

A Learning & Inference Algorithm

We present the detailed algorithms for learning and decoding from Levenshtein Transformer as follows. For simplicity, we always omit the source information x in conditional sequence generation tasks such as machine translation which is handled by the cross-attention with an encoder on x .

The learning algorithm is shown in Algorithm 1. \mathcal{E} is the environment and \mathcal{D} is denoted as the Levenshtein distance, and we can easily back-track the optimal insertion and deletion operations through dynamic programming. We only show the case with single batch-size for convenience. We also present the inference algorithm in Algorithm 2. If the initial sequence y^0 is empty ($\langle s \rangle \langle /s \rangle$), the proposed model will skip the first deletion and do sequence generation. Otherwise, the model starts with deletion operations and refine the input sequence.

Algorithm 1 Learning for Levenshtein Transformer

Initialize: Training set \mathcal{T} , expert policy π^* , model policy π_θ , random deletion policy π^{RND} , α , β
repeat
 Sample a training pair $(y^0, y^*) \sim \mathcal{Y}$
 if expert π^* is a teacher model **then**
 Set the teacher’s output as the target $y^* = y^{\text{AR}}$
 end if
 Sample $u, v \sim \text{Uniform}[0, 1]$
 if $u < \beta$ **then**
 $y_{\text{ins}} = \mathcal{E}(y^0, \tilde{d})$, where $\tilde{d} = \arg\min_d \mathcal{D}(y^*, \mathcal{E}(y^0, d))$
 else
 $y_{\text{ins}} = \mathcal{E}(y^*, \tilde{d})$, where $\tilde{d} \sim \pi^{\text{RND}}(\cdot | y^*)$
 end if
 $y'_{\text{ins}} = \mathcal{E}(y_{\text{ins}}, p^*)$, where $p^*, t^* = \arg\min_{p, t} \mathcal{D}(y^*, \mathcal{E}(y_{\text{ins}}, \{p, t\}))$
 if $v < \alpha$ **then**
 $y_{\text{del}} = y^0$
 else
 $y_{\text{del}} = \mathcal{E}(y'_{\text{ins}}, \hat{t})$, where $\hat{t} = \arg\max_t \sum_{y_i \in y'_{\text{ins}}, y_i = \langle \text{PLH} \rangle} \log \pi_\theta^{\text{tok}}(t_i | i, y'_{\text{ins}})$
 end if
 $\mathcal{L}_\theta^{\text{ins}} = - \left[\sum_{y_i \in y_{\text{ins}}, p_i^* \in p^*} \log \pi_\theta^{\text{plh}}(p_i^* | i, y_{\text{ins}}) + \sum_{y_i \in y'_{\text{ins}}, y_i = \langle \text{PLH} \rangle, t_i^* \in t^*} \log \pi_\theta^{\text{tok}}(t_i^* | i, y'_{\text{ins}}) \right]$
 $\mathcal{L}_\theta^{\text{del}} = - \sum_{y_i \in y_{\text{del}}, d_i^* \in d^*} \log \pi_\theta^{\text{del}}(d_i^* | i, y_{\text{del}})$, where $d^* = \arg\min_d \mathcal{D}(y^*, \mathcal{E}(y_{\text{del}}, d))$
 $\theta = \theta - \lambda \cdot \nabla_\theta [\mathcal{L}_\theta^{\text{ins}} + \mathcal{L}_\theta^{\text{del}}]$
until Maximum training steps reached

B Dataset and Preprocessing Details

Table 4 and 5 list the statistics (# of sentences, vocabulary) for all the datasets used in this work. We learn BPE vocabulary with 32,000 joint operations for WMT En-De and Gigaword and 40,000 joint operations for WMT Ro-En. For WAT En-Ja, we adopt the official 16,384 BPE vocabularies learned separately on source and target side.

Table 4: Dataset statistics for sequence generation tasks (MT and TS).

