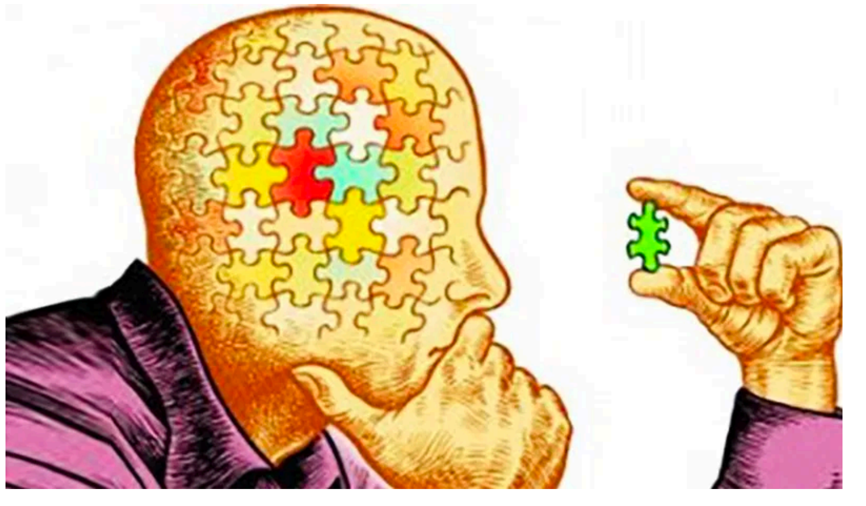
Rethinking the Generation Orders of Sequence

jcykcai



Why left-to-right?

- Humans do it
- But humans also do
  - First generate some abstract of what to say
  - Then serialize them

The Importance of Generation Order in Language Modeling

Nicolas Ford* **Daniel Duckworth** **Mohammad Norouzi** **George E. Dahl**
Google Brain
{nicf,duckworthd,mnorouzi,gdahl}@google.com

EMNLP18

Goal

- Better generation order?
- Wait! Does it really matter?

Framework

- Two-pass language models
  - Vocabulary partition: first-pass and second-pass tokens
  - Y = Y^1 + Y^2
    - Y^1 (template): only consist of first-pass tokens and special placeholders
    - Y^2 the rest second-pass tokens

Order Variants

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]

Language Models

- The total probability of a sentence y is

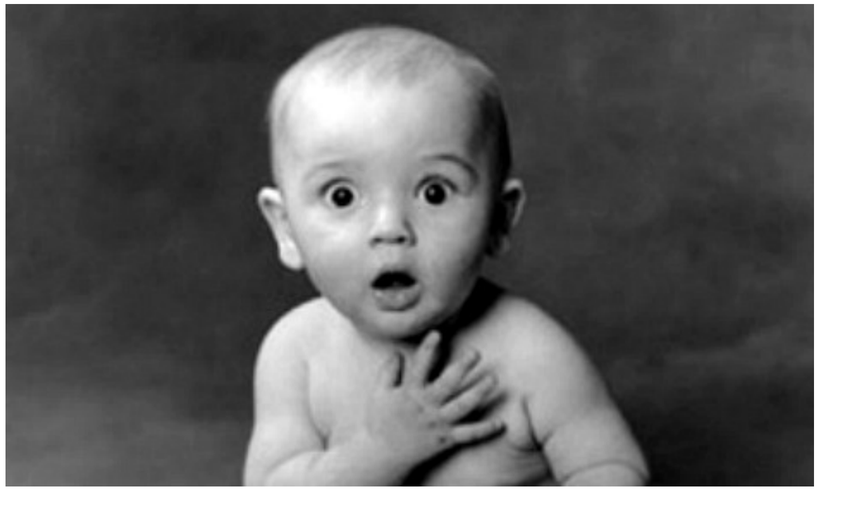
$$p(\mathbf{y}) \;=\; p_1(\mathbf{y}^{(1)})\, p_2(\mathbf{y}^{(2)} \mid \mathbf{y}^{(1)})$$

