Text Editing as Imitation Game

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

Introduction

Text Editing

- Text simplification (e.g., dyslexia friendly)
- Grammatical error correction (e.g., Grammarly)
- Post processing (e.g., MT)
- Punctuation restoration (e.g., ASR)
- To name a few

Source Text (x)

1 1 2

↓

Target Text (y)

1 + 1 = 2

amii

Introduction

From End to End (End2end)

- Simplicity
- Good results
- Not much effort

But

- Copy mechanism
- Translate overlap

Source Text (x)

1 1 2 <pad>

Target Text (y)

<s> 1 + 1 = 2 </s>

Introduction

Sequence Tagging

(Token-level Action Generation)

- Tag <keep> for overlap

But

- Action bounded by token

Source Text (x)

1 1 2

↓

Target Tag (y′)

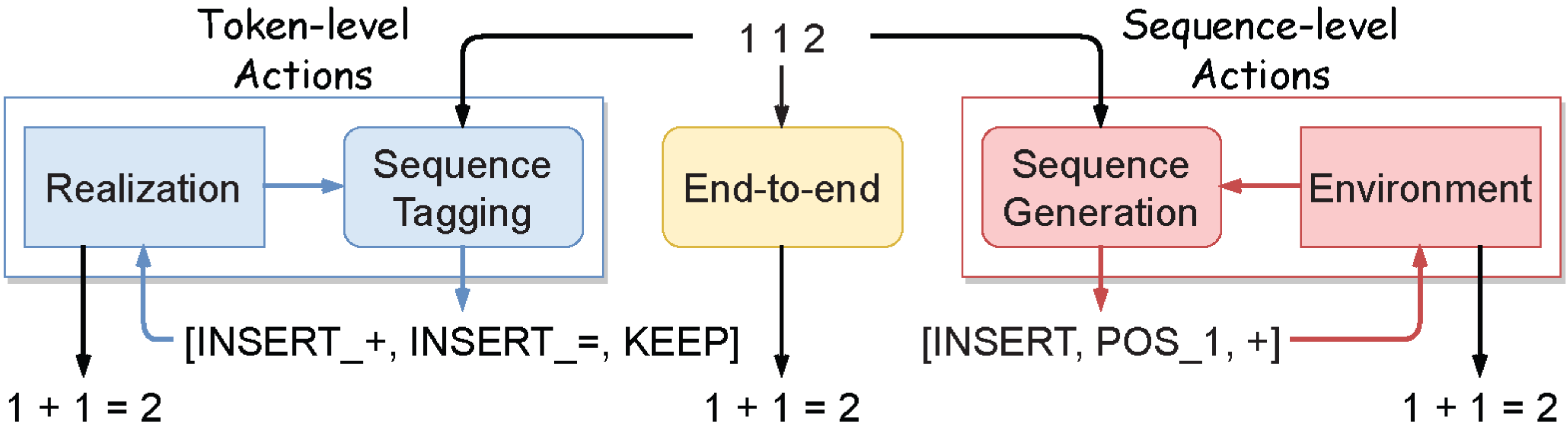
<insert_+> <insert_=> <keep>

Imitation Game

Imitation Learning (IL) & Recurrent Inference (Sequence-level Action Generation)

- Dynamic encoder context matrix
- Complex task decomposed into easier sub-tasks
- Highest degrees of flexibility at sequence-level

Imitation Game



Three approaches – sequence tagging (left), end-to-end (middle), sequence generation (right).

Imitation Game

Markov Decision Process (MDP) Definition

- State S – a set of text sequences

Source text x as initial state s_1 (e.g., 1 1 2)

Target text y as target state s_T (e.g., 1 + 1 = 2)

Every edited texts as intermediate states s_t (e.g., 1 + 1 2)

Thus, the path $X \mapsto Y$ can be a set of sequential states $s_{<T}$

Imitation Game

Markov Decision Process (MDP) Definition

- State S – a set of text sequences
- Action A – a set of action sequences

Edit metric $\boldsymbol{E}$ (e.g., Levenshtein distance)

As long as $X \mapsto Y$ given $A_{\boldsymbol{E}}$

Examples: [INSERT, POS_3, =]

INSERT -> operation token

POS_3 -> position token

Imitation Game

Markov Decision Process (MDP) Definition

- 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}

Due to the nature of text editing, we know it is always 1, meaning always happen.

