably in his diary . [EOS]	” all you _ to _ if you _ the _ ’s _ _ on _ _ is to _ you _ had a [UNK] in _ , ” he _ _ in his _ . [EOS]	_ _ _ need _ do _ _ want _ nation _ press camped _ your doorstep _ _ say _ once _ _ _ _ 1947 _ _ _ noted memorably _ _ diary _ [EOS]	” all you _ to _ if you _ the _ ’s _ _ on your _ is to _ you _ _ a _ in _ , ” he _ _ in his _ . [EOS]	_ _ _ need _ do _ _ want _ nation _ press camped _ _ doorstep _ _ say _ once had _ [UNK] _ 1947 _ _ _ noted memorably _ _ diary _ [EOS]	” all you need _ _ _ you _ the nation ’s press camped on your doorstep _ _ say you once had _ _ _ _ _ ” _ noted _ _ his _ . [EOS]
the team announced thursday that the 6-foot-1 , [UNK] starter will remain in detroit through the 2013 season . [EOS]	the _ _ _ that the _ , [UNK] _ will _ in _ _ the _ _ . [EOS]	_ team announced thursday _ _ 6-foot-1 _ _ starter _ remain _ detroit through _ 2013 season _ [EOS]	the _ _ _ that the _ , _ _ will _ in _ through the _ _ . [EOS]	_ team announced thursday _ _ 6-foot-1 _ [UNK] starter _ remain _ detroit _ _ 2013 season _ [EOS]	the team announced _ _ the 6-foot-1 _ _ _ will remain _ _ through the 2013 _ . [EOS]
scotland ’s next game is a friendly against the czech republic at hampden on 3 march . [EOS]	_ ’s _ _ is a _ _ the _ _ at _ on _ _ . [EOS]	scotland _ next game _ _ friendly against _ czech republic _ hampden _ 3 march _ [EOS]	_ ’s _ _ is a _ against the _ _ at _ on _ _ . [EOS]	scotland _ next game _ _ friendly _ _ czech republic _ hampden _ 3 march _ [EOS]	_ ’s next game _ _ _ _ the czech republic at hampden on 3 march . [EOS]
of course , millions of additional homeowners did make a big mistake : they took advantage of ” liar loans ” and other [UNK] deals to buy homes they couldn ’t afford . [EOS]	of _ , _ of _ _ _ _ a _ _ : they _ _ of ” _ _ ” and _ [UNK] _ to _ _ they _ ’t _ . [EOS]	_ course _ millions _ additional homeowners did make _ big mistake _ _ took advantage _ _ liar loans _ _ other _ deals _ buy homes _ couldn _ afford _ [EOS]	of _ , _ of _ _ _ _ a _ _ : they _ _ of ” _ _ ” and _ _ _ to _ _ they _ _ _ . [EOS]	_ course _ millions _ additional homeowners did make _ big mistake _ _ took advantage _ _ liar loans _ _ other [UNK] deals _ buy homes _ couldn ’t afford _ [EOS]	of _ _ _ of additional _ _ _ _ big _ _ they _ advantage of ” liar _ ” and other _ deals _ buy homes they couldn _ afford . [EOS]

Table 1: Some example sentences from the dataset and their corresponding templates. The placeholder token is indicated by “_”.

Results

- It is easier to first decide something about its syntactic structure.
- It is preferable to delay committing to a rare token for as long as possible as all subsequent decisions will then be conditioning on a low-probability event.

Model	Train	Validation	Test
odd first	39.925	45.377	45.196
rare first	38.283	43.293	43.077
content first	38.321	42.564	42.394
common first	36.525	41.018	40.895
function first	36.126	40.246	40.085
baseline	38.668	41.888	41.721
enhanced baseline	35.945	39.845	39.726

Table 2: The perplexities achieved by the best version of each of our models.

Insertion Transformer: Flexible Sequence Generation via Insertion Operations

Mitchell Stern, William Chan, Jamie Kiros, Jakob Uszkoreit
Google Brain, University of California, Berkeley

ICML19

Insertion Transformer

- x : source canvas (sequence)
- y : target canvas (sequence)
- $\widehat{y_t}$: hypothesis canvas at time t
- $\mathcal{C}$: content vocabulary (token vocabulary for sequences)
- l : locations $\in [0, |\widehat{y_t}|]$

Insertion Transformer: Flexible Sequence Generation via Insertion Operations

Serial generation:

t	Canvas	Insertion
0	[]	(ate, 0)
1	[ate]	(together, 1)
2	[ate, together]	(friends, 0)
3	[friends, ate, together]	(three, 0)
4	[three, friends, ate, together]	(lunch, 3)
5	[three, friends, ate, lunch, together]	($\langle$EOS$\rangle$, 5)

Parallel generation:

t	Canvas	Insertions
0	[]	(ate, 0)
1	[ate]	(friends, 0), (together, 1)
2	[friends, ate, together]	(three, 0), (lunch, 2)
3	[three, friends, ate, lunch, together]	($\langle$EOS$\rangle$, 5)

$$p(c, l \mid x, \hat{y}_t) = \mathrm{InsertionTransformer}(x, \hat{y}_t).$$

Insertion Transformer Model

- **Full Decoder Self-Attention**
  - Remove causal self attention
- **Slot Representations via Concatenated Outputs**
  - Adding special marker tokens at the beginning and end of the decoder input to extend the sequence length by two.
  - Take the resulting $n + 2$ vectors in the final layer and concatenate each adjacent pair to obtain $n + 1$ slot representations.

Model

$$p(c, l \mid x, \hat{y}_t) = \mathrm{InsertionTransformer}(x, \hat{y}_t).$$

- **Joint content-location distribution**

$$p(c, l) = \mathrm{softmax}(\mathrm{flatten}(HW))$$

flatten this matrix into a vector

$H \in \mathbb{R}^{(T+1)\times h}$ matrix of slot representations

$W \in \mathbb{R}^{h\times|C|}$

- **Joint distribution using a conditional factorization**

$$p(c, l) = p(l)p(c|l) = \mathrm{softmax}(Hq) \times \mathrm{softmax}(h_l W)$$

learnable query vector

l-th row of H

Contextualized Vocabulary Bias

context vector $g = \text{maxpool}(H) \quad g \in \mathbb{R}^h$

shared bias $b = gV \quad V \in \mathbb{R}^{h \times |C|}$

$$B = \text{repmat}(b, [T+1, 1])$$

$$p(c, l) = \text{softmax}(HW + B)$$

Global bias

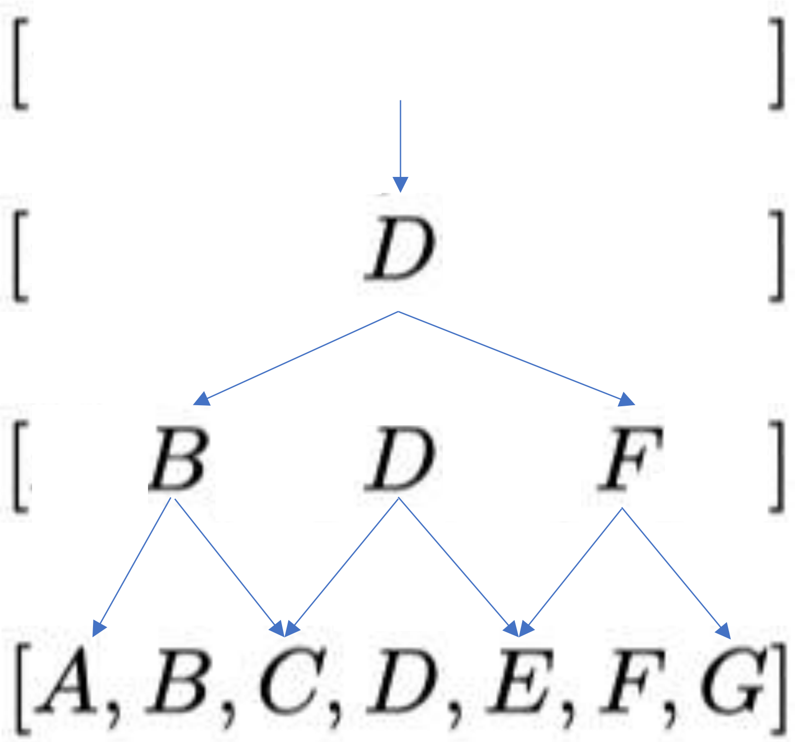
Training and Loss Functions

- **Left-to-Right**
  - Example : (x, y)
  - Sample a length $k \sim uniform([0, |y|])$
  - Create a new data point ((x, $\hat{y} = (y_1, \dots, y_k)$), y_{k+1})
  - Loss : classification loss (negative log-likelihood)
  - *Note : only concerns about* ***the last position*** *to insert*

