Text Editing as Imitation Game

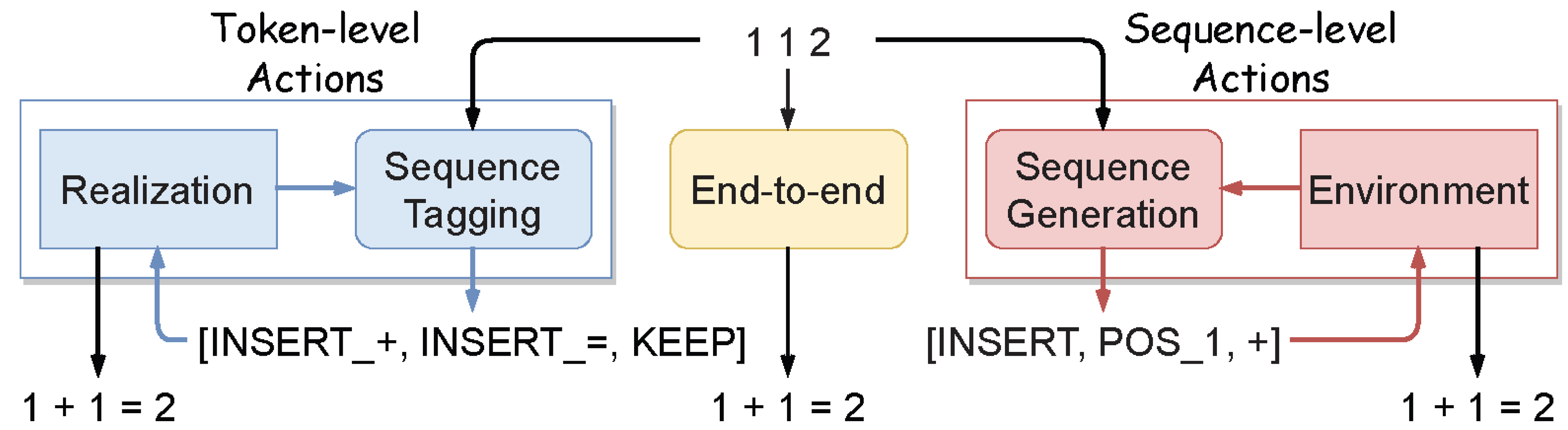
Ning Shi, Bin Tang, Bo Yuan, Longtao Huang, Yewen Pu, Jie Fu, Zhouhan Lin

ning.shi@ualberta.ca, {tangbin.tang,qiufu.yb,kaiyang.hlt}@alibaba-inc.com, yewen.pu@autodesk.com, fujie@baai.ac.cn, lin.zhouhan@gmail.com

Introduction

Text editing, such as grammatical error correction, arises naturally from imperfect textual data.
Two primary methods to solve text editing:

- End-to-end
- Sequence tagging (token-level action generation



End-to-end

Pros - the advantage of simplicity by giving direct input-output pairs
Cons - can struggle in carrying out localized, specific fixes while keeping the rest of the sequence intact

Sequence Tagging

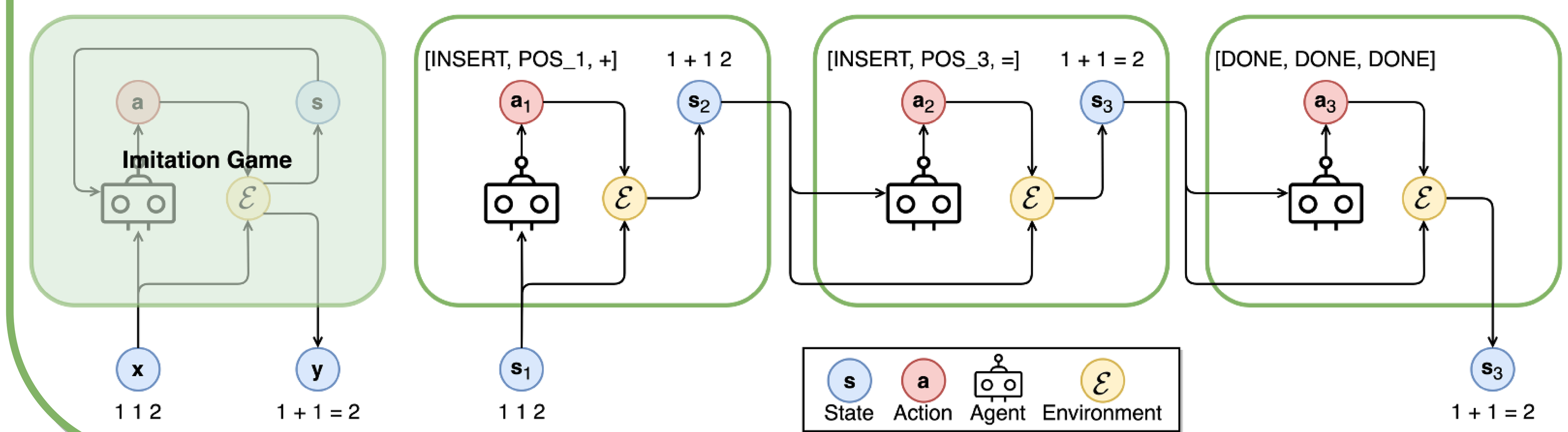
Pros - appropriate when outputs highly overlap with inputs by assigning no-op (e.g., KEEP)
Cons - action space is limited to token-level, such as deletion or insertion after a token