- The template y^1 is a deterministic function of y
- Template decoder + Template encoder + second-phrase decoder

Experiments

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

- PPL on LM1B
- Content-dependent generation orders do have a large effect on model quality
- Function-first is the best (common-first is the second)
  - It is easier to first decide syntactic structure
  - Delay the rare tokens



Recent Advances

https://arxiv.org/pdf/1902.01370.pdf
https://arxiv.org/pdf/1902.02192.pdf
https://arxiv.org/pdf/1902.03249.pdf

Insertion Transformer:
Flexible Sequence Generation via Insertion Operations

Mitchell Stern [1 2] **William Chan** [1] **Jamie Kiros** [1] **Jakob Uszkoreit** [1]

ICML19

Model

- Architecture
  - Transformer with full self-attention decoder
  - Slot representations
- Content-location distribution
  - What to insert & where to insert
  - $p(c, l \mid x, \hat{y}_t) = \mathrm{InsertionTransformer}(x, \hat{y}_t).$

Termination

- Termination conditions
  - Sequence finalization
  - Slot finalization (enable parallel inference)

Serial generation:

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

Parallel generation:

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

Figure 1. Examples demonstrating how the clause "three friends ate lunch together" can be generated using our insertion framework. On the left, a serial generation process is used in which one insertion is performed at a time. On the right, a parallel generation process is used with multiple insertions being allowed per time step. Our model can either be trained to follow specific orderings or to maximize entropy over all valid actions. Some options permit highly efficient parallel decoding, as shown in our experiments.

Training

- The form of single training instances
    - Sample generation steps (partial sentences)
- Variants
    - Left-to-right
    - Balanced Binary Tree
    - Uniform

Results

Loss	Termination	BLEU (+EOS)	BLEU (+EOS)	BLEU (+EOS)
			+Distillation	+Distillation, +Parallel
Left-to-Right	Sequence	20.92 (20.92)	23.29 (23.36)	-
Binary Tree ($\tau = 0.5$)	Slot	20.35 (21.39)	24.49 (25.55)	25.33 (25.70)
Binary Tree ($\tau = 1.0$)	Slot	21.02 (22.37)	24.36 (25.43)	25.43 (25.76)
Binary Tree ($\tau = 2.0$)	Slot	20.52 (21.95)	24.59 (25.80)	25.33 (25.80)
Uniform	Sequence	19.34 (22.64)	22.75 (25.45)	-
Uniform	Slot	18.26 (22.16)	22.39 (25.58)	24.31 (24.91)

- +Parallel is even better!
    - Greedy search may suffer from issues related to local search that are circumvented by making multiple updates to the hypothesis at once.

Results

Model	BLEU	Iterations
Autoregressive Left-to-Right		
Transformer (Vaswani et al., 2017)	27.3	n
Semi-Autoregressive Left-to-Right		
SAT (Wang et al., 2018)	24.83	$n/6$
Blockwise Parallel (Stern et al., 2018)	27.40	$\approx n/5$
Non-Autoregressive		
NAT (Gu et al., 2018)	17.69	1
Iterative Refinement (Lee et al., 2018)	21.61	10
Our Approach (Greedy)		
Insertion Transformer + Left-to-Right	23.94	n
Insertion Transformer + Binary Tree	27.29	n
Insertion Transformer + Uniform	27.12	n
Our Approach (Parallel)		
Insertion Transformer + Binary Tree	27.41	$\approx \log_2 n$
Insertion Transformer + Uniform	26.72	$\approx \log_2 n$

- Comparable performance
- Fewer generation iteration => faster?