$$\mathrm{loss}(x, \hat{y}) = -\log p(y_{k+1}, k \mid x, \hat{y})$$

Balanced Binary Tree

- Parallelism



Balanced Binary Tree

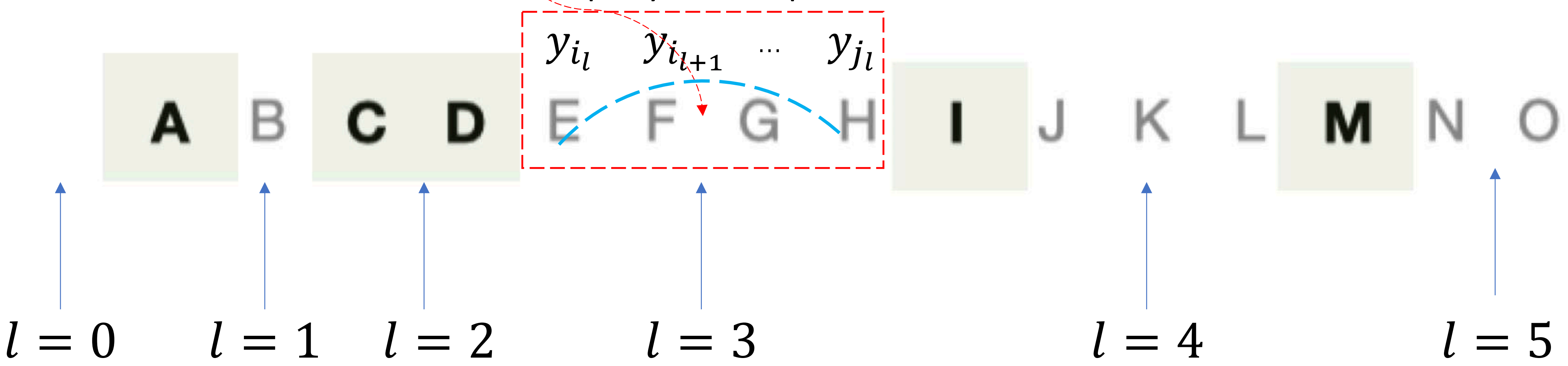
- Example : (x, y)
- Sample a length $k \sim uniform([0, |y|])$
- Sample a random subsequence of y of length k： $\hat{y}$
  1. Shuffle y
  2. Extract the first k
  3. Reorder

Soft binary tree loss

$$d_l(i) = \left| \frac{i_l + j_l}{2} - i \right| \qquad w_l(i) = \frac{\exp(-d_l(i)/\tau)}{\sum_{i'=i_l}^{j_l} \exp(-d_l(i')/\tau)}.$$

Distance ↓
$w_l(i)$ ↑

span of tokens from the target output yet to be produced



$$\text{slot-loss}(x, \hat{y}, l) = \sum_{i=i_l}^{j_l} -\log p(y_i, l \mid x, \hat{y}) \cdot \boxed{w_l(i).}$$

$$\text{loss}(x, \hat{y}) = \frac{1}{k+1} \sum_{l=0}^{k} \text{slot-loss}(x, \hat{y}, l).$$

Uniform

$$w_l(i) = \frac{\exp(-d_l(i)/\tau)}{\sum_{i'=i_l}^{j_l} \exp(-d_l(i')/\tau)}. \quad \tau \to \infty$$

$$\text{slot-loss}(x, \hat{y}, l) = \frac{1}{j_l - i_l + 1} \sum_{i=i_l}^{j_l} -\log p(y_i, l \mid x, \hat{y}).$$

Balanced binary tree and uniform losses

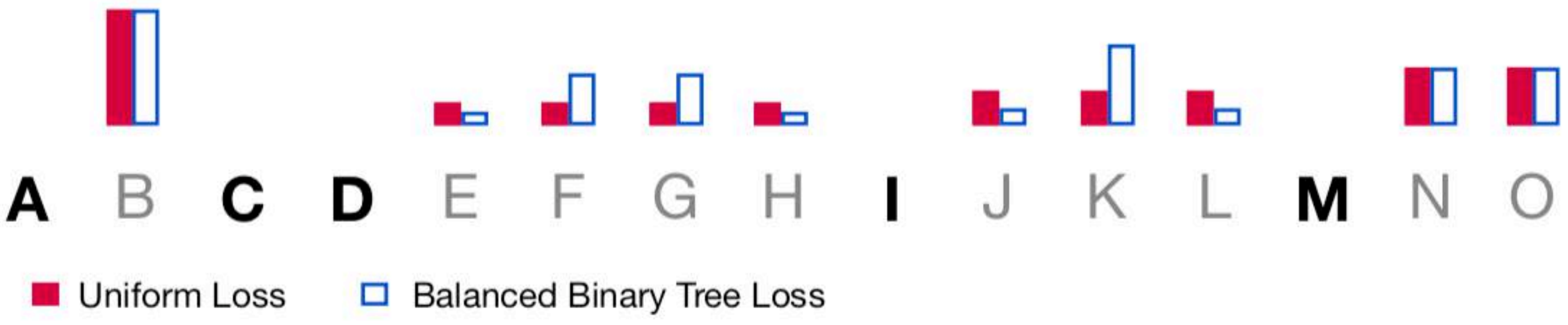


Figure 2. A visualization of the weighting of the per-token negative log-likelihoods in the balanced binary tree and uniform losses. The balanced binary tree loss strongly incentivizes the generation of the center word or center words within each slot.

Greedy Decoding

- Choose the action with the highest probability

$$(\hat{c}_t, \hat{l}_t) = \underset{c,l}{\operatorname{argmax}}\, p(c, l \mid x, \hat{y}_t).$$

- **sequence finalization**
  - until an <u>end-of-sequence</u> token gets selected
- **slot finalization**
  - restrict the argmax to locations whose <u>maximum-probability decision is not end-of-slot</u>
  - Until the model predicts an <u>end-of-slot</u> token for every location.

Parallel Decoding

- For each location l

$$p(c \mid l) \ : \ \hat{c}_{l,t} = \underset{c}{\operatorname{argmax}}\, p(c \mid l, x, \hat{y}_t).$$

$$p(c \mid l) = p(c,l)/p(l) = p(c,l)/\sum_{c'} p(c',l)$$

joint distribution

$$p(c,l) = p(l)p(c|l) = \operatorname{softmax}(Hq) \times \operatorname{softmax}(h_l W)$$

$$p(c,l) = \operatorname{softmax}(\operatorname{flatten}(HW)).$$

factorization

- slot finalization

Non-Monotonic Sequential Text Generation

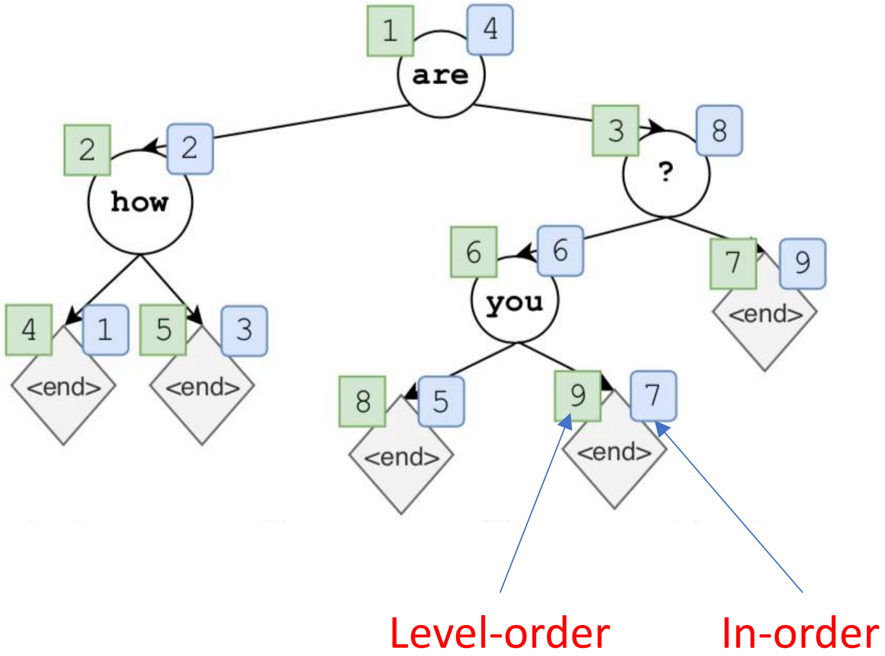
Sean Welleck , Kianté Brantley, Hal Daumé III, Kyunghyun Cho