Imitation Game

Our Markov Decision Process (MDP) is defined as follows.

State S - a set of text sequences
Action A - a set of action sequences
Transition matrix P - the probability that a_t leads s_t to s_{t+1}
Environment $\mathscr{E}$ - to update state by $s_{t+1}=\mathscr{E}(s_t, a_t)$
Reward function R - to calculate a reward for each action

The formulation turns out to be a simplified $M_{BC}=(S, A, \mathscr{E})$.



Our Contributions are summarized as follows.

- Frame text editing into an imitation game formally defined as an MDP, allowing the highest degrees of flexibility to design actions at the sequence level
- Involve Trajectory Generation (TG) to translate input-output data to state-action demonstrations for imitation learning
- Propose a corresponding Trajectory Augmentation (TA) technique to mitigate the distribution shift issue imitation learning often suffers from
- Introduce Dual Decoders (D2), a novel non-autoregressive decoder to boost imitation learning in terms of accuracy, efficiency, and robustness.
- The source code and datasets have been released to the public (please scan the QR codes at the bottom).

Trajectory Generation (TG)

Q: how to convert conventional sequence-to-sequence data into state-to-action demonstrations?

A: dynamic programming (DP) to calculate the minimum edit distance given the edit metric and back trace the editing operation after that.

Algorithm 1 Trajectory Generation (TG)

Input: Initial state $\mathbf{x}$, goal state $\mathbf{y}$, environment $\mathcal{E}$, and edit metric $\mathbf{E}$.
Output: Trajectories τ.
$\tau \leftarrow \emptyset$
$\mathbf{s} \leftarrow \mathbf{x}$
$ops \leftarrow \mathrm{DP}(\mathbf{x}, \mathbf{y}, E)$
for $op \in ops$ **do**
  $\mathbf{a} \leftarrow \mathrm{Action}(op)$ ▷ Translate operation to action
  $\tau \leftarrow \tau \cup [(\mathbf{s}, \mathbf{a})]$
  $\mathbf{s} \leftarrow \mathcal{E}(\mathbf{s}, \mathbf{a})$
end for
$\tau \leftarrow \tau \cup [(\mathbf{s}, \mathbf{a}_T)]$ ▷ Append goal state and output action
return τ

Trajectory Augmentation (TA)

Q: imitation learning often suffers from distribution shift and error accumulation. How to handle this?