Limitations

- Must recompute the decoder hidden stat for each position after each insertion
- Auto-regressive vs. non-autoregressive
  - Expressive power vs. parallel decoding

Non-Monotonic Sequential Text Generation

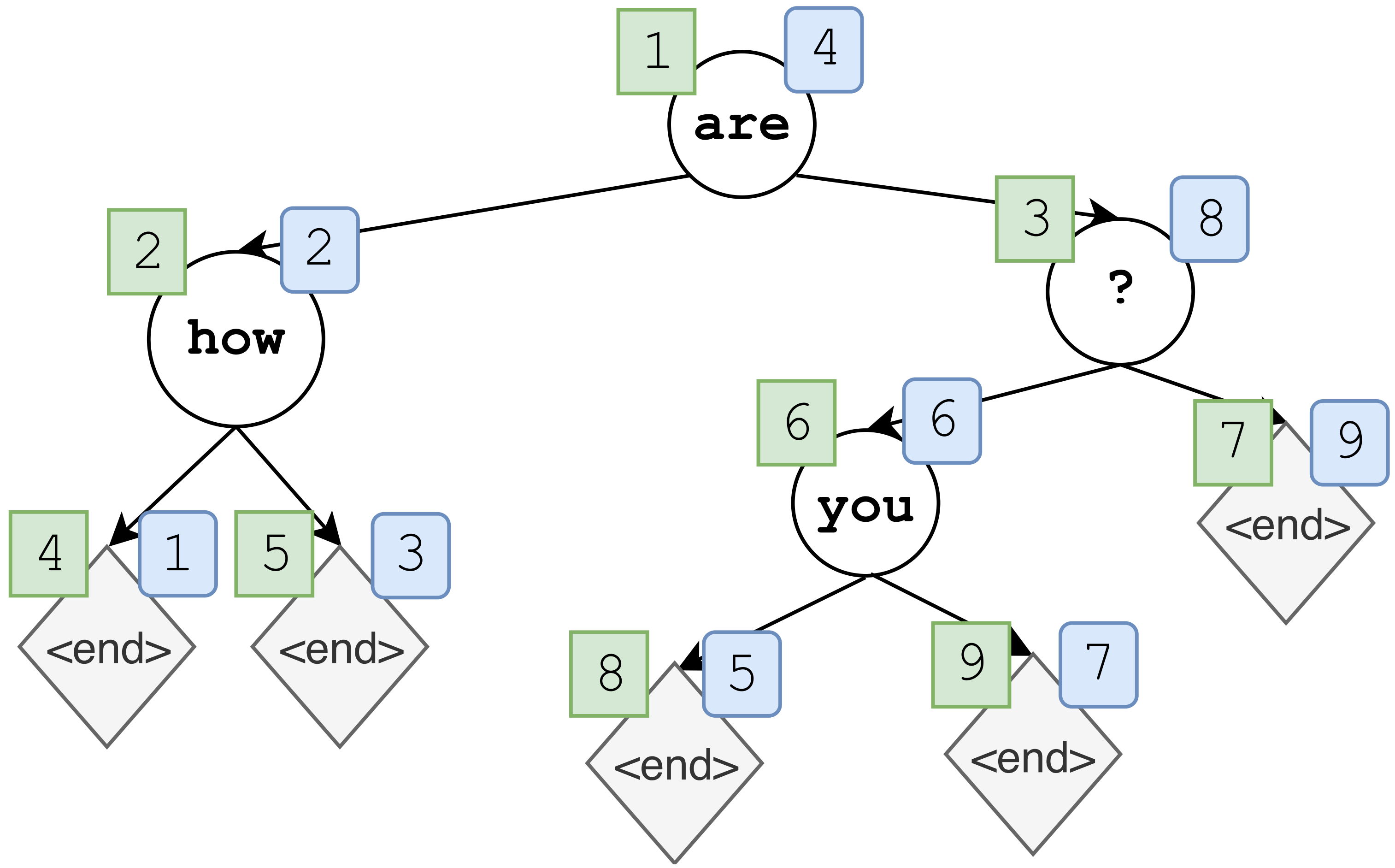
Sean Welleck [1] Kianté Brantley [2] Hal Daumé III [2 3] Kyunghyun Cho [1 4 5]