New York University, University of Maryland, College Park
Microsoft Research, Facebook AI Research
CIFAR Azrieli Global Scholar

ICML19

Overview

- Recursively generating words to its left and then words to its right, yielding a binary tree.
- Learning is framed as imitation learning, including a coaching method which moves from imitating an oracle to reinforcing the policy’s own preferences



Imitation Learning

- Imitation Learning with Recurrent Neural Networks
- Learning to Search Better than Your Teacher *ICML15*
- https://zhuanlan.zhihu.com/p/25688750
- https://blog.csdn.net/WASEFADG/article/details/83651126
- https://www.quora.com/What-is-imitation-learning

Notation

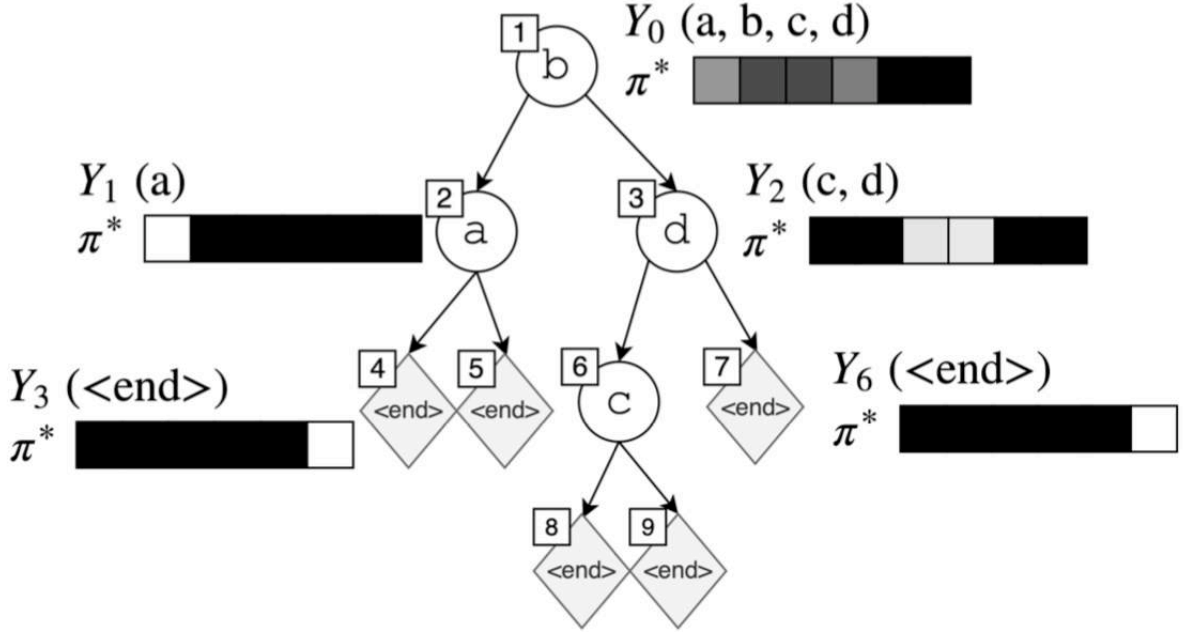
- Vocabulary $\tilde{V} = V \cup \{< end >\}$
- State space $\tilde{V}^*$
- State $s \in S$ corresponds to a sequence of tokens from $\tilde{V}$
- Init state: empty sequence $<>$
- End state: $< end >$
- Action a : select an element from vocab and append to the state
- $\tau(t)$: maps from in-order to level order
- Policy $\pi(a|s)$

Challenge

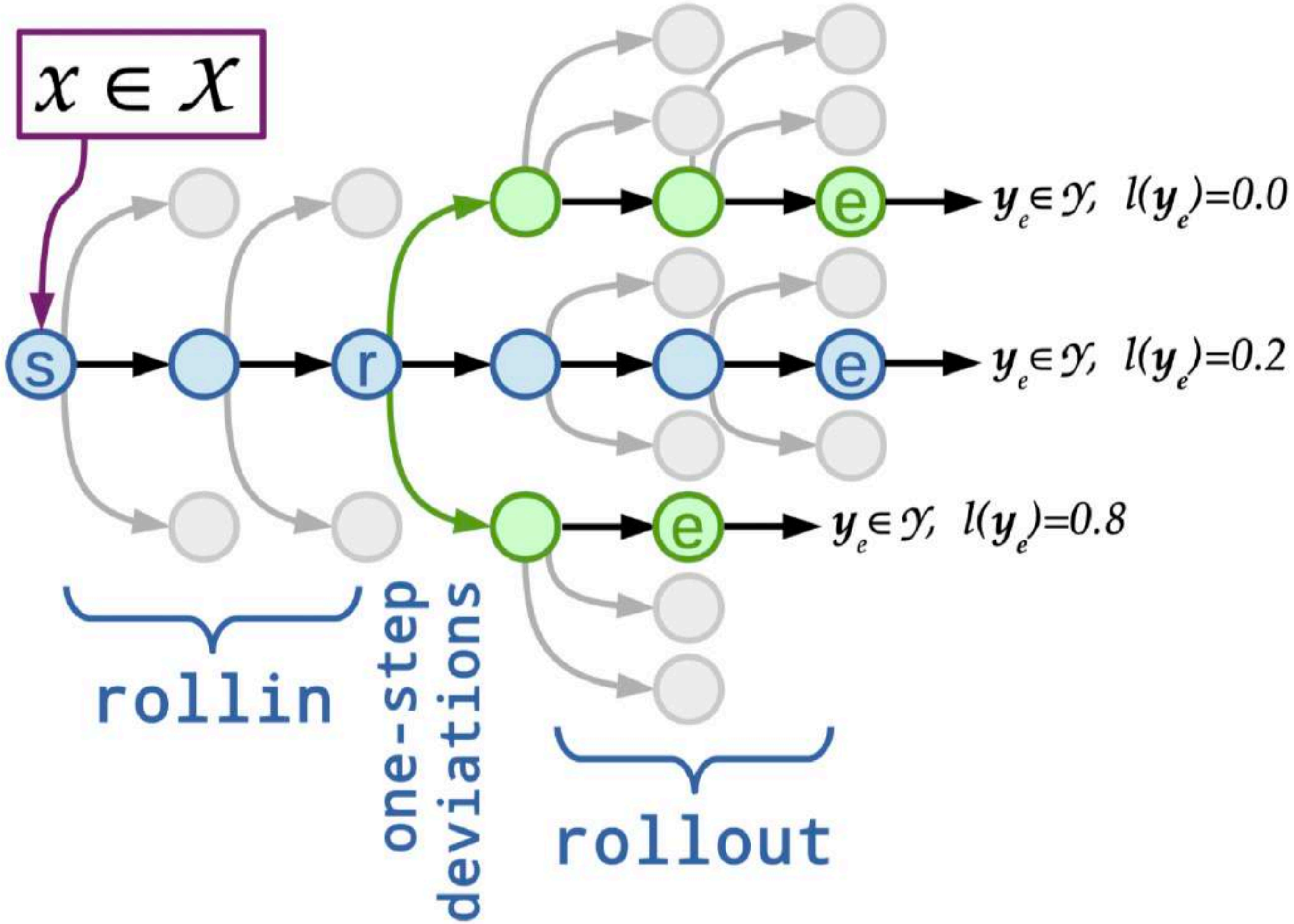
- The sequences Y alone only tell us what the final output sequences of words should be, but not what tree(s) should be used to get there.

Imitation Learning

- The first step, an oracle policy's action is to produce any word w that appears anywhere in Y.
- All words to the left of w in Y are generated recursively on the left (following the same procedure), and all words to the right of w in Y are generated recursively on the right.
- The oracle is non-deterministic (many "correct" actions are available at any given time), we inform this oracle policy with the current learned policy, encouraging it to favor actions that are preferred by the current policy.