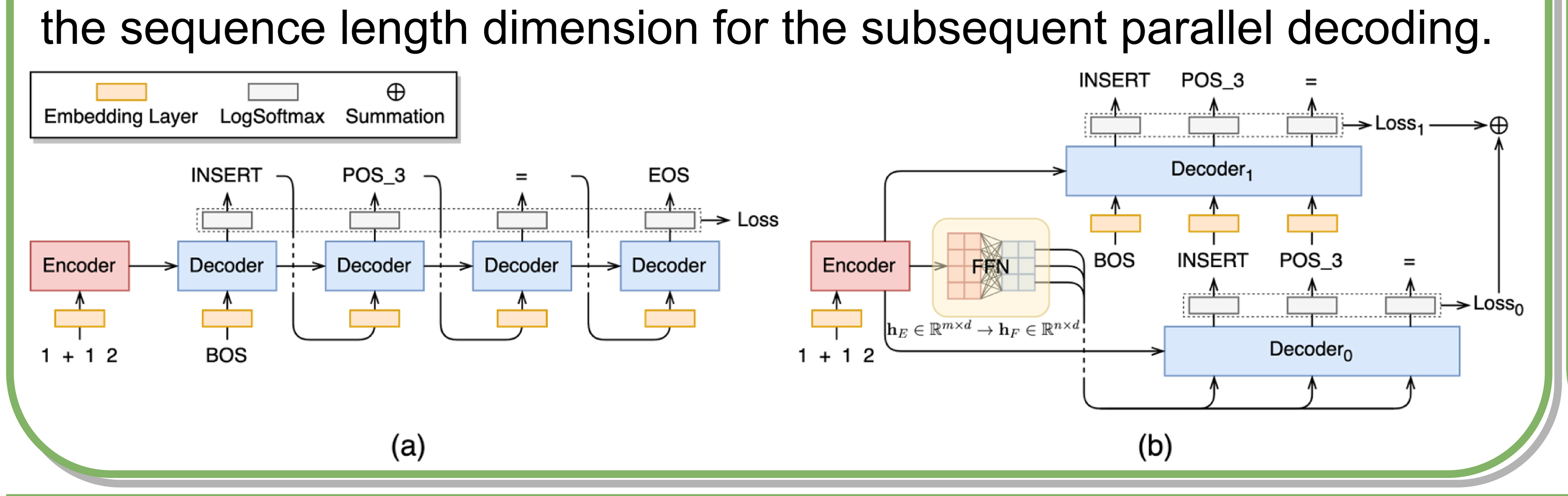
A: expand the training set by actively exposing shifted states via TA that utilizes the divide-and-conquer technique to drop out actions from demonstrations.

Algorithm 2 Trajectory Augmentation (TA)

Input: States $\mathbf{S}$, state $\mathbf{s}_t$, expert states $\mathbf{S}^*$, actions $\mathbf{A}$, and environment $\mathcal{E}$.
Output: Augmented states $\mathbf{S}$.
if $|\mathbf{A}| > 1$ **then**
  $\mathbf{a}_t \leftarrow \mathbf{A}.\mathrm{pop}(0)$
  $\mathbf{s}_{t+1} \leftarrow \mathcal{E}(\mathbf{s}_t, \mathbf{a}_t)$
  $\mathbf{S} \leftarrow \mathbf{S} \cup \mathrm{TA}(\mathbf{S}, \mathbf{s}_{t+1}, \mathbf{S}^*, \mathbf{A}, \mathcal{E})$ ▷ Execute action
  $\mathbf{A} \leftarrow \mathrm{Update}(\mathbf{A}, \mathbf{s}_t, \mathbf{s}_{t+1})$
  $\mathbf{S} \leftarrow \mathbf{S} \cup \mathrm{TA}(\mathbf{S}, \mathbf{s}_t, \mathbf{S}^*, \mathbf{A}, \mathcal{E})$ ▷ Skip action
else if $\mathbf{s}_t \notin \mathbf{S}^*$ **then**
  $\mathbf{S} \leftarrow \mathbf{S} \cup [\mathbf{s}_t]$ ▷ Merge shifted state
end if
return $\mathbf{S}$

Dual Decoders (D2)

The conventional autoregressive decoder (a) compared with the proposed non-autoregressive D2 (b) in which the linear layer aligns the sequence length dimension for the subsequent parallel decoding.



Arithmetic Equation (AE) Benchmarks

Arithmetic Operators Restoration (AOR), Arithmetic Equation Simplification (AES), and Arithmetic Equation Correction (AEC)