Imitation Game

Markov Decision Process (MDP) Definition

- 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 $\mathcal{E}$ - to update state by $s_{t+1} = \mathcal{E}(s_t, a_t)$

The game environment is episodic and allows control of the editing process.

Imitation Game

Markov Decision Process (MDP) Definition

- 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 $\mathcal{E}$ - to update state by $s_{t+1} = \mathcal{E}(s_t, a_t)$
- Reward function R – to calculate a reward for each action

In this work, we focus on behavior cloning (BC), so the reward function can be omitted for now.

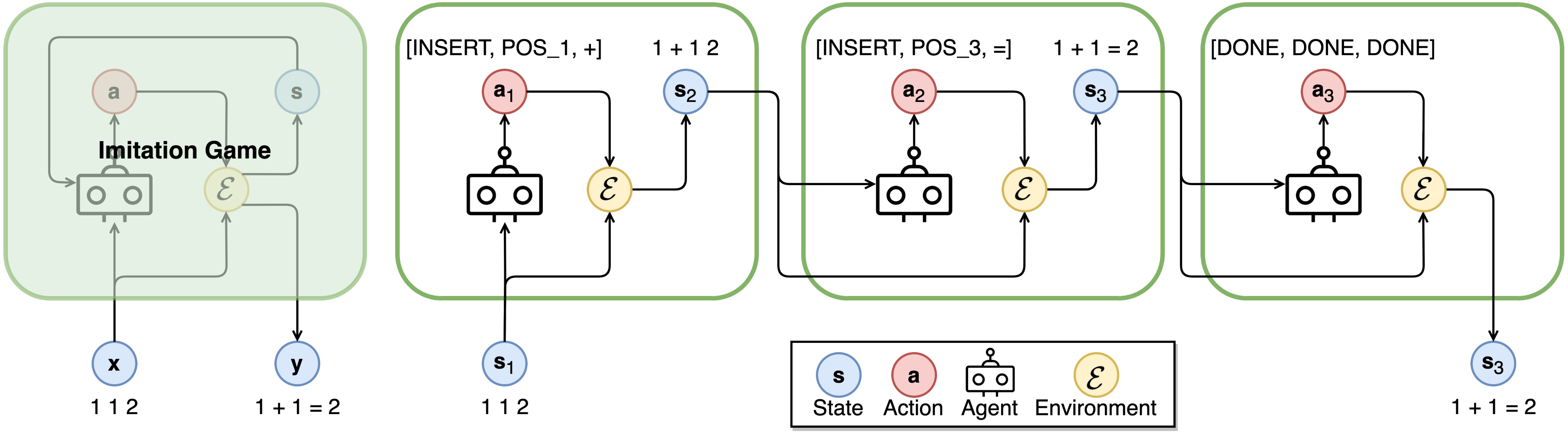
Imitation Game

Markov Decision Process (MDP) Definition

- 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 $\mathcal{E}$ - to update state by $s_{t+1} = \mathcal{E}(s_t, a_t)$
- Reward function R – to calculate a reward for each action

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

Imitation Game



An example of the imitation game to complete "1 1 2" as "1 + 1 = 2".

Imitation Game

Trajectory Generation (TG)

How to convert conventional sequence-to-sequence data into state-to-action demonstrations?

Dynamic programming (DP) to back trace the minimum edit distance given the edit metric.

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.

```
τ ← ∅
s ← x
ops ← DP(x, y, E)
for op ∈ ops do
    a ← Action(op)            ▷ Translate operation to action
    τ ← τ ∪ [(s, a)]
    s ← ℰ(s, a)
end for
τ ← τ ∪ [(s, a_T)]  ▷ Append goal state and output action
return τ
```

Imitation Game

Trajectory Augmentation (TA)

IL suffers from distribution shift and error accumulation.

TA to expand the expert demonstrations and actively expose shifted states utilizing the divide-and-conquer technique.