	Dataset	Train	Valid	Test	Vocabulary
Translation	WMT’16 Ro-En	608,319	1999	1999	34,983
	WMT’14 En-De	4,500,966	3000	3003	37,009
	WAT’17 En-Ja	2,000,000	1790	1812	17,952 / 17,801
Summarization	English Gigaword	3,803,957	189,651	1951	30,004

Algorithm 2 Decoding for Levenshtein Transformer

Initialize: Input $\mathbf{y} = \mathbf{y}^0$, step $t = 0$, maximum step T_{\max} , model policy π_θ .
repeat
 if $\mathbf{y} = \langle \text{s} \rangle \langle / \text{s} \rangle$ **then**
 Empty sequence, skip deletion: $\mathbf{y}' = \mathbf{y}$
 else
 Delete tokens: $\mathbf{y}' = \mathcal{E}(\mathbf{y}, \hat{\mathbf{d}})$, where $\hat{\mathbf{d}} = \arg\max_{\mathbf{d}} \sum_{y_i \in \mathbf{y}} \log \pi_\theta^{\text{del}}(d_i | i, \mathbf{y})$
 end if
 if $(t > 0) \ \& \ (\mathbf{y}' = \tilde{\mathbf{y}})$ **then**
 Termination condition satisfied: direct loop
 break
 end if
 Assign deleted output for back-up $\tilde{\mathbf{y}} = \mathbf{y}'$
 Insert placeholders: $\mathbf{y}'' = \mathcal{E}(\mathbf{y}', \hat{\mathbf{p}})$, where $\hat{\mathbf{p}} = \arg\max_{\mathbf{p}} \sum_{y_i y_{i+1} \in \mathbf{y}'} \log \pi_\theta^{\text{plh}}(p_i | i, \mathbf{y}')$
 if $\mathbf{y}'' = \mathbf{y}' = \mathbf{y}$ **then**
 Termination condition satisfied: nothing to delete, nothing to insert.
 break
 end if
 if $\mathbf{y}'' = \mathbf{y}'$ **then**
 Nothing to insert, skip insertion: $\mathbf{y} = \mathbf{y}''$
 else
 Replace placeholders: $\mathbf{y} = \mathcal{E}(\mathbf{y}'', \hat{\mathbf{t}})$, where $\hat{\mathbf{t}} = \arg\max_{\mathbf{t}} \sum_{y_i \in \mathbf{y}'', y_i = \langle \text{PLH} \rangle} \log \pi_\theta^{\text{tok}}(t_i | i, \mathbf{y}'')$
 end if
 Update steps: $t = t + 1$
until Reach the maximum length $t = T_{\max}$
return \mathbf{y}

Table 5: Dataset statistics for sequence refinement tasks (APE).

Dataset		MT-Train	APE-Train	Valid	Test	Vocabulary
Synthetic	WMT'16 Ro-En	300,000	308,319	1999	1999	34,983
	WMT'14 En-De	2,250,000	2,250,967	3000	3003	37,009
	WAT'17 En-Ja	1,000,000	1,000,000	1790	1812	17,952 / 17,801
Real	WMT'17 APE En-De	4,391,180	526,368 (fake) + 24,000 (real)	2000	2000	40,349

C Model and Training Details

C.1 Sequence Generation Tasks

Transformer models are used for autoregressive baselines as well as teacher models (for the expert policy). By default, we set $d_{\text{model}} = 512$, $d_{\text{hidden}} = 2048$, $n_{\text{heads}} = 8$, $n_{\text{layers}} = 6$, $\text{lr}_{\max} = 0.0005$, $\text{label-smooth} = 0.1$, $\text{warmup} = 10000$ and $\text{dropout} = 0.3$. Source and target side share embeddings in all the training pairs except for WAT En-Ja where BPE vocabularies of both side are learned separately and are almost non-overlapping.

Since the training objectives for Levenshtein Transformer contains randomness terms (Eq. (6) (7)), we instead use BLEU (for MT) or ROUGE-2 (for TS) to select the best checkpoint by validation scores. We do not average checkpoints in this work.