Background: Learning to Search



Learning to Search Better than Your Teacher ICML15

Loss

- 3 $\mathbb{E}$
  - draw states s according to the state distribution induced by π^{in}
  - compute cost-to-go under π^{out}, for all possible actions a at that state.
- 2 $\mathbb{E}$
  - running π for t-many steps
- 1 $\mathbb{E}$
  - for one instance

$$\underbrace{\mathbb{E}_{Y\sim D}}_{1}\underbrace{\mathbb{E}_{t\sim U}}_{2}{}_{[2|Y|+1]}\underbrace{\mathbb{E}_{s_t\sim d^t_{\pi^{\text{in}}}}}_{3}\left[\mathcal{C}(\pi;\pi^{\text{out}},s_t)\right]$$

Cost Measurement

- when dealing with recurrent neural network policies using a cost function more analogous to a cross-entropy loss can be preferred
- use a KL-divergence type loss, measuring the difference between the action distribution produced by π and the action distribution preferred by π^{out}.
- first sampling one training sequence, running the roll-in policy for t steps, and computing the KL divergence at that state using π^* (reference or oracle)as π^{out}. Learning corresponds to minimizing this KL divergence iteratively with respect to the parameters of π.

$$\begin{aligned}\mathcal{C}(\pi;\pi^{\text{out}},s) &= D_{\text{KL}}\left(\pi^{\text{out}}(\cdot|s)\,||\,\pi(\cdot|s)\right)\\ &= \sum_{a\in\tilde{V}}\pi^{\text{out}}(a|s)\log\pi(a|s)+const.\end{aligned}$$

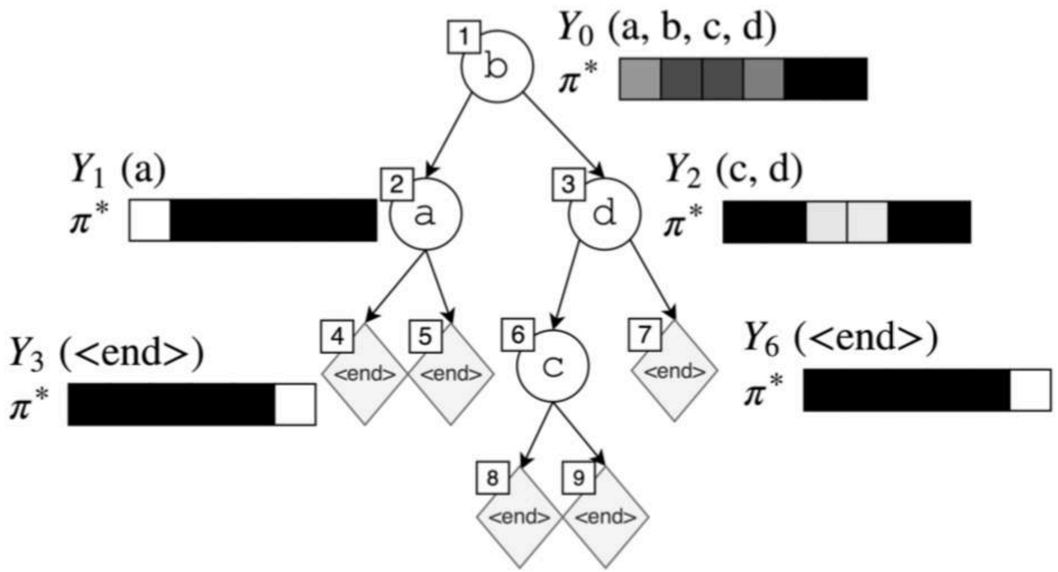
Roll-In Policies

- In most formal analyses, the roll-in policy is a stochastic mixture of the learned policy π and the oracle policy π^*
- *Experimentally*, it has often been found that simply **using the oracle's state distribution** is optimal

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

Learning to Search Better than Your Teacher ICML15

Oracle Policies



$$\pi^*(a|s_t) = \begin{cases} 1 & \text{if } a = \langle \text{end} \rangle \text{ and } Y_t = \langle \rangle \\ p_a & \text{if } a \in Y_t \\ 0 & \text{otherwise} \end{cases}$$

- Uniform Oracle. $p_a = 1/n$
- Coaching Oracle
    - preferring actions that are preferred by the current parameterized policy

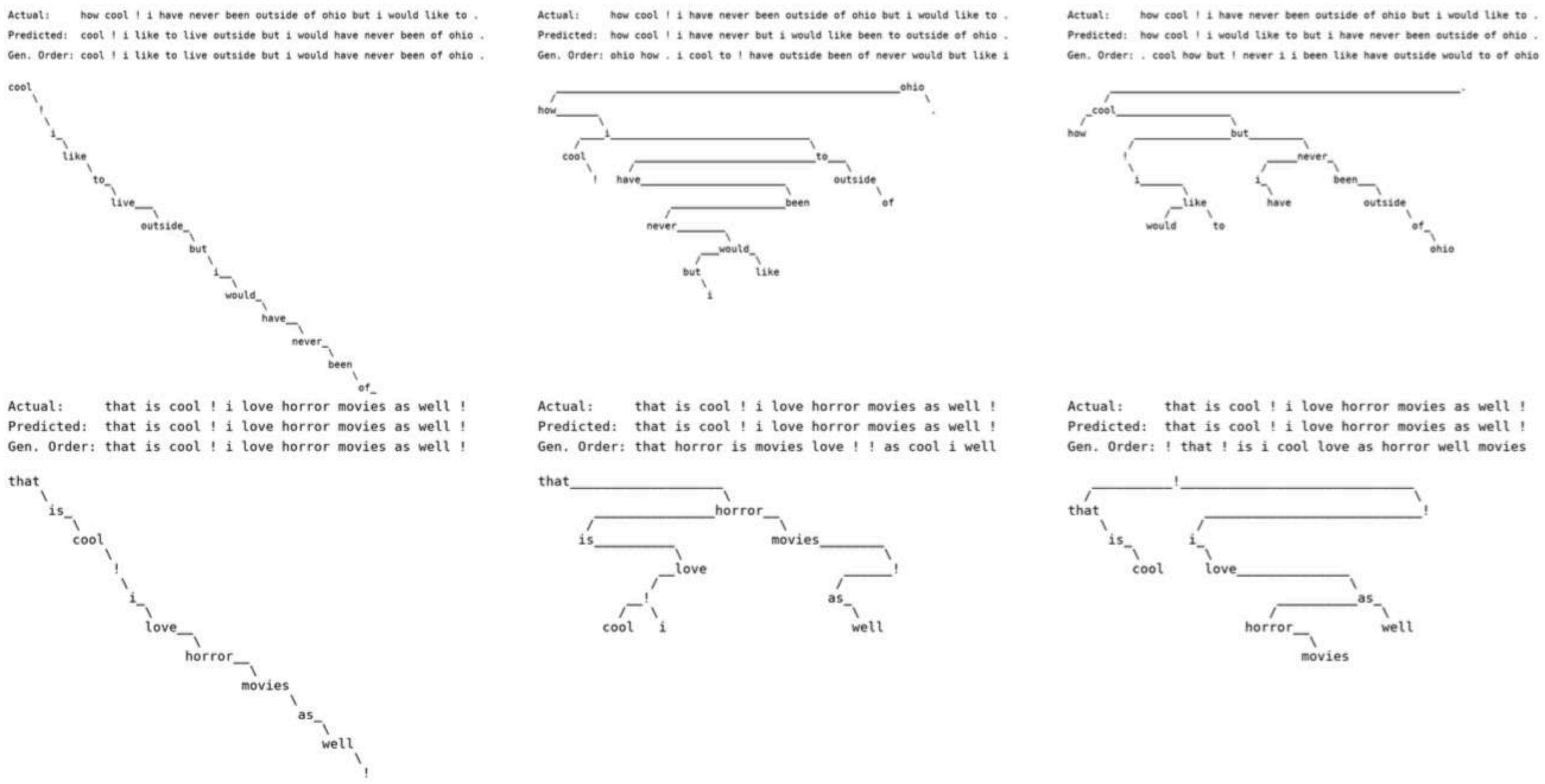
$$\pi^*_{\text{coaching}}(a|s) \propto \pi^*_{\text{uniform}}(a|s)\ \pi(a|s)$$

- Annealed Coaching Oracle(β from 1 to 0)

$$\pi^*_{\text{annealed}}(a|s) = \beta\pi^*_{\text{uniform}}(a|s) + (1-\beta)\pi^*_{\text{coaching}}(a|s)$$

Word Reordering Examples

Figure 8. Word Reordering Examples. The columns show policies trained with $\pi^*_{\text{left-right}}$, π^*_{uniform}, and π^*_{annealed}, respectively.