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**
    $\mathbf{a}_t \leftarrow \mathbf{A}.\text{pop}(0)$
    $\mathbf{s}_{t+1} \leftarrow \mathcal{E}(\mathbf{s}_t, \mathbf{a}_t)$
    $\mathbf{S} \leftarrow \mathbf{S} \cup \text{TA}(\mathbf{S}, \mathbf{s}_{t+1}, \mathbf{S}^*, \mathbf{A}, \mathcal{E})$ $\triangleright$ Execute action
    $\mathbf{A} \leftarrow \text{Update}(\mathbf{A}, \mathbf{s}_t, \mathbf{s}_{t+1})$
    $\mathbf{S} \leftarrow \mathbf{S} \cup \text{TA}(\mathbf{S}, \mathbf{s}_t, \mathbf{S}^*, \mathbf{A}, \mathcal{E})$ $\triangleright$ Skip action
else if $\mathbf{s}_t \notin \mathbf{S}^*$ **then**
    $\mathbf{S} \leftarrow \mathbf{S} \cup [\mathbf{s}_t]$ $\triangleright$ Merge shifted state
end if
return $\mathbf{S}$

Imitation Game

Trajectory Augmentation (TA)

Advantages:

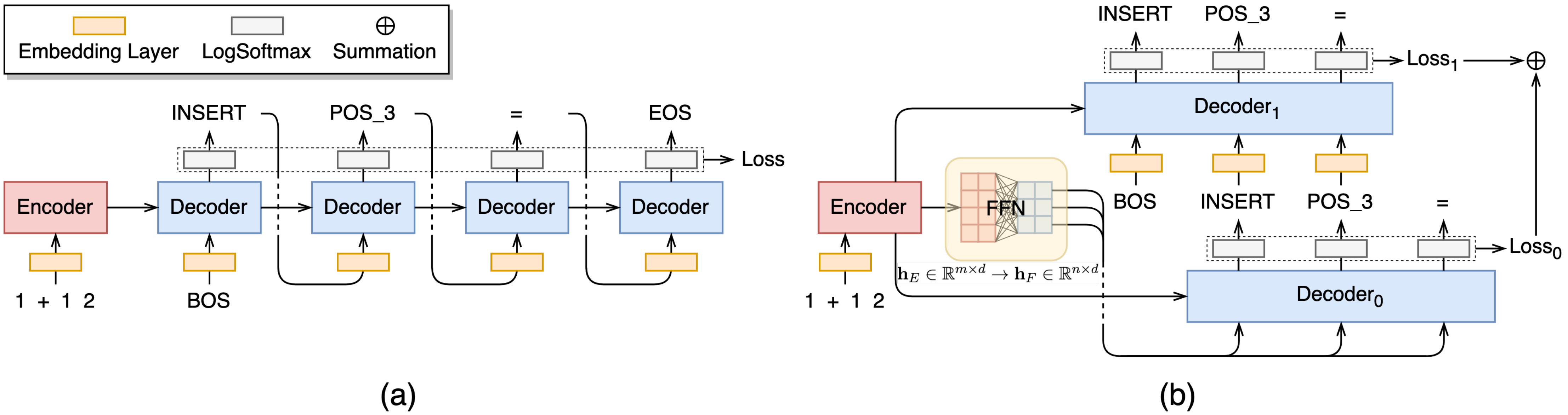
- To preserve the i.i.d. assumption
- No dependency on the task
- No domain knowledge
- No labeling work
- No further evaluation

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**
  $\mathbf{a}_t \leftarrow \mathbf{A}.\text{pop}(0)$
  $\mathbf{s}_{t+1} \leftarrow \mathcal{E}(\mathbf{s}_t, \mathbf{a}_t)$
  $\mathbf{S} \leftarrow \mathbf{S} \cup \text{TA}(\mathbf{S}, \mathbf{s}_{t+1}, \mathbf{S}^*, \mathbf{A}, \mathcal{E})$ ▷ Execute action
  $\mathbf{A} \leftarrow \text{Update}(\mathbf{A}, \mathbf{s}_t, \mathbf{s}_{t+1})$
  $\mathbf{S} \leftarrow \mathbf{S} \cup \text{TA}(\mathbf{S}, \mathbf{s}_t, \mathbf{S}^*, \mathbf{A}, \mathcal{E})$ ▷ 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 $\mathbf{S}$

Non-Autoregressive Decoding



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)

AOR ($N = 10$, $L = 5$, $D = 10$K)			**AES** ($N = 100$, $L = 5$, $D = 10$K)			**AEC** ($N = 10$, $L = 5$, $D = 10$K)		
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

Table 1: Data statistics of AE benchmarks.