ICML19

Goal

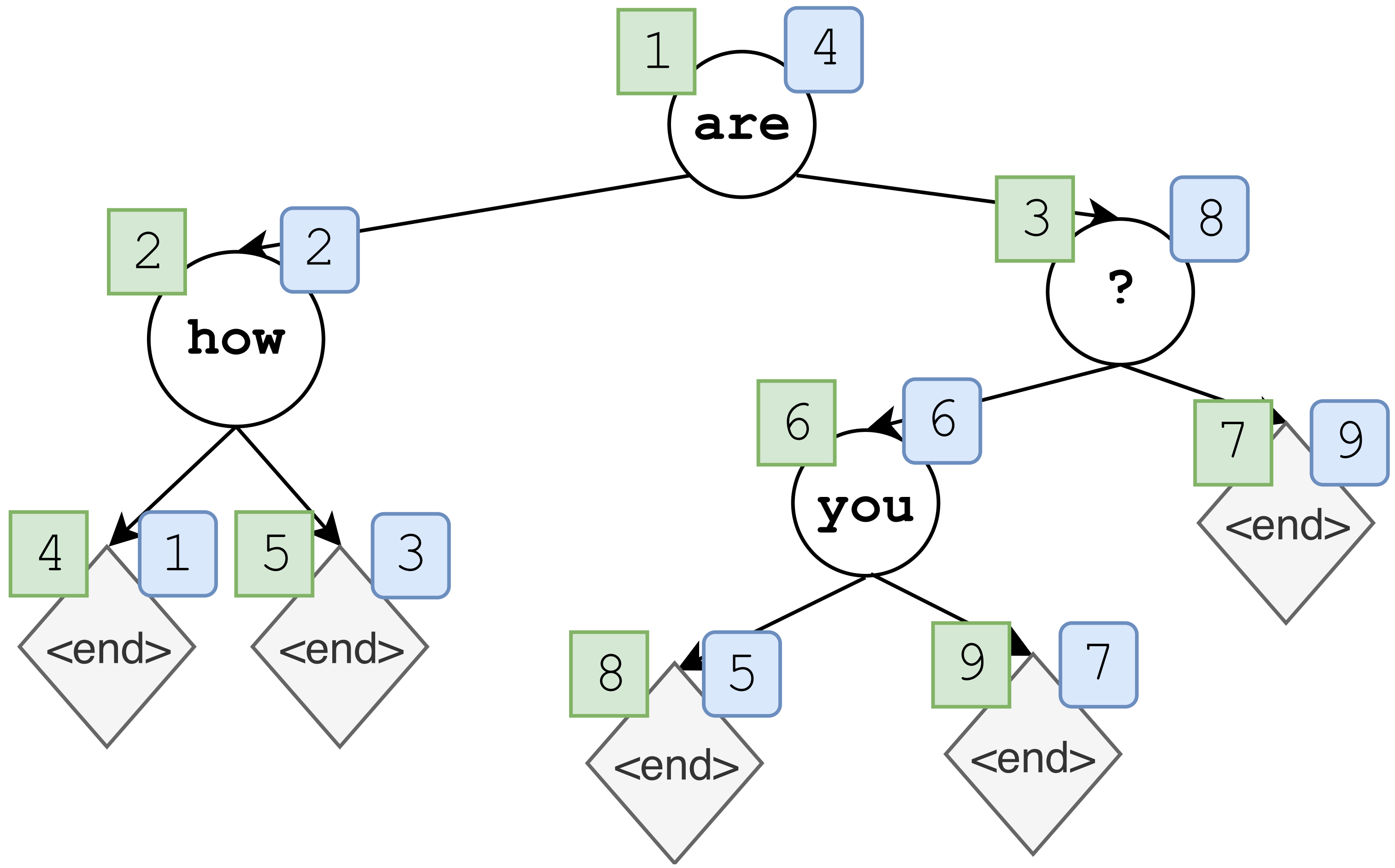
- Learn a good order without
  - specifying an order in advance.
  - additional annotation

Formulation



- Generating a word at an arbitrary position, then recursively generating words to its left and words to its right.

