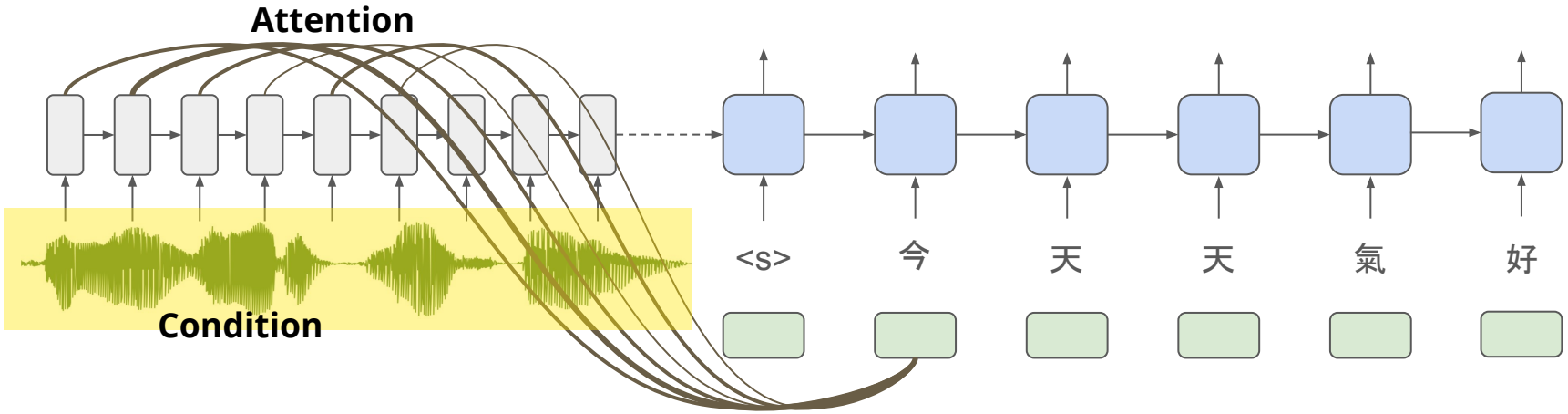
Non-Autoregressive Sequence Generation

Yung-Sung Chuang 莊永松
https://voidism.github.io/home/

2020/05/13

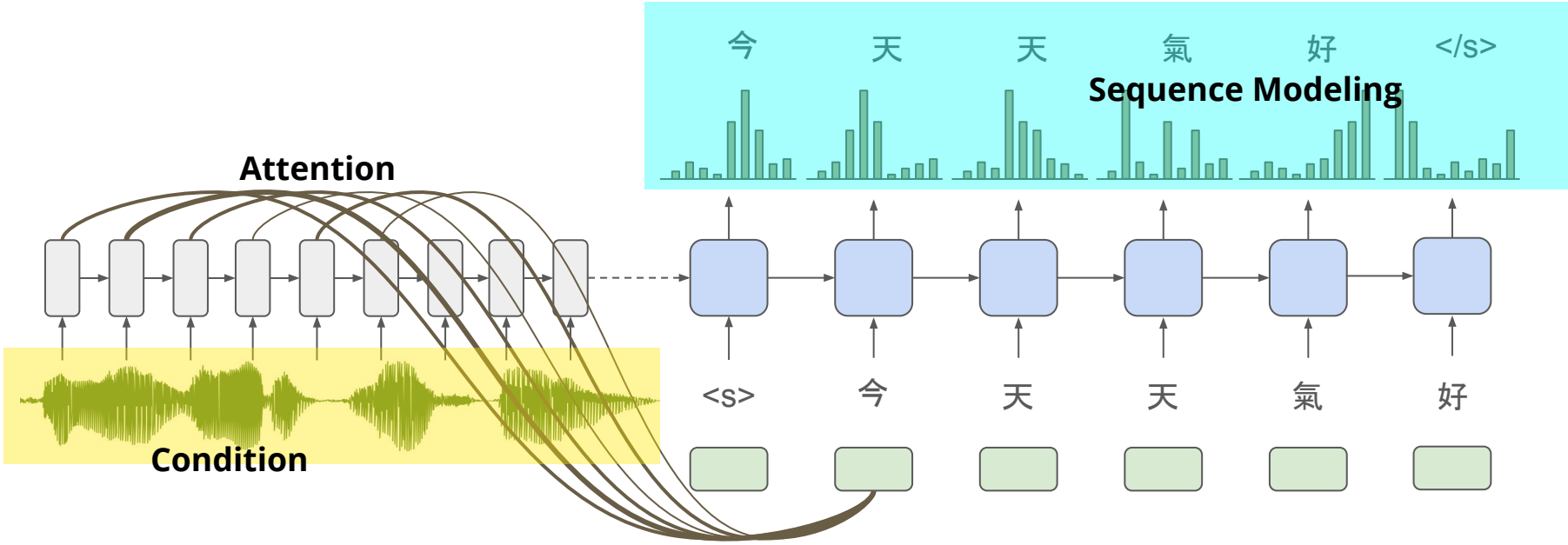
What We have done in HW1...



=> Conditional Sequence Generation

What We have done in HW1...

$$p(Y|X) = \prod_{t=1}^{T} p(y_t \mid y_{<t}, X)$$



=> Conditional Sequence Generation

More Conditional Sequence Generation

A woman is throwing a frisbee in a park.

Image Caption Generation

Hello, World!