Term	**AOR** ($N = 10$, $L = 5$, $D = 10$K)	**AES** ($N = 100$, $L = 5$, $D = 10$K)	**AEC** ($N = 10$, $L = 5$, $D = 10$K)
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

Table 2: Examples from AE with specific N for integer size, L for the number of integers, and D for data size.

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

Models

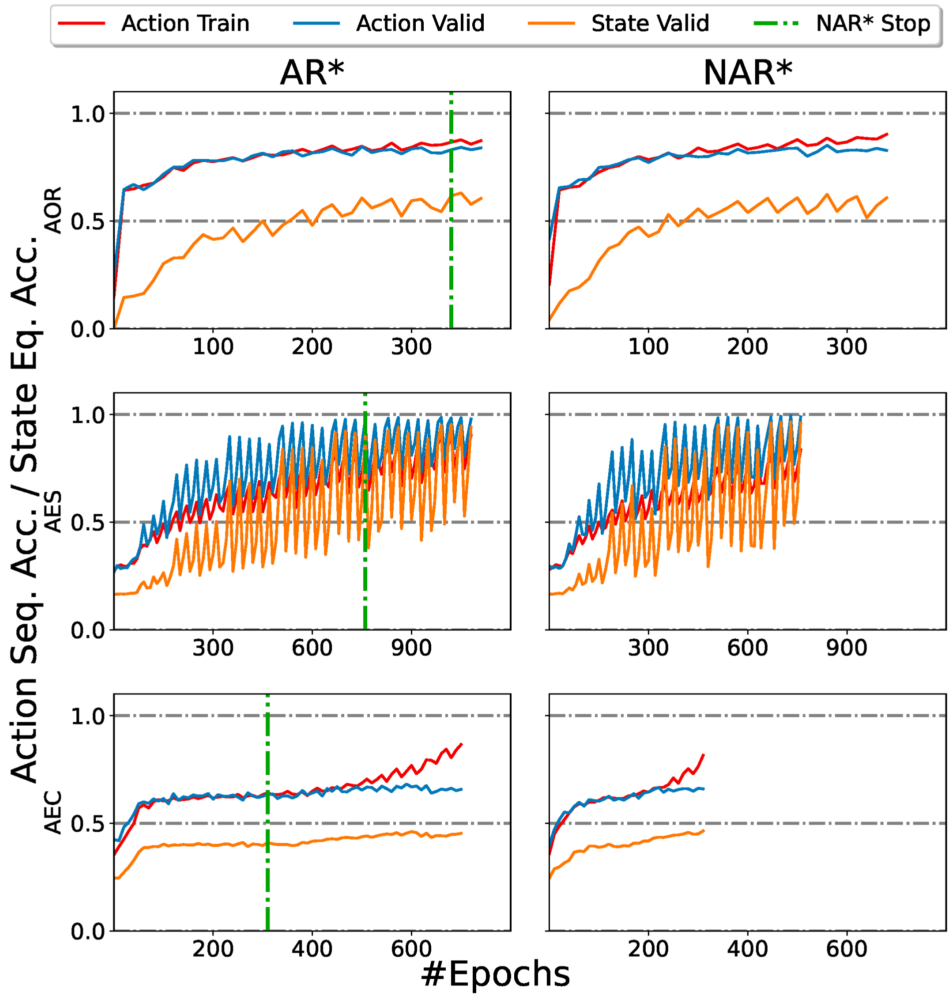
- **End2end** – translate x to y from end to end
- **Tagging** – token level action
- **Recurrence** – recurrent inference via autoregressive LSTM
- **Recurrence*** - rerun the source code of Recurrence that only has access to the fixed training set
- **AR** – our reproduction of Recurrence* in our pipeline
- **AR*** - increase the encoder layers in AR from 1 to 4
- **NAR** – replace autoregressive decoder of AR* with a linear layer to enable non-autoregressive decoding
- **NAR*** - our method with D2 non-autoregressive decoder
- **+TA** – enable trajectory augmentation