Formulation



- The full generation is performed in a level-order traversal. (green)
- The output is read off from an in-order traversal. (blue)

Imitation Learning

- Learn a generation policy that mimics the actions of an oracle generation policy
- Oracle policies
  - Uniform oracle: similar to quick-sort
  - Coaching oracle: reinforce the policy's own preferences $\pi^*_{\text{coaching}}(a|s) \propto \pi^*_{\text{uniform}}(a|s)\ \pi(a|s)$
  - Annealed coaching oracle: $\pi^*_{\text{annealed}}(a|s) = \beta\pi^*_{\text{uniform}}(a|s) + (1-\beta)\pi^*_{\text{coaching}}(a|s)$

Imitation Learning

- Annealed coaching oracle
    - Random oracle encourages exploration
    - Reinforcement leads to a specific generation order
- A special case for comparison
    - Deterministic Left-to-Right Oracle (standard order)

Policy Networks

- Partial binary tee is considered as a flat sequence of nodes in a level-order traversal.
- Essentially, still a sequence model
- Transformer, LSTM can be applied.

Experiments

- Language Modeling on Persona-Chat dataset

Oracle	**%Novel**	**%Unique**	**Avg. Tokens**	**Avg. Span**	**BLEU**
left-right	17.8	97.0	11.9	1.0	47.0
uniform	98.3	99.9	13.0	1.43	40.0
annealed	93.1	98.2	10.6	1.31	56.2
Validation	97.0	100	12.1	-	-

Table 1. Statistics computed over 10,000 sampled sentences (in-order traversals of sampled trees with $\langle end \rangle$ tokens removed) for policies trained on Persona-Chat. A sample is novel when it is not in the training set. Percent unique is the cardinality of the set of sampled sentences divided by the number of sampled sentences.

Experiments

```
Sentence:   i wish you could study lol . i work a lot .
Gen. Order: . you . i study i wish could lol a work lot

          _________________._____________
         /                               \
  _____you________                ___________.
 /                \              /
i_               __study_       i______
  \             /        \            \
  wish       could       lol         __a_
                                    /    \
                               work    lot
```

```
Sentence:   oh , i am a big fan of dairy myself . i am a receptionist .
Gen. Order: . i . , am i oh a am of receptionist fan dairy a big myself

       _____________________________.____________________
      /                                                   \
     i                                  __________________.
    / \                                /
  _,   am                             i
 /       \                             \
oh        a________                     am________
                   \                             \
                   _of__                     receptionist
                  /     \                   /
               _fan    dairy__             a
              /               \
            big              myself
```

- By POS analysis on different levels of the trees
  - Punctuation-first => easy-first
  - Pronoun before noun and verb => like dependency tree

Experiments

- Machine translation

Oracle	Validation BLEU (BP)	Validation Meteor	Validation YiSi	Validation Ribes	Test BLEU (BP)	Test Meteor	Test YiSi	Test Ribes
left-right	32.30 (0.95)	31.96	69.41	84.80	28.00 (1.00)	30.10	65.22	82.29
uniform	24.50 (0.84)	27.98	66.40	82.66	21.40 (0.86)	26.40	62.41	80.00
annealed	26.80 (0.88)	29.67	67.88	83.61	23.30 (0.91)	27.96	63.38	80.91
+tree-encoding	28.00 (0.86)	30.15	68.43	84.36	24.30 (0.91)	28.59	63.87	81.64
+⟨end⟩-tuning	29.10 (0.99)	31.00	68.81	83.51	24.60 (1.00)	29.30	64.18	80.53

- BLEU focuses on getting a large number of 4-grams correct
- The other three measures are less sensitive to exact word order and focus more on semantics.

Limitations

- Binary-tree => N-ary tree
- Only produce a subset of all possible generation orders
    - Projective generation, no crossing of two edges when nodes are lined up following the ignorer traversal.