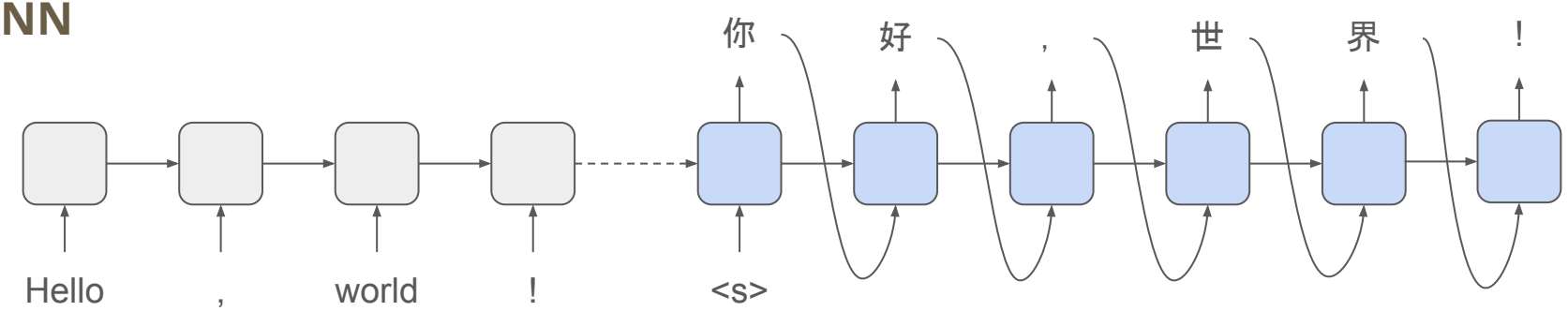
你好，世界！

Machine Translation

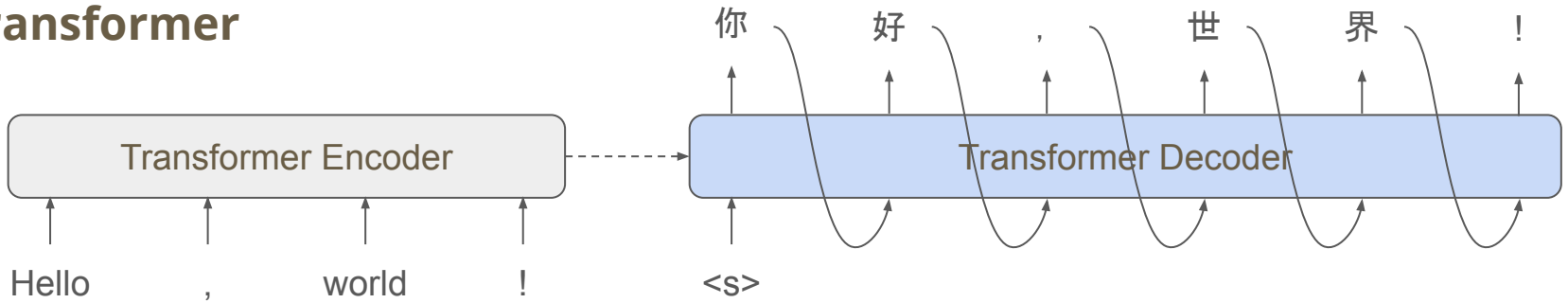
Autoregressive model (inference stage)

$$p(Y|X) = \prod_{t=1}^{T} p(y_t \mid y_{<t}, X)$$

RNN



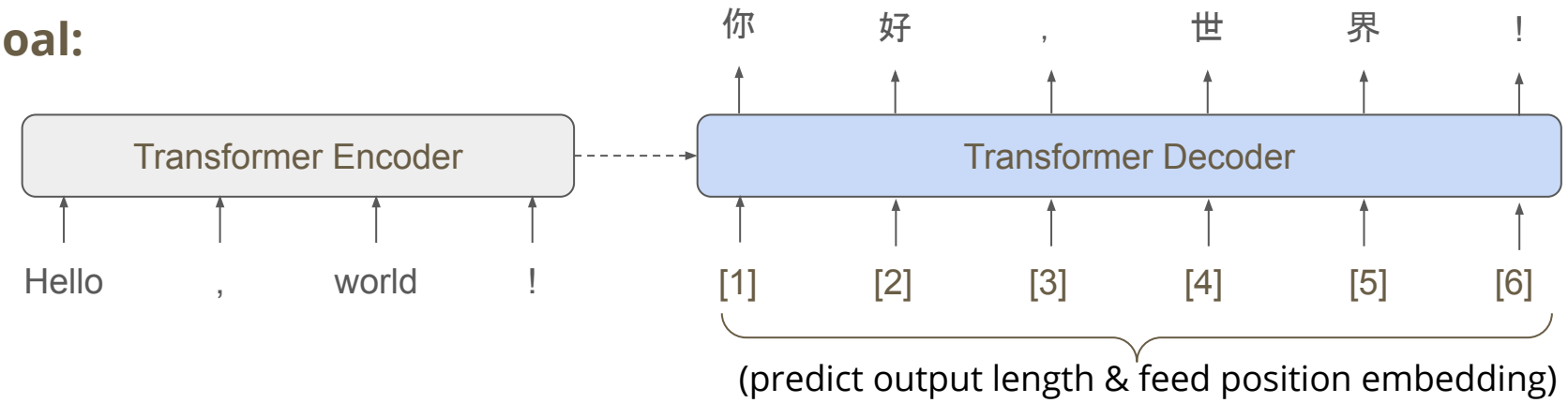
Transformer



Non-autoregressive model (mostly by Transformer)

$$p(Y|X) = \prod_{t=1}^{T} p(y_t \mid X)$$

Goal:

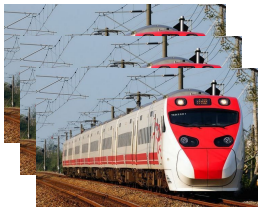


What's the problem?