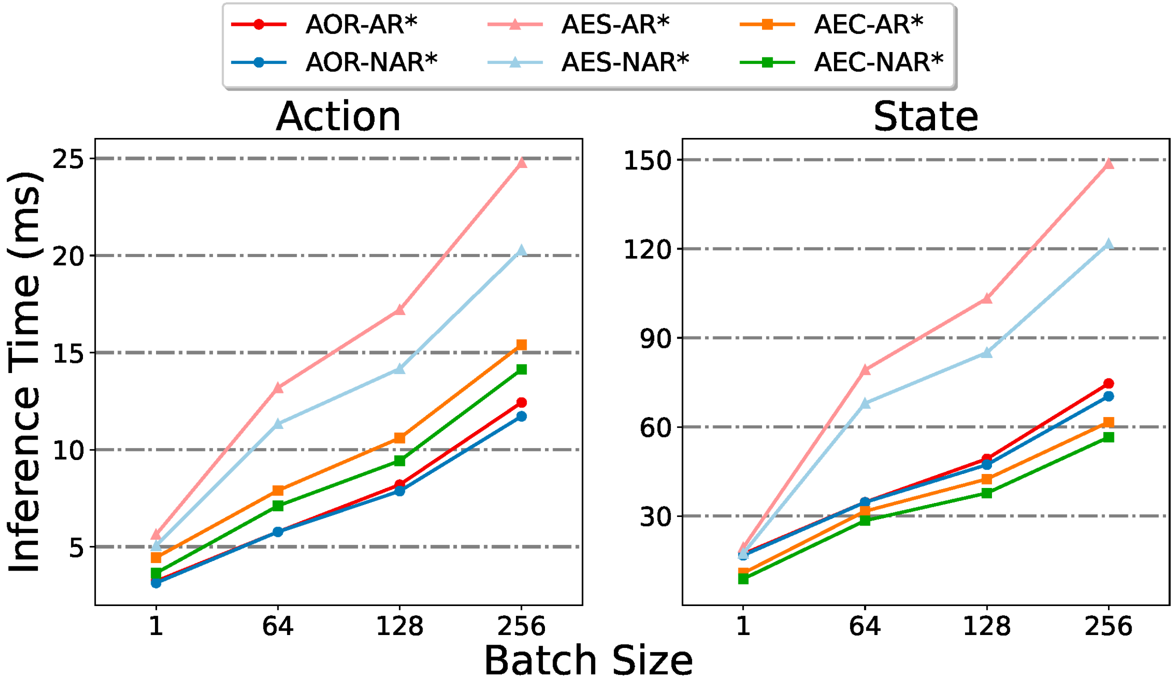
Experimental Results

Method	AOR ($N = 10$, $L = 5$, $D = 10$K)			AES ($N = 100$, $L = 5$, $D = 10$K)		AEC ($N = 10$, $L = 5$, $D = 10$K)		
	Tok. Acc. %	**Seq. Acc. %**	**Eq. Acc. %**	**Tok. Acc. %**	**Eq. Acc. %**	**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	$\mathbf{49.64 \pm 0.59}^*$	$\mathbf{50.15 \pm 0.55}^*$

Table 3: Evaluation results on AOR, AES, and AEC with specific N, L, and D. The token and sequence accuracy for AOR were not reported, thus we leave these positions blank here. With or without TA, our proposed NAR* achieves the best performance in terms of equation accuracy across the board.

Experimental Results





Analysis

Action Design

Due to the liberty of sequence generation, the same operation can be represented as different action sequences by, for example, a simple swap of action tokens.

Our NAR* stays nearly consistent across three designs.