C.2 Sequence Refinement Tasks

For synthetic APE tasks, we keep the same training conditions for LevT as those for MT tasks (§C.1). As described earlier in §4.2, we build the baseline Transformer by concatenating the source and MT system's output as the input sequence for the encoder. Specially, we restart the positional embeddings

Table 6: The percentage of WMT En-De test sentence generation terminated at each iteration using LevT(T) with a maximum iteration of 10.

Iterations	1	2	3	4	5	6	7	8	9	10	2.43
%	12.3	48.1	28.5	8.5	2.0	0.4	0.1	0	0	0.1	AVG

for the MT output, add an additional language embedding for each token of the input sequence to show its language type. The detailed hyperparameters are the same as the standard Transformer.

As described in §4.2, we consider the following two different imperfect MT systems to provide the refinement inputs. Firstly, we consider the traditional statistical phrase-based machine translation system (PBMT). We follow the instruction to build the basic baseline model via moses⁹. As for the NMT-based model, we use a single layer attention-based model composed by LSTM. We build this model on fairseq-py¹⁰ with the default configuration.

For the real APE task, we follow the procedures introduced in Junczys-Dowmunt and Grundkiewicz (2016). Synthetic corpus has two subsets: a 500K one and a 4M one. We over-sample real data by 10 times and merge it with the 500K synthetic data to train APE models. Besides, we also train a LevT MT model on the bigger (4M) synthetic corpus where we only use the source and target pairs.

C.3 Implementation

Both the proposed Levenshtein Transformer and the baseline Transformer are implemented using PyTorch¹¹. The codes are released as part of the Fairseq-py¹².

C.4 Maximum Number of Iterations

We also presented in general how many sentences will be generated using the maximum iteration (for instance 10). As shown in Table 6, surprisingly, most predictions are gotten in 1-4 iterations, and the average number of iterations is 2.43. Only a tiny portion ($\sim 0.1\%$) require the maximum number of iterations demonstrating the efficiency of the proposed approach.

⁹<http://www.statmt.org/moses/?n=Moses.Baseline>

¹⁰<https://github.com/pytorch/fairseq/blob/master/fairseq/models/lstm.py>

¹¹<https://pytorch.org/>

¹²https://github.com/pytorch/fairseq/tree/master/examples/nonautoregressive_translation

D More Decoding Examples

We present more examples from the proposed Levenshtein Transformer as follows.

(a)	__The __too __high __rotation __speed __produces __the __reverse __deformation .	→	しかし、回転速度が大きすぎると、逆向きの変形が生じる。
(iteration 1)	nothing to delete >> insert >>		[[回轉回轉]]すぎ[[ると]]逆[[変形]]が生じる[[。]]
(iteration 2)	delete >> insert >>		[[回轉回轉]]すぎ[[ると]]逆[[変形]]が生じる[[。]]
(iteration 3)	nothing to delete >> insert >>		[[回轉]]速度が[[すぎ]]ると[[、]]逆[[変形]]が生じる[[。]]
	nothing to delete, nothing to insert >>		[[回轉]]速度が[[高]]すぎ[[ると]]、逆[[変形]]が生じる[[。]]
			[Terminate]
(b)	__Some __possible __structures __and __circuits __were __proposed __and __verified .	→	いくつかの可能な構造と回路を提案し検証した。
(iteration 1)	nothing to delete >> insert >>		[[可能な]]構造[[回路]]を提案し[[、]]検証した[[。]]
(iteration 2)	delete >> insert >>		[[可能な]] 構造 [[回路]]を提案し[[、]]検証した[[。]]
	nothing to delete, nothing to insert >>		[[いくつかの]]可能な[[構造]]と[[回路]]を提案し[[、]]検証した[[。]]
			[Terminate]

Figure 6: Translation examples for WAT’ 17 Small-NMT En-Ja with the Levenshtein Transformer.