http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2018/Lecture/CGAN.pdf

Output would be average of images

70%

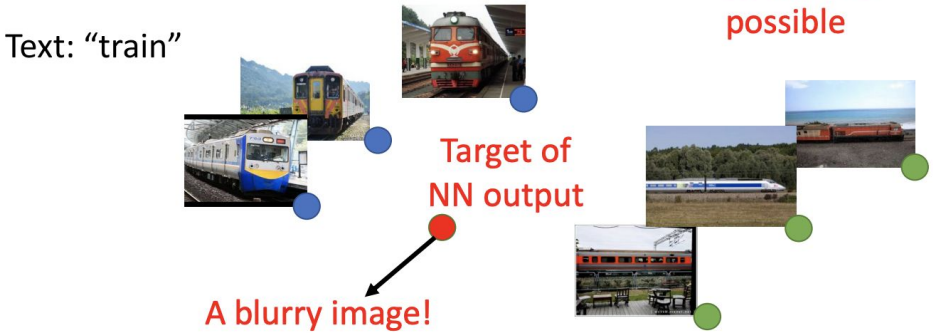


30%

model learned

Text-to-Image

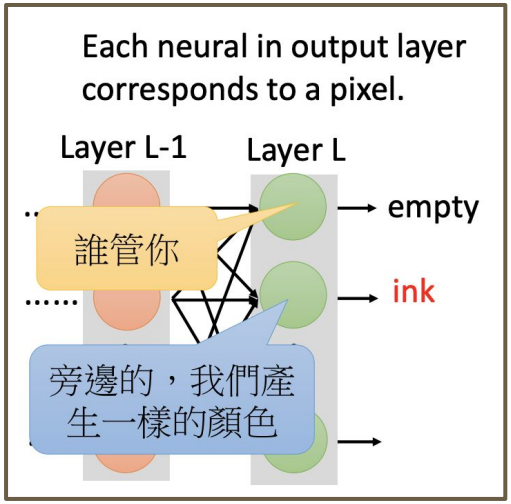
- **Traditional supervised approach**



What's the problem?

https://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2018/Lecture/GAN%20(v2).pdf

No dependency on output structure

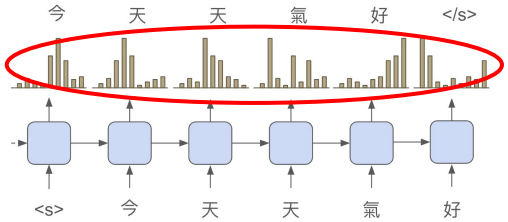


No Latent Variable

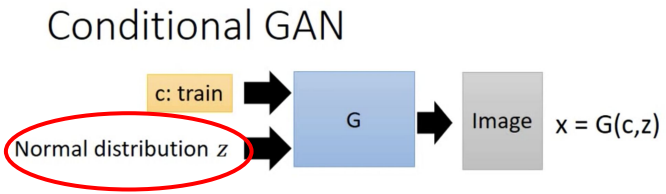
70% 30%

Solutions:

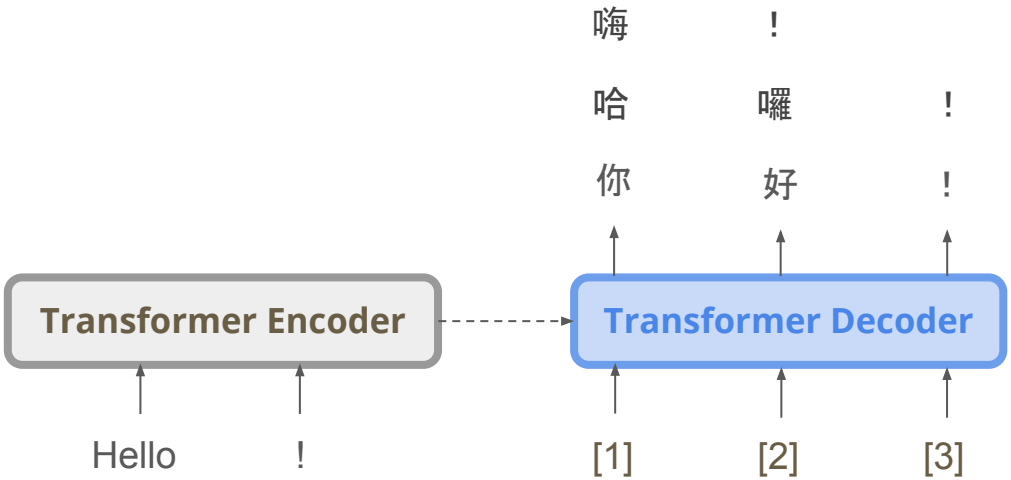
Autoregressive model
ex: PixelRNN

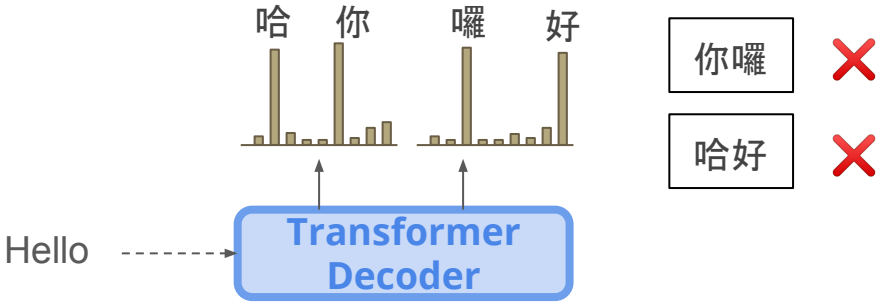


GAN

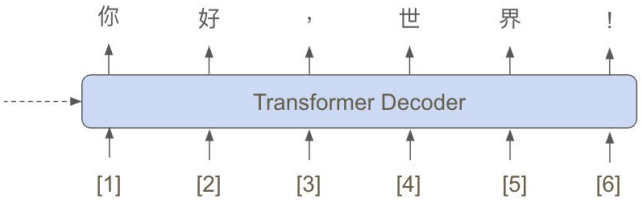
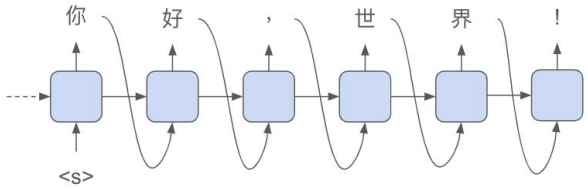


What's the problem?