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	$\mathbf{99.61 \pm 0.06}^*$	$\mathbf{96.55 \pm 0.46}^*$

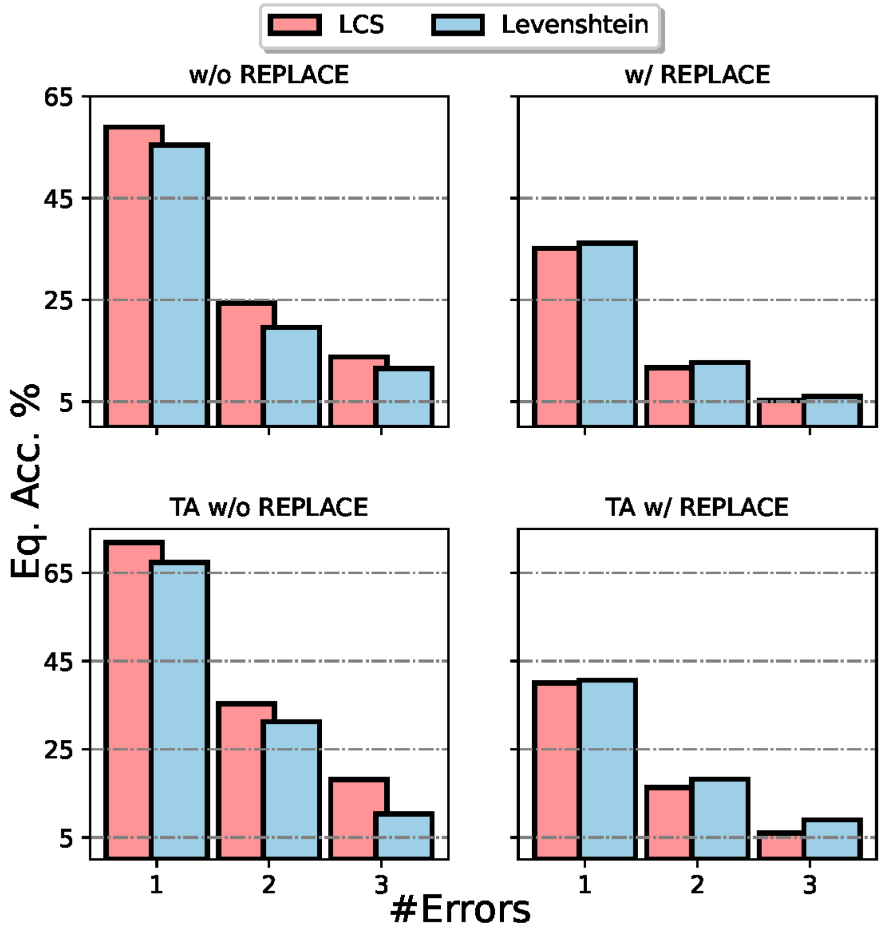
Table 4: Evaluation of AR* and NAR* in AES across three action designs that vary from each other by token order. They directs to the same operation with Pos.$_L$/Pos.$_R$/Tok. denoting left parenthesis/right parenthesis/target token.

Analysis

Trajectory Optimization

A better edit metric $\boldsymbol{E}$ often means a smaller action vocabulary space, shorter trajectory length, and, therefore, an easier IL.

An appropriate edit metric $\boldsymbol{E}$ depends on the specific task.



Analysis

Dual Decoders

As an ablation study, we freeze the encoder of NAR* and vary its decoder to reveal the contributions of each component in D2.

- **Linear** – replace the decoder with a linear layer
- **Decoder$_0$** – remove the second decoder from D2
- **Shared D2** – share the parameters between two decoders in D2
- **D2 (NAR*)** – our method with D2 non-autoregressive decoder
- **+TA** – enable trajectory augmentation

Analysis

Dual Decoders

As an ablation study, we freeze the encoder of NAR* and vary its decoder to reveal the contributions of each component in D2.