Insertion-based Decoding with automatically Inferred Generation Order

Jiatao Gu[†], Qi Liu[†] and Kyungyhyun Cho[†‡]

[†]Facebook AI Research
[‡]New York University, CIFAR Azrieli Global Scholar
[†]{jgu, qiliu, kyunghyuncho}@fb.com

Goal

- How can we decode a sequence in its best order?

Model Design

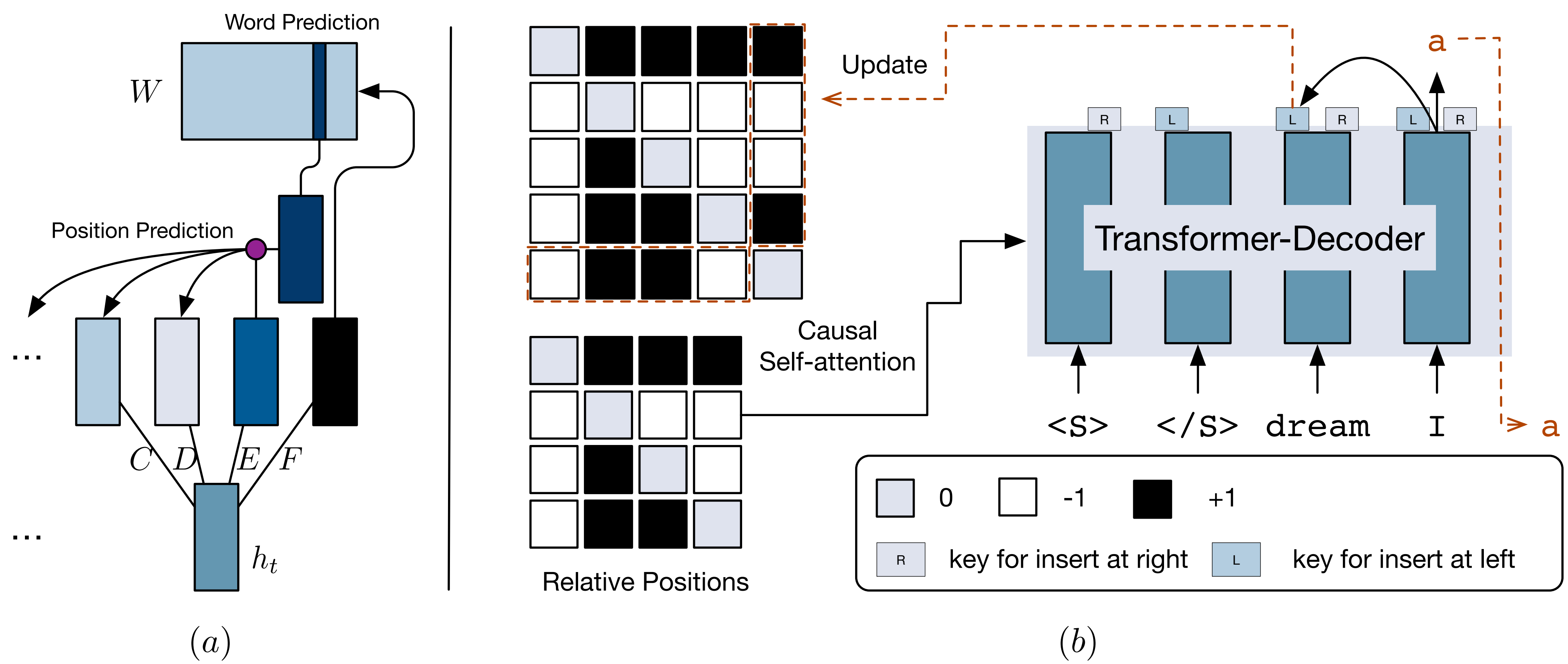
- Insertion-based (again)
  - Joint prediction of position and token
- The problem of absolute position
  - Changes over decoding time (recomputing is costly!)