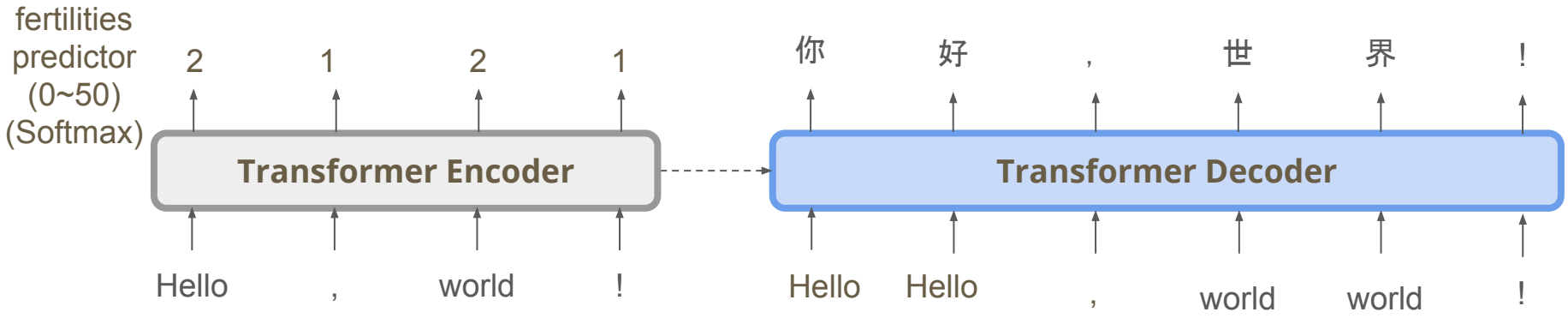
=> multi-modality problem!

generate target / methods	**Image**	**Text**
naive approach	Deconvolution layer + L2 loss  Average of images (bad)	non-autoregressive decoder 
autoregressive	ex: PixelRNN, VQVAE-2 	autoregressive decoder 
Generative Adversarial Networks w/ non-auto model		

Vanilla NAT (Non-Autoregressive Translation)

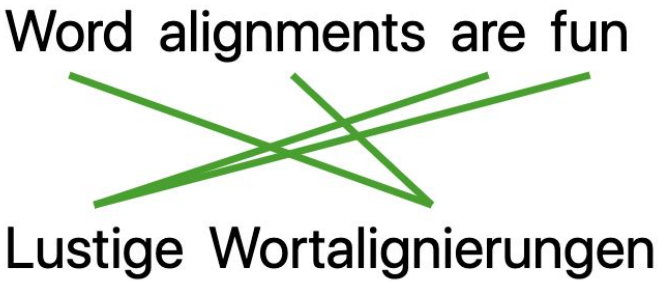
[1711.02281] Non-Autoregressive Neural Machine Translation
Gu et al. ICLR 2018.

- Predict **fertility** as latent variable & Copy input words
- Represents sentence-level “plan” before writing Y