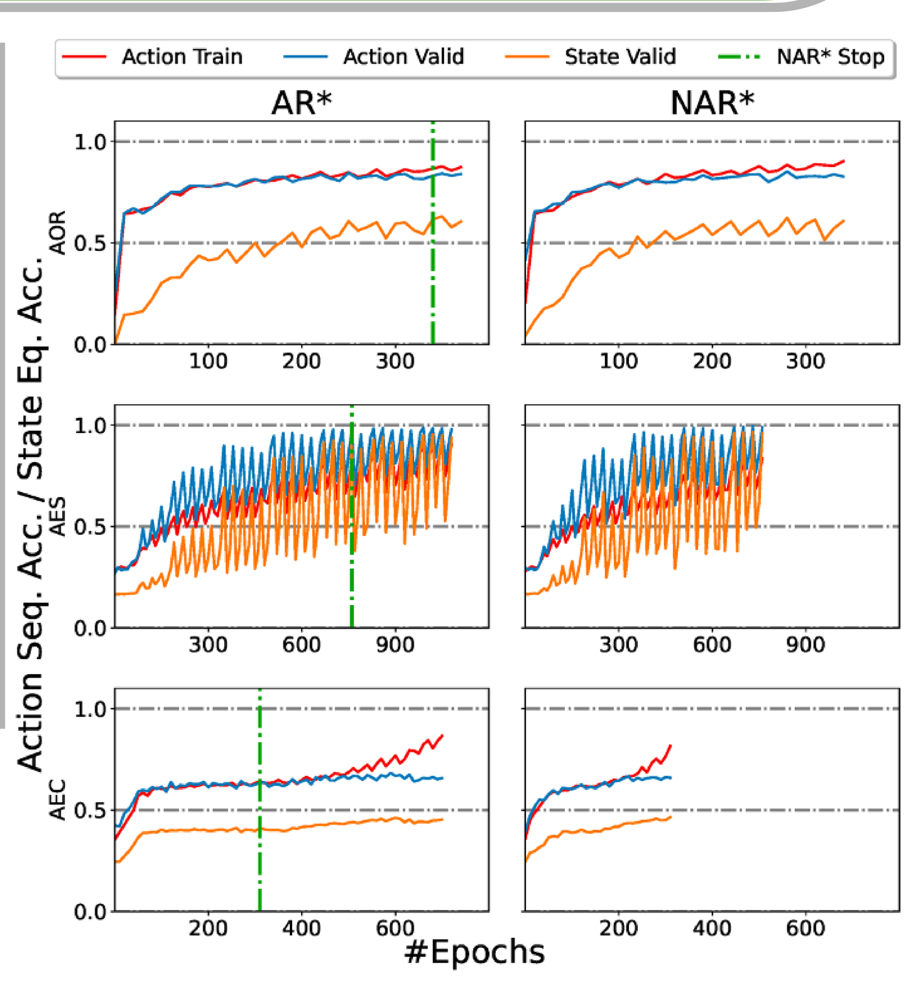
AOR ($N=10, L=5, D=10K$)			**AES** ($N=100, L=5, D=10K$)			**AEC** ($N=10, L=5, D=10K$)		
Train/Valid/Test	**Train TA**	**Traj. Len.**	**Train/Valid/Test**	**Train TA**	**Traj. Len.**	**Train/Valid/Test**	**Train TA**	**Traj. Len.**
7,000/1,500/1,500	145,176	6	7,000/1,500/1,500	65,948	6	7,000/1,500/1,500	19,764	4

Term	**AOR** ($N=10, L=5, D=10K$)	**AES** ($N=100, L=5, D=10K$)	**AEC** ($N=10, L=5, D=10K$)
Source $\mathbf{x}$	3 6 2 9 3	65 + (25 - 20) - (64 + 32) + (83 - 24) = (- 25 + 58)	- 2 * + 4 10 + 8 / 8 = 8
Target $\mathbf{y}$	- 3 - 6 / 2 + 9 = 3	65 + 5 - 96 + 59 = 33	- 2 + 10 * 8 / 8 = 8
State $\mathbf{s}_t^*$	- 3 - 6 / 2 9 3	65 + 5 - (64 + 32) + (83 - 24) = (- 25 + 58)	- 2 + **4** 10 + 8 / 8 = 8
Action $\mathbf{a}_t^*$	[POS_6, +]	[POS_4, POS_8, 96]	[DELETE, POS_3, POS_3]
Next State $\mathbf{s}_{t+1}^*$	- 3 - 6 / 2 **+** 9 3	65 + 5 - **96** + (83 - 24) = (- 25 + 58)	- 2 + 10 + 8 / 8 = 8
Shifted State $\mathbf{s}_t'$	- 3 - 6 / 2 9 **=** 3	65 + 5 - (64 + 32) + **59** = (- 25 + 58)	- 2 + 4 10 ***** 8 / 8 = 8