Relative Positions

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

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

Decoding



Word Prediction
W
Position Prediction
...
C D E F
h_t
(a)
Update
a
R L L R L R
Transformer-Decoder
Causal Self-attention
<S> </S> dream I
a
Relative Positions
0 -1 +1
R key for insert at right
L key for insert at left
(b)

Learning

- Maximize the evident lower bound (ELBO)
- Approximate posterior distribution of generation orders $q(\pi \,|\, x, y)$

$$\mathcal{L}_{\text{ELBO}} = \mathop{\mathbb{E}}_{\boldsymbol{\pi} \sim q} \log p_\theta(\boldsymbol{y}_{\boldsymbol{\pi}} | \boldsymbol{x}) + \mathcal{H}(q)$$

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

Searched Adaptive Order (SAO)

- $q(\pi|x, y)$ is approximated by beam search

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

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

Experiments

Pre-defined Order	Descriptions
Left-to-right (L2R)	Generate words from left to right. (Wu et al., 2018)
Right-to-left (R2L)	Generate words from right to left. (Wu et al., 2018)
Odd-Even (ODD)	Generate words at odd positions from left to right, then generate even positions. (Ford et al., 2018)
Balanced-tree (BLT)	Generate words with a top-down left-to-right order from a balanced binary tree. (Stern et al., 2019)
Syntax-tree (SYN)	Generate words with a top-down left-to-right order from the dependency tree. (Wang et al., 2018b)
Common-First (CF)	Generate all common words first from left to right, and then generate the others. (Ford et al., 2018)
Rare-First (RF)	Generate all rare words first from left to right, and then generate the remaining. (Ford et al., 2018)
Random (RND)	Generate words in a random order shuffled every time the example was loaded.

	WMT16 Ro $\rightarrow$ En				WMT18 En $\rightarrow$ Tr				KFTT En $\rightarrow$ Ja			
Model	BLEU	Ribes	Meteor	TER	BLEU	Ribes	Meteor	TER	BLEU	Ribes	Meteor	TER
RND	20.20	79.35	41.00	63.20	03.04	55.45	19.12	90.60	17.09	70.89	35.24	70.11
L2R	31.82	83.37	52.19	50.62	14.85	69.20	33.90	**71.56**	30.87	77.72	48.57	59.92
R2L	31.62	83.18	52.09	50.20	14.38	68.87	33.33	71.91	30.44	**77.95**	47.91	61.09
ODD	30.11	83.09	50.68	50.79	13.64	68.85	32.48	72.84	28.59	77.01	46.28	60.12
BLT	24.38	81.70	45.67	55.38	08.72	65.70	27.40	77.76	21.50	73.97	40.23	64.39
SYN	29.62	82.65	50.25	52.14	–				–			
CF	30.25	83.22	50.71	50.72	12.04	67.61	31.18	74.75	28.91	77.06	46.46	61.56
RF	30.23	83.29	50.72	51.73	12.10	67.44	30.72	73.40	27.35	76.40	45.15	62.14
SAO	**32.47**	**84.10**	**53.00**	**49.02**	**15.18**	**70.06**	**34.60**	**71.56**	**31.91**	77.56	**49.66**	**59.80**

XLNet: Generalized Autoregressive Pretraining for Language Understanding

Zhilin Yang[*1], **Zihang Dai**[*12], **Yiming Yang**[1], **Jaime Carbonell**[1],
Ruslan Salakhutdinov[1], **Quoc V. Le**[2]
[1]Carnegie Mellon University, [2]Google Brain
{zhiliny,dzihang,yiming,jgc,rsalakhu}@cs.cmu.edu, qvl@google.com