External Aligner: https://github.com/clab/fast_align

Fertility

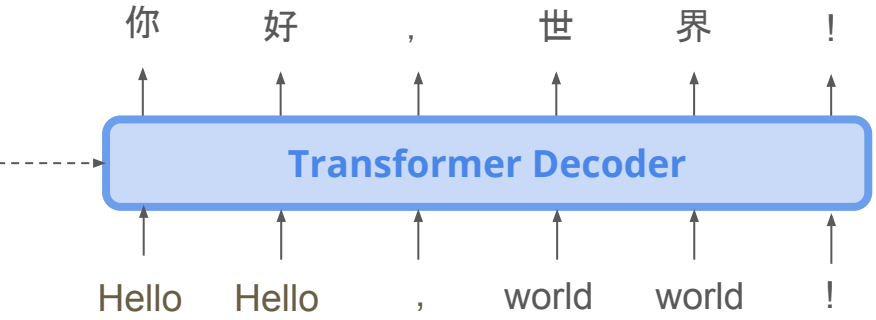


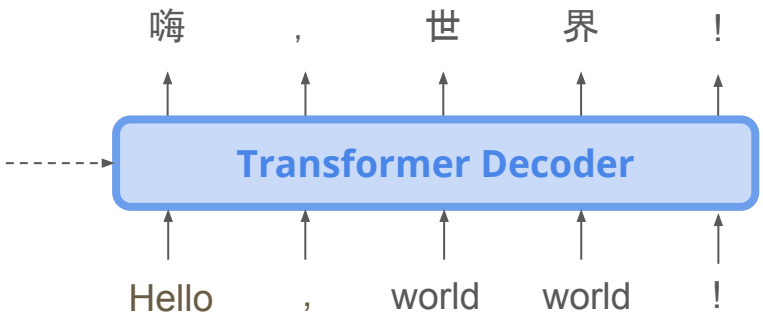
Fixed targets

1. Labels comes from external aligner
2. Observing attention weights in auto-regressive models

Fine-tune *(after NAT model converges)*

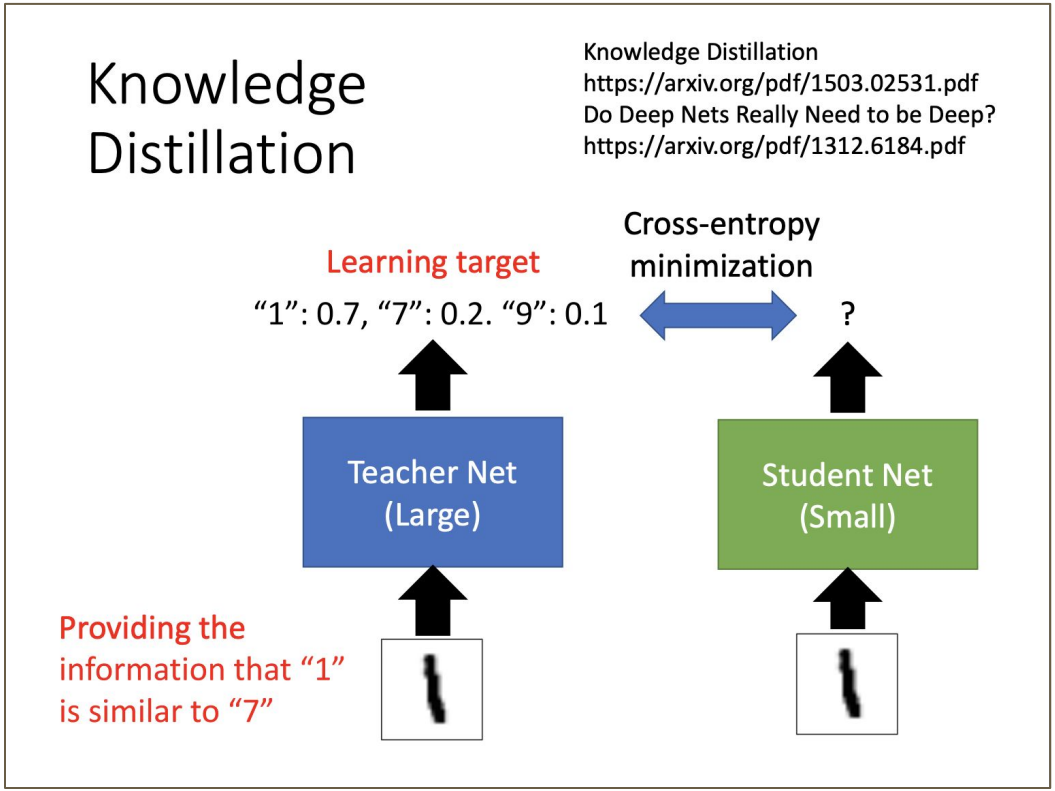
1. Updating fertility classifier with REINFORCE





http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2019/Lecture/Small%20(v6).pdf

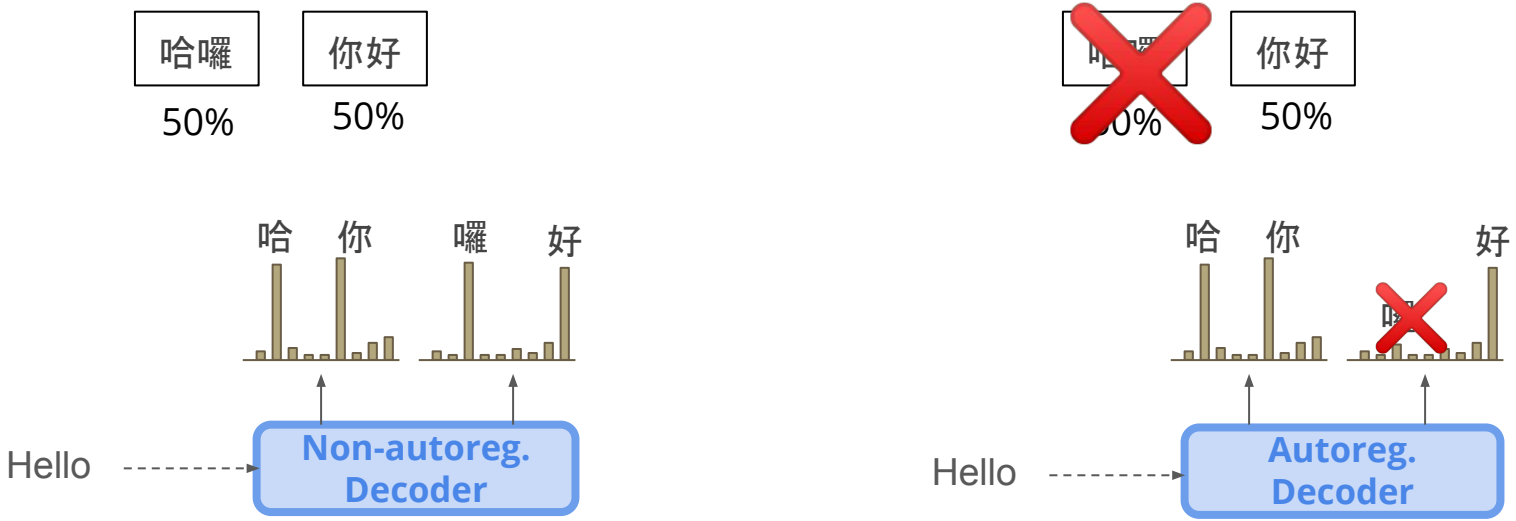
Sequence-level knowledge distillation



http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2019/Lecture/Small%20(v6).pdf