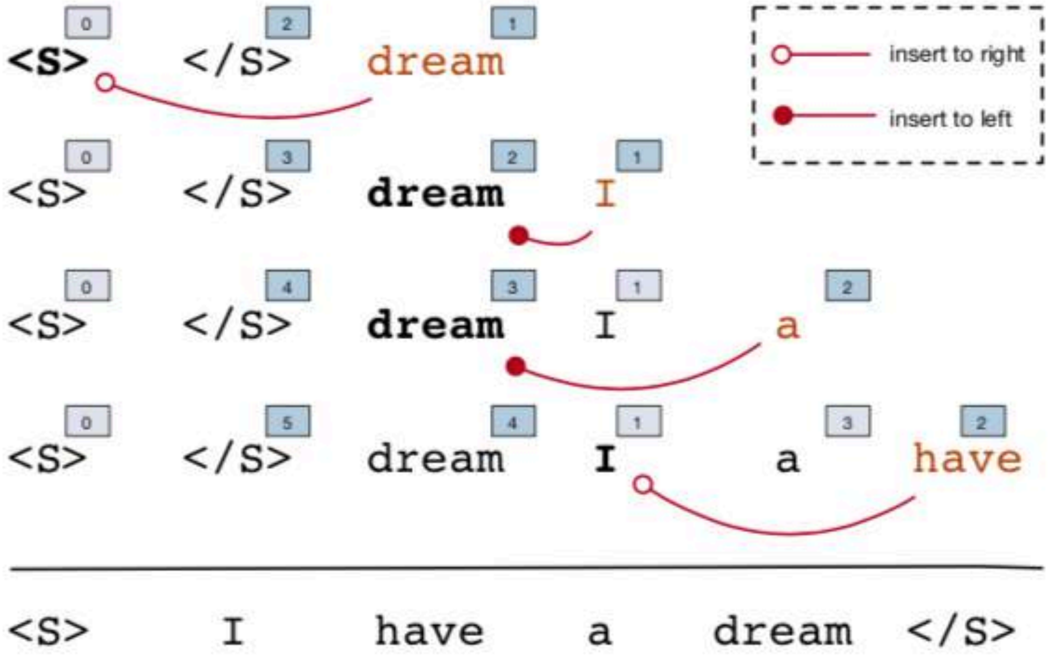
Insertion-based Decoding with automatically Inferred Generation Order

JiataoGu, QiLiu, KyunghyunCho

Facebook AI Research
New York University

Motivation

- L2R is not necessarily the optimal option for generating sequences.
- For instance, people sometimes tend to think of central phrases first before building up a whole sentence.



Orders as Latent Variables

- P_T is the set of all the permutations of $(1, \ldots, T\ )$
- $\pi = (z_2, z_3, \ldots z_T, z_{T+1}) \in P_T$
- $y_\pi = \{(y_2, z_2), \ldots, (y_{T+1}, z_{T+1})\}$, (y_T, z_T) represents the $t-th$ generated token and its absolute position
- Two special tokens
    - $(y_0, z_0) = (< s >, 0)$、$(y_1, z_1) = (</s>, T+1)$
- Object

$$p_\theta(\boldsymbol{y}|\boldsymbol{x}) = \sum_{\boldsymbol{\pi} \in \mathcal{P}_T} p_\theta(\boldsymbol{y_\pi}|\boldsymbol{x})$$

$$p_\theta(\boldsymbol{y_\pi}|\boldsymbol{x}) = p_\theta(\underline{y_{T+2}}|y_{0:T+1}, z_{0:T+1}, x_{1:T'}) \cdot \qquad y_{T+2} =< eod >$$

$$\prod_{t=1}^{T} p_\theta(\underline{y_{t+1}, z_{t+1}}|y_{0:t}, z_{0:t}, x_{1:T'})$$

Relative Representation of Positions

- r_i^t: the relative-position representations of token i at decode step t
- r_i^t is a vector
- Value : 0, 1, -1

$$\boldsymbol{r}_{i,j}^t = \begin{cases} -1 & z_j^t > z_i^t \ \text{(left)} \\ 0 & z_j^t = z_i^t \ \text{(middle)} \\ 1 & z_j^t < z_i^t \ \text{(right)} \end{cases}$$

- Matrix $R^t = [r_0^t, r_1^t, \dots, r_t^t]$ shows the relative-position representations of all the words in the sequence.
- Mapped back to the absolute position $z_i^t = \sum_{j=0}^{t} \max(0, \boldsymbol{r}_{i,j}^t)$
- Update

$$R^{t+1} = \left[\begin{array}{ccc|c} & & & \boldsymbol{r}_{t+1,0}^{t+1} \\ & R^t & & \vdots \\ & & & \boldsymbol{r}_{t+1,t}^{t+1} \\ \hline -\boldsymbol{r}_{t+1,0}^{t+1} & \cdots & -\boldsymbol{r}_{t+1,t}^{t+1} & 0 \end{array}\right]$$

Insertion-based Decoding

- Given$y_{0:t}$ and $r_{0:t}$
- Predict y_{t+1} and r_{t+1}
- Note : only concerns about the y_k which has been selected
- $s = -1$ if y_{t+1} is on the left of y_k, and $s = 1$ otherwise.

$$\boldsymbol{r}_{t+1,j} = \begin{cases} s & j = k \\ \boldsymbol{r}_{k,j} & j \neq k \end{cases}, \quad \forall j \in [0, t]$$

Insertion-based Decoding

Algorithm 1 Insertion-based Decoding

Initialize: $\boldsymbol{y} = (\langle \text{s} \rangle, \langle \text{/s} \rangle)$, $R = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$, $t = 1$

repeat

  Predict the next word y_{t+1} based on $\boldsymbol{y}, R$.

  if y_{t+1} is $\langle \text{eod} \rangle$ **then**

    break

  end if

  Choose an existing word $y_k \in \boldsymbol{y}$;

  Choose the left or right (s) of y_k to insert;

  Obtain the next position $\boldsymbol{r}_{t+1}$ with k, s (Eq. (6)).

  Update R by appending $\boldsymbol{r}_{t+1}$ (Eq. (5)).

  Update $\boldsymbol{y}$ by appending y_{t+1}

  Update $t = t + 1$

until Reach the maximum length

Map back to absolute positions $\boldsymbol{\pi}$ (Eq. (4))

Reorder $\boldsymbol{y}$: $y_{z_i} = y_i \quad \forall z_i \in \boldsymbol{\pi}, i \in [0, t]$

Transformer-InDIGO

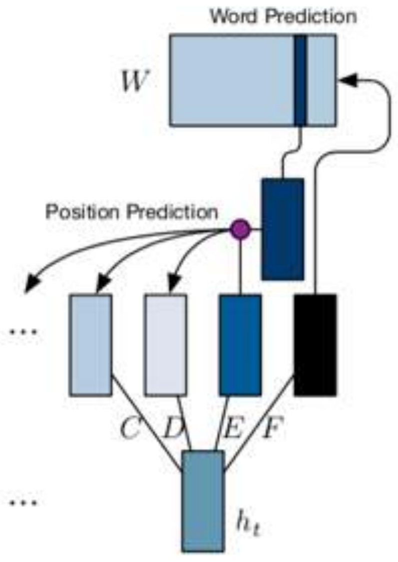
- Relative position-based self-attention

$$e_{i,j} = \frac{\left(\boldsymbol{u}_i^\top Q\right) \cdot \left(\boldsymbol{u}_j^\top K + \boxed{A_{[\boldsymbol{r}_{i,j}+1]}}\right)^\top}{\sqrt{d_{\text{model}}}}$$

$$A \in \mathbb{R}^{3 \times d_{\text{model}}}$$

Transformer-InDIGO

- Word & Position Prediction $H = (\boldsymbol{h}_0, ..., \boldsymbol{h}_t)$



(a)

$$p(y_{t+1}, \boldsymbol{r}_{t+1}|H) = p(y_{t+1}|H) \cdot p(\boldsymbol{r}_{t+1}|y_{t+1}, H)$$

$$p_{\text{word}}(y|H) = \text{softmax}\left((\boldsymbol{h}_t^\top F) \cdot W^\top\right)$$

$$p_{\text{pointer}}(k|y_{t+1}, H) = \text{softmax}\left((\boldsymbol{h}_t^\top E + W_{[y_{t+1}]}) \cdot \begin{bmatrix} H^\top C \\ H^\top D \end{bmatrix}^\top\right)$$