(a)	primii oameni observa abia dupa cateva minute .	→	the first people noticed after a few minutes .
(iteration 1)	nothing to delete >> insert >>		the first people see see a few minutes later .
(iteration 2)	delete >> insert >>		the first people see see a few minutes later .
	nothing to delete, nothing to insert >>		the first people can see only a few minutes later .
			[Terminate]
(b)	cinema city cauta " micile voci mari ";	→	cinema city seeks " small big voices ";
(iteration 1)	nothing to delete >> insert >>		cinema city seeks " small voices voices ";
(iteration 2)	delete >> insert >>		cinema city seeks " small voices voices ";
	nothing to delete, nothing to insert >>		cinema city seeks " the small big voices . ";
			[Terminate]
(c)	vietile lor aveau sa se des@@@ parta definitiv .	→	their lives would forever part ways .
(iteration 1)	nothing to delete >> insert >>		their lives were to a permanently apart .
(iteration 2)	delete >> insert >>		their lives were to to a permanently apart .
	nothing to delete, nothing to insert >>		their lives were to break apart permanently .
			[Terminate]

Figure 7: Translation examples for WMT’ 16 Ro-En with the Levenshtein Transformer.

(a) Or search for planets similar to the Earth and thus perhaps discover extraterrestrial life ?	Oder suchen Sie nach Planeten ähnlich der Erde und entdecken Sie damit vielleicht das extraterre- stische Leben ?
<i>nothing to delete >></i>	
(iteration 1) <i>insert >></i>	Oder suchen Sie Planeten Planeten Planeten Erde Erde Erde Erde und vielleicht Extraterre- stische entdecken ?
(iteration 2) <i>delete >></i>	Oder suchen Sie Planeten Planeten Planeten Erde Erde Erde Erde und vielleicht Extraterre- stische entdecken ?
(iteration 2) <i>insert >></i>	Oder suchen Sie nach Planeten ähnlich der Erde und entdecken so vielleicht außerirdische Leben ?
(iteration 3) <i>delete >></i>	Oder suchen Sie nach Planeten ähnlich der Erde und entdecken so vielleicht außerirdische Leben ?
(iteration 3) <i>insert >></i>	Oder suchen Sie nach Planeten ähnlich der Erde und entdecken so vielleicht das Leben ?
<i>nothing to delete >></i>	Oder suchen Sie nach Planeten ähnlich der Erde und entdecken so vielleicht das Leben ?
(iteration 4) <i>insert >></i>	Oder suchen Sie nach Planeten ähnlich der Erde und entdecken so vielleicht das eigene Leben ?
<i>nothing to delete, nothing to insert >></i>	[Terminate]
(b) Local public transport will also become more expensive .	Der öffentliche Nahverkehr werde auch teurer .
<i>nothing to delete >></i>	
(iteration 1) <i>insert >></i>	Auch der öffentliche Nahverkehr werden teurer .
(iteration 2) <i>delete >></i>	Auch der öffentliche Nahverkehr werden teurer .
(iteration 2) <i>insert >></i>	Auch der öffentliche Nahverkehr wird teurer .
<i>nothing to delete, nothing to insert >></i>	[Terminate]

Figure 8: Translation examples for WMT'14 En-De with the Levenshtein Transformer.

(a) strengthened border patrol has led to a ##-percent drop in arrests of undocumented migrants this year at the u.s.-mexico frontier , it was reported on wednesday .	arrests of border-crossers drop
<i>nothing to delete >></i>	
(iteration 1) <i>insert >></i>	border patrol leads arrests ## migrants at u.s.-mexico border
(iteration 2) <i>delete >></i>	border patrol leads arrests ## migrants at u.s.-mexico border
(iteration 2) <i>insert >></i>	border patrol reduces arrests of migrants at u.s.-mexico border
<i>nothing to delete, nothing to insert >></i>	[Terminate]
(b) us lawyer ed fagan said wednesday he will bring a multi-million dollar lawsuit in the united states against the polish government unless it takes concrete steps to repay a huge debt to holders of bonds issued before world war ii .	us star lawyer ed fagan to sue poland for unpaid bonds
<i>nothing to delete >></i>	
(iteration 1) <i>insert >></i>	fagan to multi-million sue poland poland over debt debt
(iteration 2) <i>delete >></i>	fagan to multi-million sue poland poland over debt debt
(iteration 2) <i>insert >></i>	fagan threatens to sue poland over debt repayment
<i>nothing to delete, nothing to insert >></i>	[Terminate]

Figure 9: Translation examples for English Gigaword with the Levenshtein Transformer.