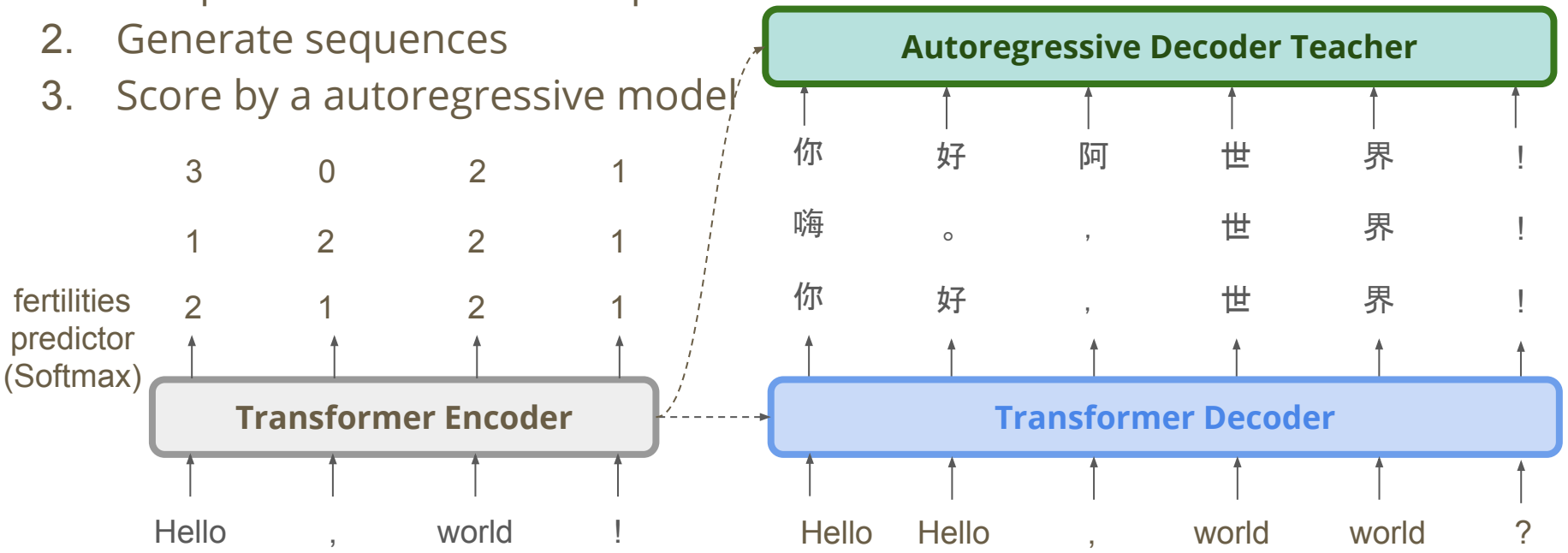
Sequence-level knowledge distillation

- Teacher: Autoregressive model, Student: Non-autoregressive model
- Construct new corpus by autoregressive teacher model
- Teacher's greedy decode output as student's training target

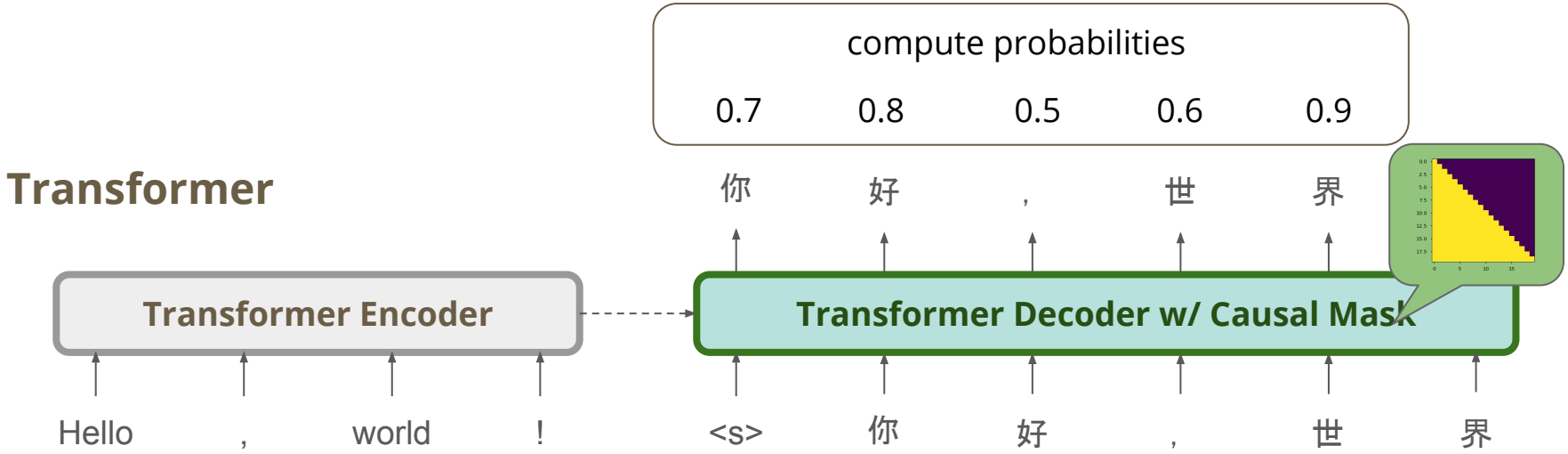


Noisy Parallel Decoding (NPD)

1. Sample several fertilitie sequences
2. Generate sequences
3. Score by a autoregressive model



Autoregressive model w/ teacher forcing



Experiments

Models	WMT14		WMT16		IWSLT16		
	En→De	De→En	En→Ro	Ro→En	En→De	Latency	Speedup
NAT	17.35	20.62	26.22	27.83	25.20	39 ms	15.6×
NAT (+FT)	17.69	21.47	27.29	29.06	26.52	39 ms	15.6×
NAT (+FT + NPD $s = 10$)	18.66	22.41	29.02	30.76	27.44	79 ms	7.68×
NAT (+FT + NPD $s = 100$)	19.17	23.20	29.79	**31.44**	28.16	257 ms	2.36×
Autoregressive ($b = 1$)	22.71	26.39	31.35	31.03	28.89	408 ms	1.49×
Autoregressive ($b = 4$)	23.45	27.02	31.91	31.76	29.70	607 ms	1.00×

Experiments

Models	WMT14		WMT16		IWSLT16		
	En→De	De→En	En→Ro	Ro→En	En→De	Latency	Speedup
NAT	17.35	20.62	26.22	27.83	25.20	39 ms	15.6×
NAT (+FT)	17.69	21.47	27.29	29.06	26.52	39 ms	15.6×
NAT (+FT + NPD $s = 10$)	18.66	22.41	29.02	30.76	27.44	79 ms	7.68×
NAT (+FT + NPD $s = 100$)	19.17	23.20	29.79	**31.44**	28.16	257 ms	2.36×
Autoregressive ($b = 1$)	22.71	26.39	31.35	31.03	28.89	408 ms	1.49×
Autoregressive ($b = 4$)	23.45	27.02	31.91	31.76	29.70	607 ms	1.00×