Decoder	**AOR** ($N = 10$, $L = 5$, $D = 10$K)			**AES** ($N = 100$, $L = 5$, $D = 10$K)		**AEC** ($N = 10$, $L = 5$, $D = 10$K)		
	Tok. Acc. %	**Seq. Acc. %**	**Eq. Acc. %**	**Tok. Acc. %**	**Eq. Acc. %**	**Tok. Acc. %**	**Seq. Acc. %**	**Eq. Acc. %**
Linear	61.84 ± 0.94	28.55 ± 1.57	57.72 ± 1.55	99.41 ± 0.26	95.01 ± 2.01	81.35 ± 0.92	42.47 ± 1.85	42.81 ± 1.87
Decoder_0	61.78 ± 0.83	28.20 ± 1.57	58.36 ± 1.58	99.24 ± 0.23	93.49 ± 2.03	80.84 ± 0.66	43.97 ± 1.82	44.32 ± 1.82
Shared D2	61.74 ± 0.71	28.68 ± 0.94	58.05 ± 1.01	99.28 ± 0.24	93.85 ± 2.14	81.38 ± 1.04	43.64 ± 2.03	44.09 ± 2.02
D2 (NAR*)	$\mathbf{62.81 \pm 0.89}$	$\mathbf{30.13 \pm 1.31}$	$\mathbf{61.45 \pm 1.61}$	$\mathbf{99.51 \pm 0.13}$	$\mathbf{95.67 \pm 0.93}$	$\mathbf{81.82 \pm 0.68}$	$\mathbf{45.97 \pm 1.07}$	$\mathbf{46.43 \pm 1.10}$
Linear +TA	61.41 ± 0.28	31.75 ± 0.93	63.15 ± 0.96	99.42 ± 0.17	95.08 ± 1.47	81.54 ± 0.66	46.79 ± 2.26	47.33 ± 2.30
Decoder_0 +TA	62.50 ± 1.24	32.48 ± 1.87	64.47 ± 1.88	99.47 ± 0.13	95.33 ± 1.13	82.02 ± 0.40	46.80 ± 2.04	47.32 ± 1.91
Shared D2 +TA	61.64 ± 0.87	31.21 ± 0.34	62.77 ± 0.85	99.53 ± 0.12	95.91 ± 1.25	81.80 ± 0.47	47.23 ± 1.07	47.61 ± 1.14
D2 (NAR*) +TA	$\mathbf{63.48 \pm 0.38}^*$	$\mathbf{34.23 \pm 0.92}^*$	$\mathbf{67.13 \pm 0.99}^*$	$\mathbf{99.58 \pm 0.15}^*$	$\mathbf{96.44 \pm 1.29}^*$	$\mathbf{82.70 \pm 0.42}^*$	$\mathbf{49.64 \pm 0.59}^*$	$\mathbf{50.15 \pm 0.55}^*$

Table 6: Evaluation of agents equipped with same encoders but different decoders on AE benchmarks.

Conclusion

Contributions:

- Frame text editing into an imitation game

This allows the *highest* degree of flexibility to design actions at the sequence-level, which are arguably more *controllable*, *interpretable*, and *similar* to human behavior.

Conclusion

Contributions:

- Frame text editing into an imitation game
- We involve TG to translate standard datasets

Free to translate the conventional input-output data to state-action demonstrations for a friendly IL.

Conclusion

Contributions:

- Frame text editing into an imitation game
- We involve TG to translate standard datasets
- We introduce D2 as a novel non-autoregressive decoder

To boost the learning in terms of *accuracy*, *efficiency*, and *robustness*

Conclusion

Contributions:

- Frame text editing into an imitation game
- We involve TG to translate standard datasets
- We introduce D2 as a novel non-autoregressive decoder
- We propose TA technique

To mitigate the distribution shift problem IL often suffers

Conclusion

Contributions:

- Frame text editing into an imitation game
- We involve TG to translate standard datasets
- We introduce D2 as a novel non-autoregressive decoder
- We propose TA technique

Future work:

- Reward function, action design, trajectory optimization

Conclusion

Limitations

- Efficiency issue due to multiple calls of encoder (e.g., a heavy pretrained language model)
- Application in more realistic editing tasks (e.g., text simplification)

TLDR

Turning tasks into games that agents feel more comfortable with sheds light on future studies in the direction of reinforcement learning in the application of text editing.

Thanks

arXiv

GitHub