XLNet: Generali
for Lan

Zhilin Yang*[1], Ziha
Rusla
[1]Carnegie
{zhiliny,dzihang,yimi

Top Comments Comment

Vincent 171
大佬，快停停，跟不上了

G-500 43
求你们慢点吧。。。我跟不上

洋洋同学 33
大佬，慢一点，我不想老是吃尾气

Champion 22
跟不上了呀，BERT还没应用明白呀

Author 6
加油～

鸡丁 16
哭了，营养跟不上了😭

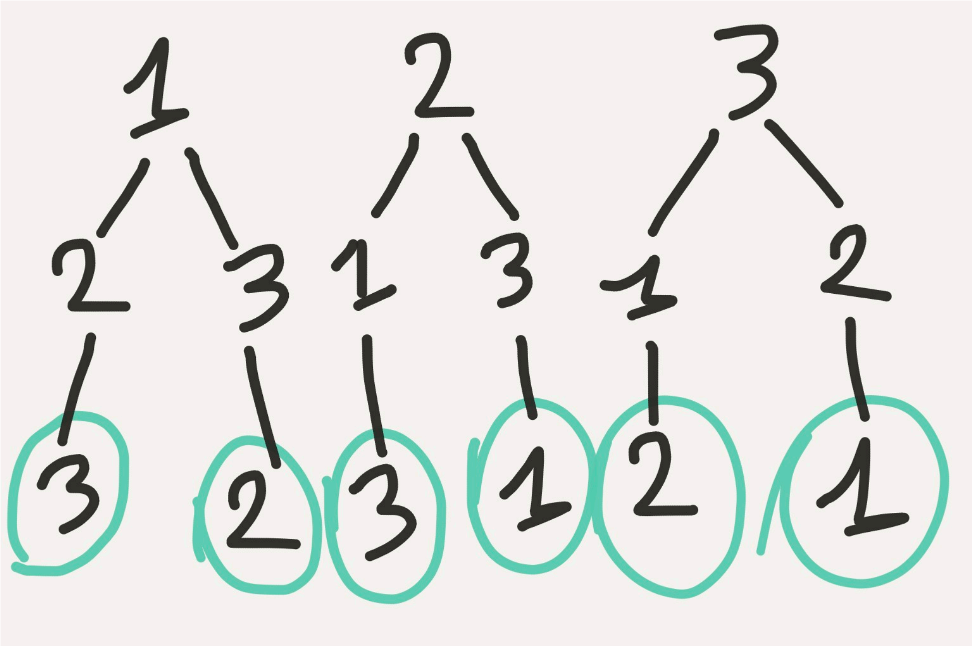
???

BERT

- Motivation of BERT: utilize bidirectional context
- Solution of BERT: denoising auto-encoder
- Problem of BERT:
    - pretrain-finetune discrepancy (the mask symbol)
    - Independent assumption (non-autoregressive)

XLNet

- Left-to-right ? No
- Right-to-left ? No
- Both ? No
- All possible factorization orders



Benefits

- Still an auto-regressive model
  - Learn to utilize bidirectional context
  - No data corruption, no pretrain-finetune discrepancy
  - No independent assumption, more expressive

Lesson

- Given aforementioned papers, the idea of XLNet seems very natural.
- It is not hard to make a BIG NEWS if we
  - Always think of fundamental problems
  - Read some good papers
  - Have TPUs

Other Techniques

- Transformer-XL
- Partial prediction
    - only predict the last tokens in a factorization order
- Span-based prediction
    - mask a consecutive span

Ablation Study

#	Model	RACE	SQuAD2.0 F1	SQuAD2.0 EM	MNLI m/mm	SST-2
1	BERT-Base	64.3	76.30	73.66	84.34/84.65	92.78
2	DAE + Transformer-XL	65.03	79.56	76.80	84.88/84.45	92.60
3	XLNet-Base ($K = 7$)	66.05	**81.33**	**78.46**	**85.84/85.43**	92.66
4	XLNet-Base ($K = 6$)	66.66	80.98	78.18	85.63/85.12	**93.35**
5	- memory	65.55	80.15	77.27	85.32/85.05	92.78
6	- span-based pred	65.95	80.61	77.91	85.49/85.02	93.12
7	- bidirectional data	66.34	80.65	77.87	85.31/84.99	92.66
8	+ next-sent pred	**66.76**	79.83	76.94	85.32/85.09	92.89

- The new permutation LM objective is superior.
- The transformer-XL, span-based pred, etc also matter.

Discussions

- Why token-by-token?
- Can we do deletion and substitution?