Experiments

Models	WMT14		WMT16		IWSLT16		
	En→De	De→En	En→Ro	Ro→En	En→De	Latency	Speedup
NAT	17.35	20.62	26.22	27.83	25.20	39 ms	15.6×
NAT (+FT)	17.69	21.47	27.29	29.06	26.52	39 ms	15.6×
NAT (+FT + NPD $s = 10$)	18.66	22.41	29.02	30.76	27.44	79 ms	7.68×
NAT (+FT + NPD $s = 100$)	19.17	23.20	29.79	**31.44**	28.16	257 ms	2.36×
Autoregressive ($b = 1$)	22.71	26.39	31.35	31.03	28.89	408 ms	1.49×
Autoregressive ($b = 4$)	23.45	27.02	31.91	31.76	29.70	607 ms	1.00×

Experiments

Models	WMT14		WMT16		IWSLT16		
	En→De	De→En	En→Ro	Ro→En	En→De	Latency	Speedup
NAT	17.35	20.62	26.22	27.83	25.20	39 ms	15.6×
NAT (+FT)	17.69	21.47	27.29	29.06	26.52	39 ms	15.6×
NAT (+FT + NPD $s = 10$)	18.66	22.41	29.02	30.76	27.44	79 ms	7.68×
NAT (+FT + NPD $s = 100$)	19.17	23.20	29.79	**31.44**	28.16	257 ms	2.36×
Autoregressive ($b = 1$)	22.71	26.39	31.35	31.03	28.89	408 ms	1.49×
Autoregressive ($b = 4$)	23.45	27.02	31.91	31.76	29.70	607 ms	1.00×

Experiments

Models	WMT14		WMT16		IWSLT16		
	En→De	De→En	En→Ro	Ro→En	En→De	Latency	Speedup
NAT	17.35	20.62	26.22	27.83	25.20	39 ms	15.6×
NAT (+FT)	17.69	21.47	27.29	29.06	26.52	39 ms	15.6×
NAT (+FT + NPD $s = 10$)	18.66	22.41	29.02	30.76	27.44	79 ms	7.68×
NAT (+FT + NPD $s = 100$)	19.17	23.20	29.79	**31.44**	28.16	257 ms	2.36×
Autoregressive ($b = 1$)	22.71	26.39	31.35	31.03	28.89	408 ms	1.49×
Autoregressive ($b = 4$)	23.45	27.02	31.91	31.76	29.70	607 ms	1.00×

uniform: 看對應長度，四捨五入直接 Copy inputs

Ablation

Distillation b=1	Distillation b=4	Decoder Inputs +uniform	Decoder Inputs +fertility	Decoder Inputs +PosAtt	Fine-tuning $+\mathcal{L}_{KD}$	Fine-tuning $+\mathcal{L}_{BP}$	Fine-tuning $+\mathcal{L}_{RL}$	BLEU	BLEU (T)
				✓				≈ 2	
		✓		✓				16.51	
			✓	✓				18.87	
✓		✓		✓				20.72	
	✓	✓		✓				21.12	
✓			✓					24.02	43.91
✓			✓	✓				25.20	45.41
✓		✓		✓	✓	✓		22.44	
✓			✓	✓			✓	×	×
✓			✓	✓		✓		×	×
✓			✓	✓	✓	✓		25.76	46.11
✓			✓	✓	✓	✓	✓	**26.52**	**47.38**

uniform: 看對應長度，四捨五入直接 Copy inputs

Ablation

Distillation b=1	Distillation b=4	Decoder Inputs +uniform	Decoder Inputs +fertility	+PosAtt	Fine-tuning +$\mathcal{L}_{KD}$	Fine-tuning +$\mathcal{L}_{BP}$	Fine-tuning +$\mathcal{L}_{RL}$	BLEU	BLEU (T)
				✓				≈ 2	
		✓		✓				16.51	
			✓	✓				18.87	
✓		✓		✓				20.72	
	✓	✓		✓				21.12	
✓			✓					24.02	43.91
✓			✓	✓				25.20	45.41
✓		✓		✓	✓	✓		22.44	
✓			✓	✓			✓	×	×
✓			✓	✓		✓		×	×
✓			✓	✓	✓	✓		25.76	46.11
✓			✓	✓	✓	✓	✓	**26.52**	**47.38**

uniform: 看對應長度，四捨五入直接 Copy inputs

Ablation

Distillation		Decoder Inputs			Fine-tuning			BLEU	BLEU (T)
b=1	b=4	+uniform	+fertility	+PosAtt	+$\mathcal{L}_{\text{KD}}$	+$\mathcal{L}_{\text{BP}}$	+$\mathcal{L}_{\text{RL}}$		
				✓				≈ 2	
		✓		✓				16.51	
			✓	✓				18.87	
✓		✓		✓				20.72	
	✓	✓		✓				21.12	
✓			✓					24.02	43.91
✓			✓	✓				25.20	45.41
✓		✓		✓	✓	✓		22.44	
✓			✓	✓			✓	×	×
✓			✓	✓		✓		×	×
✓			✓	✓	✓	✓		25.76	46.11
✓			✓	✓	✓	✓	✓	**26.52**	**47.38**