$$k_{t+1} \in [0, 2t+1]$$

Transformer-InDIGO

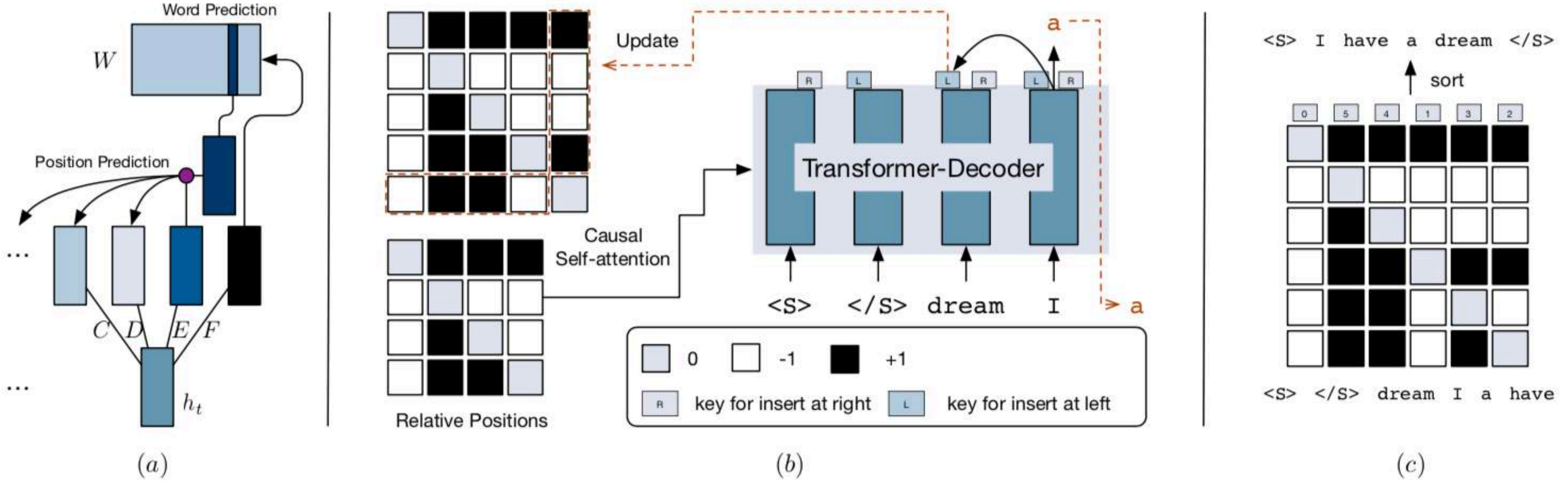


Figure 2: The overall framework of the proposed Transformer-InDIGO which includes (a) the word & position prediction module; (b) the one step decoding with position updating; (c) final decoding output by reordering.

Learning

- This is intractable since we need to enumerate all of the T! permutations of tokens. $p_\theta(\boldsymbol{y}|\boldsymbol{x}) = \sum_{\boldsymbol{\pi} \in \mathcal{P}_T} p_\theta(\boldsymbol{y}_{\boldsymbol{\pi}}|\boldsymbol{x})$
- Maximize the evidence lower-bound (ELBO) of the original objective by introducing an approximate posterior distribution of generation orders $q(\pi|x, y)$, which provides the probabilities of latent generation orders based on the ground-truth sequences x and y:

$$
\begin{aligned}
\mathcal{L}_{\text{ELBO}} &= \mathop{\mathbb{E}}_{\boldsymbol{\pi} \sim q} \log p_\theta(\boldsymbol{y}_{\boldsymbol{\pi}}|\boldsymbol{x}) + \mathcal{H}(q) \\
&= \mathop{\mathbb{E}}_{\boldsymbol{r}_{2:T+1} \sim q} \left(\sum_{t=1}^{T+1} \underbrace{\log p_\theta(y_{t+1}|y_{0:t}, \boldsymbol{r}_{0:t}, x_{1:T'})}_{\text{Word Prediction Loss}} \right. \\
&\quad \left. + \sum_{t=1}^{T} \underbrace{\log p_\theta(\boldsymbol{r}_{t+1}|y_{0:t+1}, \boldsymbol{r}_{0:t}, x_{1:T'})}_{\text{Position Prediction Loss}} \right) + \mathcal{H}(q)
\end{aligned}
$$

Searched Adaptive Order (SAO)

- beam-search in the space of all the permutations of the target sequence
- Sub-sequence： $y_{0:t}^{(b)} \in \mathcal{B}$
- Left words： $y' \in \mathbf{y} \backslash y_{0:t}^{(b)}$
- corresponding position $\boldsymbol{r}'$
- select *top-B* sub-sequences as the new set B for the next step.

$$\mathcal{L}_{SAO} = \frac{1}{B} \sum_{\pi \in \mathcal{B}} \log p_\theta(\boldsymbol{y}_{\boldsymbol{\pi}} | \boldsymbol{x})$$

$$q(\boldsymbol{\pi} | \boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1/B & \boldsymbol{\pi} \in \mathcal{B} \\ 0 & \text{otherwise} \end{cases}$$

Levenshtein Transformer

Jiatao Gu, Changhan Wang, and Jake Zhao (Junbo)

Facebook AI Research
New York University
Tigerobo Inc

Levenshtein Distance

4 = Levenshtein Distance(Saturday, Sundays)

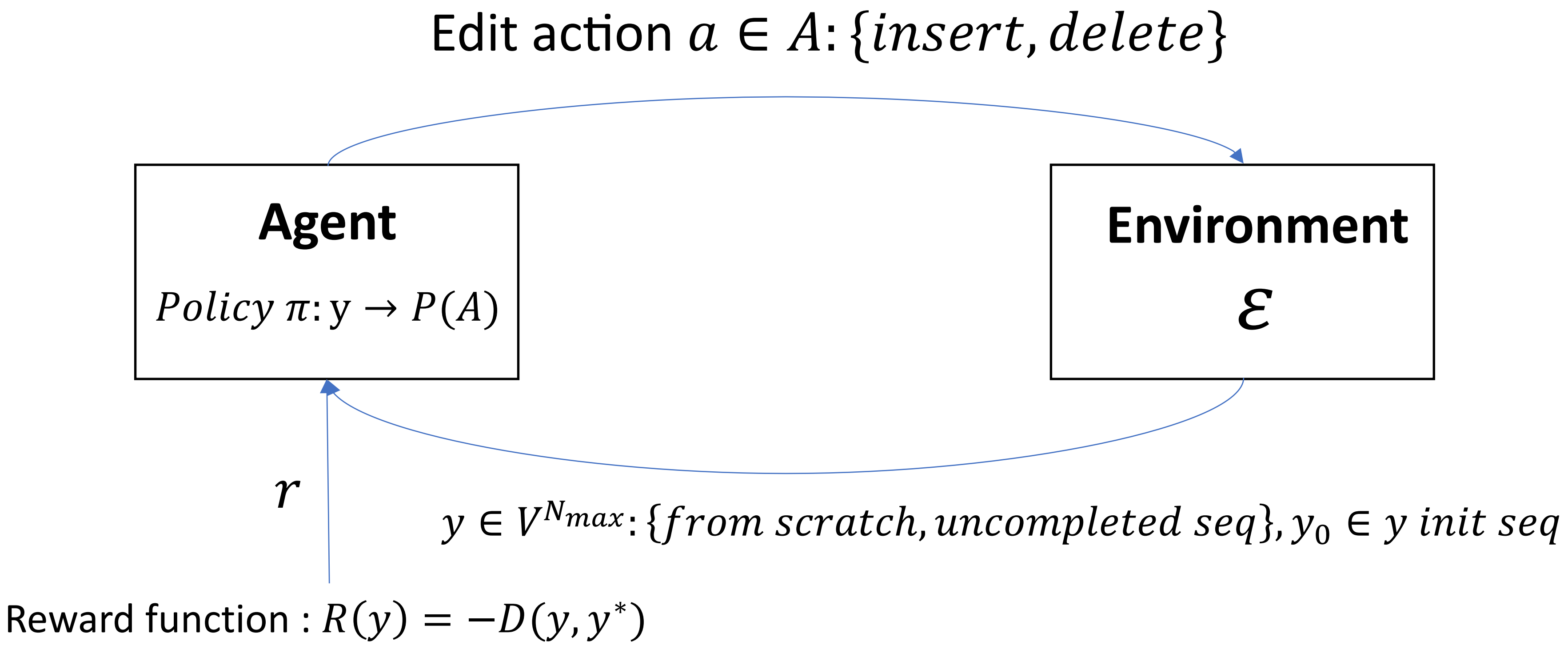
1. Saturday → Sturday // delete the first a
2. Sturday → Surday // delete the first t
3. Surday → Sunday // replace r with n
4. Sunday → Sundays // add s at the end