(a)	In the tag , insert the ActionScript code to create the behavior .	→	Fügen Sie im Tag den ActionScript-@@ Codes ein , um das Verhalten zu erstellen .
(iteration 1)	<i>delete >></i>		Klicken Sie im Tag des ActionScript-@@ Codes einfügen , um das Verhalten zu erstellen .
	<i>insert >></i>		Fügen Sie im Tag des ActionScript-@@ Codes ein , um das Verhalten zu erstellen .
	<i>nothing to delete, nothing to insert >></i>		[Terminate]
(b)	In the tag , insert the ActionScript code to create the behavior .	→	Verwenden Sie die Schaltfläche " Bearbeiten , " um eine neue Java@@ Script@@ Aktion zu ändern oder zu erstellen .
(iteration 1)	<i>delete >></i>		Verwenden Sie die Schaltfläche " Bearbeiten , " um eine neue JavaScript Aktion zu ändern oder erstellen .
	<i>insert >></i>		Verwenden Sie die Schaltfläche " Bearbeiten , " um eine neue JavaScript-@@ Aktion zu ändern oder zu erstellen .
	<i>nothing to delete, nothing to insert >></i>		[Terminate]
(c)	To resize the canvas , drag the frame corners .	→	Um die Größe der Leinwand zu verändern , ziehen Sie die Rahmen@@ e@@ cken .
(iteration 1)	<i>delete >></i>		Um die Größe der Leinwand , ziehen Sie den Rahmen .
	<i>insert >></i>		Um die Größe der Leinwand zu ändern , ziehen Sie die Rahmen@@ e@@ cken .
	<i>nothing to delete, nothing to insert >></i>		[Terminate]

Figure 10: Post-editing examples for WMT'17-APE En-De with the Levenshtein Transformer.

administratia t@@ v@@ r a facut constant eforturi pentru diminuarea cheltuielilor cu personalul si productia tv .	→	the administration of t@@ v@@ r has made constant efforts to reduce personnel and tv production expenses .
MT		
(iteration 1)	<i>nothing to delete >></i>	
	<i>insert >></i>	the t@@ v@@ r administration has constantly constantly efforts to cut spending on personnel and tv production .
(iteration 2)	<i>delete >></i>	the t@@ v@@ r administration has constantly constantly efforts to cut spending on personnel and tv production .
	<i>insert >></i>	the t@@ v@@ r administration has constantly made efforts to cut spending on personnel and tv production .
	<i>nothing to delete, nothing to insert >></i>	[Terminate]
APE (zero-shot on PBMT)		
(iteration 1)	<i>delete >></i>	the t@@ v@@ r made constant efforts to reduce expenditure on staff and tv production .
	<i>insert >></i>	the t@@ v@@ r administration administration has making constant efforts to reduce expenditure on staff and tv production .
(iteration 2)	<i>delete >></i>	the t@@ v@@ r administration administration has making constant efforts to reduce expenditure on staff and tv production .
	<i>insert >></i>	the t@@ v@@ r administration has made constant efforts to reduce expenditure on staff and tv production .
	<i>nothing to delete, nothing to insert >></i>	[Terminate]

Figure 11: An example for machine translation and zero-shot post-editing over a PBMT system's output on WMT'16 Ro-En with the Levenshtein Transformer (LevT) trained for MT. It is clear to find that, the pre-trained LevT can directly adapt to the PBMT's output and have a different refinement results compared to translate from scratch.