Experimental Results

Method	**AOR** ($N=10, L=5, D=10K$) Tok. Acc. %	Seq. Acc. %	Eq. Acc. %	**AES** ($N=100, L=5, D=10K$) Tok. Acc. %	Eq. Acc. %	**AEC** ($N=10, L=5, D=10K$) Tok. Acc. %	Seq. Acc. %	Eq. Acc. %
End2end	–	–	29.33	84.60	25.20	88.08	57.27	57.73
Tagging	–	–	51.40	87.00	36.67	84.46	46.93	47.33
Recurrence	–	–	58.53	98.63	87.73	83.64	57.47	58.27
Recurrence*	60.30 ± 1.30	27.31 ± 1.33	56.73 ± 1.33	79.82 ± 0.37	22.28 ± 0.52	82.32 ± 0.56	41.72 ± 0.74	42.13 ± 0.75
AR	61.85 ± 0.51	28.83 ± 1.14	59.09 ± 0.95	88.12 ± 2.37	37.05 ± 6.57	$\mathbf{82.61 \pm 0.53}$	45.81 ± 0.36	46.31 ± 0.31
AR*	62.51 ± 0.62	$\mathbf{30.85 \pm 0.41}$	61.35 ± 0.33	99.27 ± 0.32	93.57 ± 2.91	82.29 ± 0.39	45.99 ± 0.49	46.35 ± 0.52
NAR	59.72 ± 0.70	24.16 ± 1.16	51.64 ± 1.97	83.87 ± 1.60	29.49 ± 2.51	80.28 ± 0.76	44.91 ± 1.71	45.40 ± 1.78
NAR*	$\mathbf{62.81 \pm 0.89}$	30.13 ± 1.31	$\mathbf{61.45 \pm 1.61}$	$\mathbf{99.51 \pm 0.13}$	$\mathbf{95.67 \pm 0.93}$	81.82 ± 0.68	$\mathbf{45.97 \pm 1.07}$	$\mathbf{46.43 \pm 1.10}$
AR +TA	62.35 ± 0.61	32.28 ± 0.67	63.56 ± 1.06	88.05 ± 1.20	38.39 ± 3.45	$\mathbf{83.94 \pm 0.42}$*	49.36 ± 1.23	49.83 ± 1.21
AR* +TA	62.58 ± 0.63	33.01 ± 1.31	65.73 ± 1.38	99.44 ± 0.27	95.24 ± 2.38	83.39 ± 0.74	48.95 ± 0.65	49.47 ± 0.73
NAR +TA	61.30 ± 0.86	32.04 ± 1.99	63.75 ± 2.08	90.38 ± 2.21	47.91 ± 8.18	81.36 ± 0.40	48.01 ± 1.07	48.47 ± 1.15
NAR* +TA	$\mathbf{63.48 \pm 0.38}^*$	$\mathbf{34.23 \pm 0.92}^*$	$\mathbf{67.13 \pm 0.99}^*$	$\mathbf{99.58 \pm 0.15}^*$	$\mathbf{96.44 \pm 1.29}^*$	82.70 ± 0.42	$\mathbf{49.64 \pm 0.59}^*$	$\mathbf{50.15 \pm 0.55}^*$

Design	Action Sequence	Method	Tok. Acc. %	Eq. Acc. %
#1	[Pos.$_L$, Pos.$_R$, Tok.]	AR*	99.27 ± 0.32	93.57 ± 2.91
		NAR*	$\mathbf{99.51 \pm 0.13}$	$\mathbf{95.67 \pm 0.93}$
		AR* +TA	99.44 ± 0.27	95.24 ± 2.38
		NAR* +TA	$\mathbf{99.58 \pm 0.15}^*$	$\mathbf{96.44 \pm 1.29}^*$
#2	[Pos.$_L$, Tok., Pos.$_R$]	AR*	99.08 ± 0.93	92.35 ± 7.21
		NAR*	$\mathbf{99.50 \pm 0.27}$	$\mathbf{95.55 \pm 2.28}$
		AR* +TA	99.52 ± 0.29	95.68 ± 2.49
		NAR* +TA	$\mathbf{99.54 \pm 0.20}^*$	$\mathbf{95.97 \pm 1.64}^*$
#3	[Tok., Pos.$_L$, Pos.$_R$]	AR*	98.06 ± 0.79	83.79 ± 6.25
		NAR*	$\mathbf{99.53 \pm 0.14}$	$\mathbf{95.99 \pm 0.81}$
		AR* +TA	98.43 ± 0.49	87.29 ± 3.70
		NAR* +TA	$\mathbf{99.61 \pm 0.06}^*$	$\mathbf{96.55 \pm 0.46}^*$



arXiv GitHub YouTube

This work was supported by Shining Lab and Alibaba Group.

@EMNLP2022