Overview

- Humans can revise, replace, revoke or delete any part of their generated text.
- Atomic operations : insertion and deletion
- Not only generation but also sequence refinement allowing dynamic length changes.
- Partially autoregressive model

Problem Formulation

- Markov Decision Process (MDP) $(\mathcal{Y}, \mathcal{A}, \mathcal{E}, \mathcal{R}, \boldsymbol{y_0})$



Actions

$$\boldsymbol{y}^{k+1} = \mathcal{E}(\boldsymbol{y}^k, \boldsymbol{a}^{k+1})$$

$$\boldsymbol{y}^k = (y_1, y_2, \ldots, y_n)$$

$a \in A: \{insert, delete\}$

<s> </s>

Deletion

- $\pi^{\text{del}}(d|i, \boldsymbol{y})$ makes a binary decision which is 1 (delete this token) or 0 (keep it)
- Avoid sequence boundary being broken $\pi^{\text{del}}(0|1, \boldsymbol{y}) = \pi^{\text{del}}(0|n, \boldsymbol{y})$

Insertion

- placeholder prediction and token prediction
- All locations (y_i, y_{i+1}) in $\boldsymbol{y}$
- $\pi^{\text{plh}}(p|i, \boldsymbol{y})$ the possibility of adding one or several placeholders
- $\pi^{\text{tok}}(t|i, \boldsymbol{y})$ for every placeholder predicted as above, replaces the placeholders with actual tokens in the vocabulary

Policy combination

- **delete tokens – insert placeholders - replace placeholders** with new tokens
- parallelize the computation within each sub-tasks.

$$\boldsymbol{a} = \{\underbrace{d_0, \ldots d_n}_{\boldsymbol{d}}; \underbrace{p_0, \ldots, p_{n-1}}_{\boldsymbol{p}}; \underbrace{t_0^1, \ldots t_0^{p_0}, \ldots, t_{n-1}^{p_{n-1}}}_{\boldsymbol{t}}\}$$

$$\pi(\boldsymbol{a}|\boldsymbol{y}) = \prod_{d_i \in \boldsymbol{d}} \pi^{\text{del}}(d_i|i, \boldsymbol{y}) \cdot \prod_{p_i \in \boldsymbol{p}} \pi^{\text{plh}}(p_i|i, \boldsymbol{y}') \cdot \prod_{t_i \in \boldsymbol{t}} \pi^{\text{tok}}(t_i|i, \boldsymbol{y}'')$$

Levenshtein Transformer

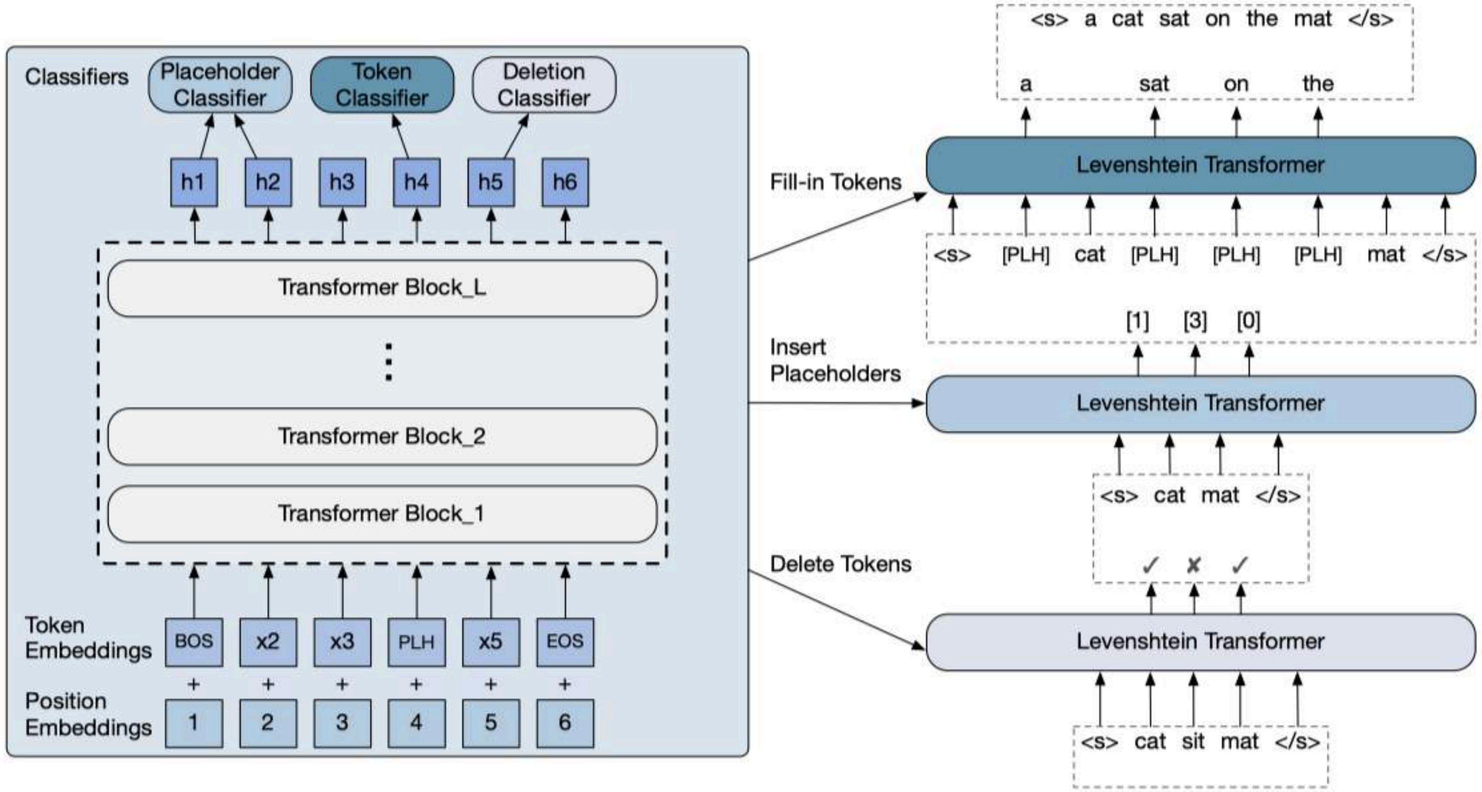


Figure 1: The overall framework of the decoder of the proposed Levenshtein Transformer. We show how the same architecture can be applied for three different tasks with specific classifiers. For simplicity, the attention between the encoder outputs is omitted within each Transformer-Block.

Levenshtein Transformer

- Decoder output : $(h_0, h_1, \ldots h_n)$, passed to three policy classifiers

1. **Deletion Classifier**: scans over the input tokens (except for the boundaries) and predict "deleted" (0) or "kept" (1) for each token position

$$\pi_\theta^{\text{del}}(d|i, \boldsymbol{y}) = \text{softmax}\left(\boldsymbol{h}_i \cdot A^\top\right), \quad i = 1, \ldots n-1$$

2. **Placeholder Classifier:** predicts the number of tokens to be inserted at every consecutive position pairs

$$\pi_\theta^{\text{plh}}(p|i, \boldsymbol{y}) = \text{softmax}\left(\texttt{concat}(\boldsymbol{h}_i, \boldsymbol{h}_{i+1}) \cdot B^\top\right), \quad i = 0, \ldots n-1 \quad B \in \mathbb{R}^{(K_{\max}+1)\times(2d_{\text{model}})}$$

3. **Token Classifier:** fill in tokens replacing all the placeholders.

$$\pi_\theta^{\text{tok}}(t|i, \boldsymbol{y}) = \text{softmax}\left(\boldsymbol{h}_i \cdot C^\top\right), \quad \forall y_i = \text{<PLH>}$$

Dual-policy Learning

Roll-in policy

Roll-in policy

$$\underbrace{\mathbb{E}_{\substack{\boldsymbol{y}_{\text{del}} \sim d_{\tilde{\pi}_{\text{del}}} \\ \boldsymbol{d}^* \sim \pi^*}} \sum_{d_i^* \in \boldsymbol{d}^*} \log \pi_\theta^{\text{del}}(d_i^* | i, \boldsymbol{y}_{\text{del}})}_{\textit{Deletion Objective}} + \underbrace{\mathbb{E}_{\substack{\boldsymbol{y}_{\text{ins}} \sim d_{\tilde{\pi}_{\text{ins}}} \\ \boldsymbol{p}^*, \boldsymbol{t}^* \sim \pi^*}} \left[\sum_{p_i^* \in \boldsymbol{p}^*} \log \pi_\theta^{\text{plh}}(p_i^* | i, \boldsymbol{y}_{\text{ins}}) + \sum_{t_i^* \in \boldsymbol{t}^*} \log \pi_\theta^{\text{tok}}(t_i^* | i, \boldsymbol{y}'_{\text{ins}}) \right]}_{\textit{Insertion Objective}}$$

Expert policy

Expert policy

suggested actions

Roll-in Policy

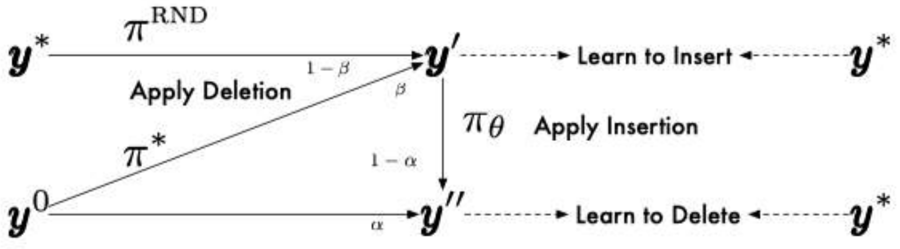


Figure 2: The data-flow of learning.

- **Learning to Delete**

mixture factor

any sequence ready to insert tokens

$$d_{\tilde{\pi}_{\mathrm{del}}} = \{\boldsymbol{y}^0 \quad \text{if} \quad u < \alpha \quad \text{else} \quad \underbrace{\mathcal{E}\left(\mathcal{E}\left(\boldsymbol{y}', \boldsymbol{p}^*\right), \tilde{\boldsymbol{t}}\right)}_{\text{output by applying insertion}}, \quad \boldsymbol{p}^* \sim \pi^*, \tilde{\boldsymbol{t}} \sim \pi_\theta\}$$

initial input

$u{\sim}uniform[0{,}1]$

obtained by sampling instead of doing argmax

- **Learning to Insert**

$$d_{\tilde{\pi}_{\mathrm{ins}}} = \{\underbrace{\mathcal{E}\left(\boldsymbol{y}^0, \boldsymbol{d}^*\right)}_{\text{deletion output}}, \quad \boldsymbol{d}^* \sim \pi^* \quad \text{if} \quad u < \beta \quad \text{else} \quad \underbrace{\mathcal{E}\left(\boldsymbol{y}^*, \tilde{\boldsymbol{d}}\right), \quad \tilde{\boldsymbol{d}} \sim \pi^{\mathrm{RND}}\}}_{\text{random word dropping sequence of the round-truth}}$$

$u{\sim}uniform[0{,}1]$

Expert Policy

- Oracle:

Levenshtein distance

$$\boldsymbol{a}^* = \arg\min_{\boldsymbol{a}} \mathcal{D}(\boldsymbol{y}^*, \mathcal{E}(\boldsymbol{y}, \boldsymbol{a}))$$

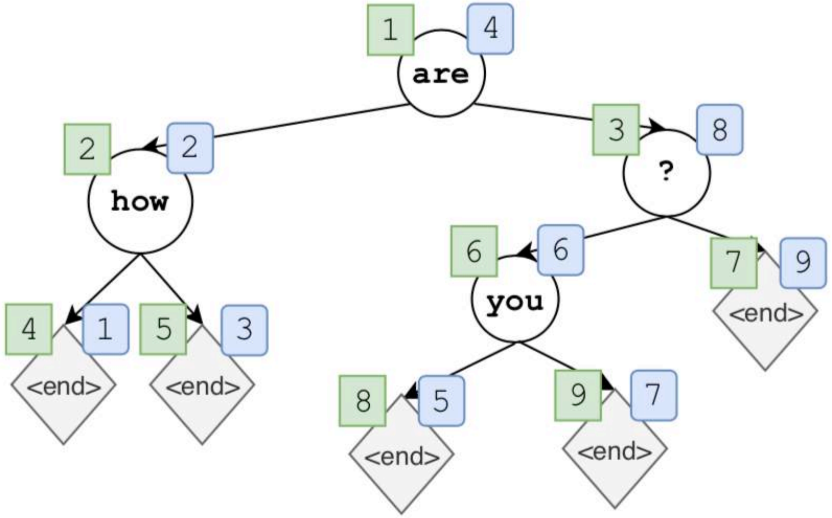
- Teacher Model:
  - first train an autoregressive teacher model using the same datasets and then replace the ground-truth sequence y^* by the beam-search result of this teacher-model, y^{AR}

Conclusion

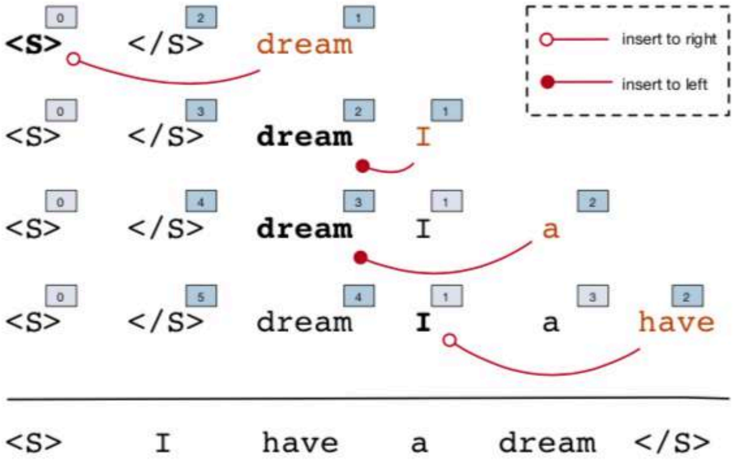
Parallel generation:

t	Canvas	Insertions
0	[]	(ate, 0)
1	[ate]	(friends, 0), (together, 1)
2	[friends, ate, together]	(three, 0), (lunch, 2)
3	[three, friends, ate, lunch, together]	(⟨EOS⟩, 5)

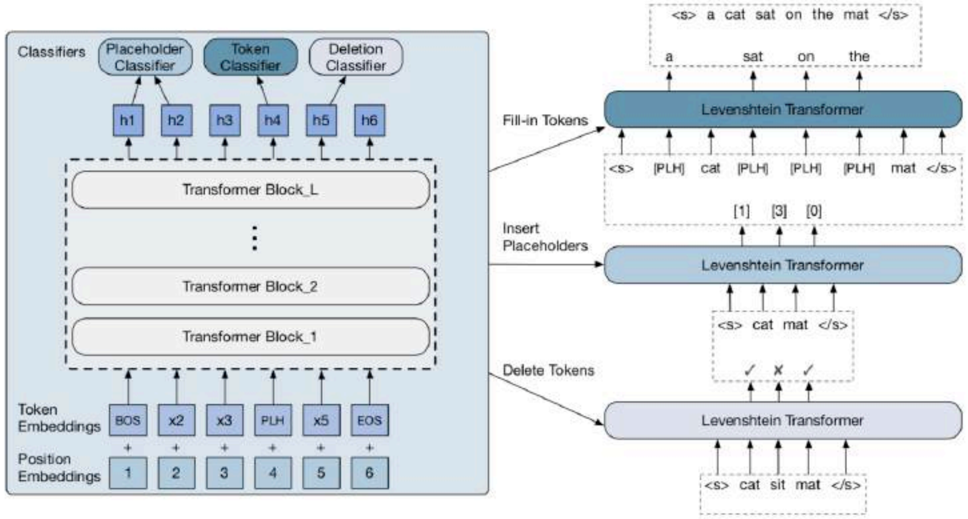
Insertion transformer



Non-Monotonic



InDIGO



Levenshtein

Paper List

Paper	Conference
Levenshtein Transformer	
Insertion-based Decoding with automatically Inferred Generation Order	
KERMIT: Generative Insertion-Based Modeling for Sequences	
Non-Monotonic Sequential Text Generation	ICML19
Insertion Transformer: Flexible Sequence Generation via Insertion Operations	ICML19
Sequence Generation: From Both Sides to the Middle	IJCAI19
Correct-and-Memorize:Learning to Translate from Interactive Revisions	IJCAI19
Non-Autoregressive Neural Machine Translation	ICLR18
The Importance of Generation Order in Language Modeling	EMNLP18

Reference

- 香侬读 | 按什么套路生成？基于插入和删除的序列生成方法 https://zhuanlan.zhihu.com/p/73417154
- https://cips-upload.bj.bcebos.com/ssatt2019/CIPS_SSATT_2019_问答系统_唐都钰_段楠.pdf

Thanks!