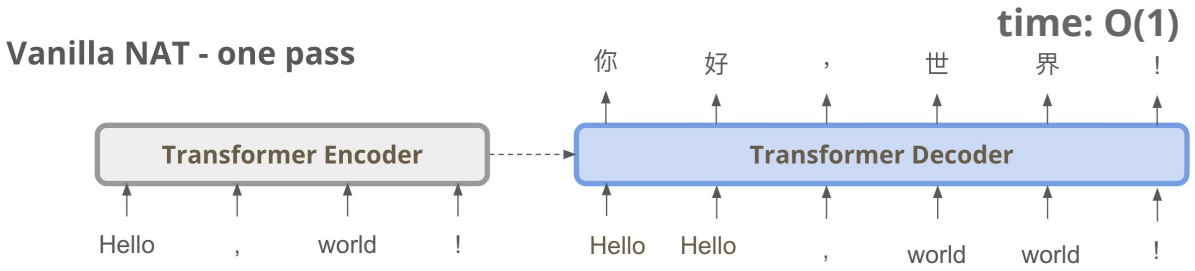
uniform: 看對應長度，四捨五入直接 Copy inputs

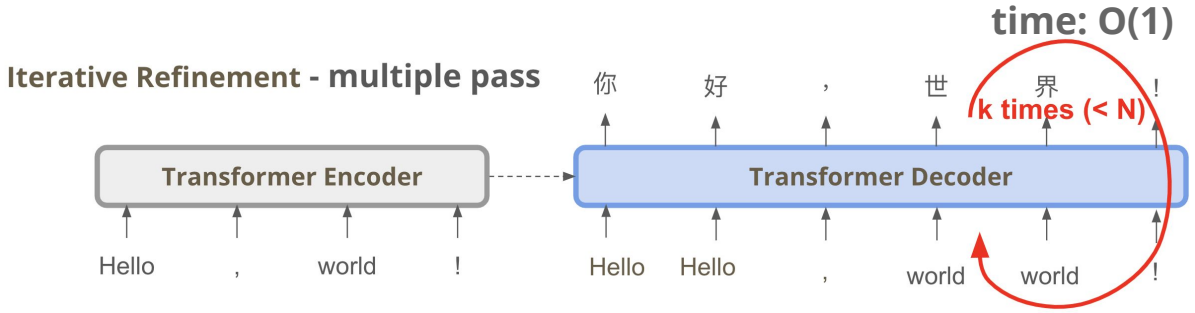
Ablation

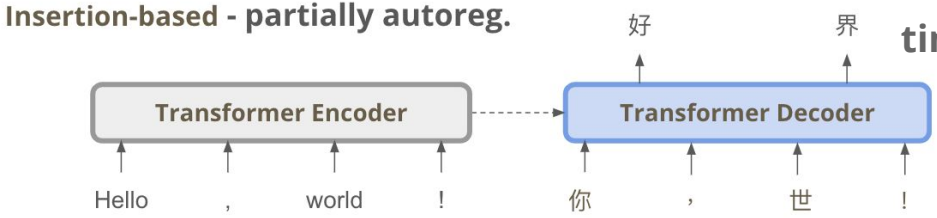
Distillation b=1	Distillation b=4	Decoder Inputs +uniform	Decoder Inputs +fertility	Decoder Inputs +PosAtt	Fine-tuning $+\mathcal{L}_{KD}$	Fine-tuning $+\mathcal{L}_{BP}$	Fine-tuning $+\mathcal{L}_{RL}$	BLEU	BLEU (T)
				✓				≈ 2	
		✓		✓				16.51	
			✓	✓				18.87	
✓		✓		✓				20.72	
	✓	✓		✓				21.12	
✓			✓					24.02	43.91
✓			✓	✓				25.20	45.41
✓		✓		✓	✓	✓		22.44	
✓			✓	✓			✓	×	×
✓			✓	✓		✓		×	×
✓			✓	✓	✓	✓		25.76	46.11
✓			✓	✓	✓	✓	✓	**26.52**	**47.38**

Evolution of NAT

1. **Vanilla NAT**
   a. Gu et al., ICLR'18
2. **Iterative Refinement**
   a. iNAT, Lee et al., EMNLP'18
   b. Mask-Predict, Ghazvininejad et al., EMNLP'19
3. **Insertion-based**
   a. Insertion Transformer, Stern et al., ICML'19
   b. KERMIT, Chan et al., arXiv'19
4. **Insert+Delete**
   a. Levenshtein Transformer, Gu et al., NeurIPS'19
5. **CTC-based**
   a. E2E NAT w/ CTC, Jindrich Libovick et al., EMNLP'20
   b. NAT w/ Latent Alignments, Saharia et al., arXiv'20





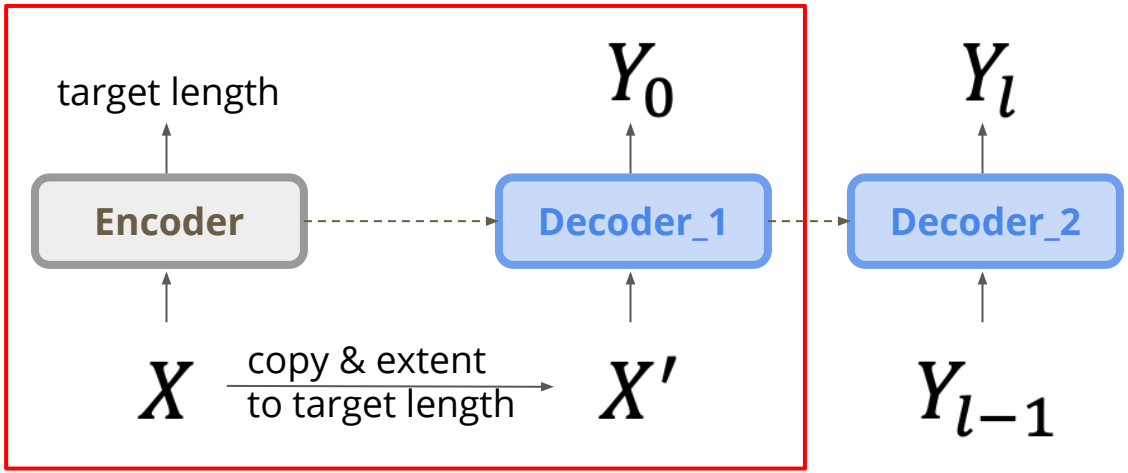


$$[] \rightarrow [D] \rightarrow [B, D, F] \rightarrow [A, B, C, D, E, F, G]$$

[1802.06901] Deterministic Non-Autoregressive Neural Sequence Modeling by Iterative Refinement, Lee et al., EMNLP 2018

NAT with Iterative Refinement

$$X \rightarrow Y_0 \rightarrow Y_1 \rightarrow Y_2 \rightarrow \cdots \rightarrow Y_L = Y$$



Corruption Process

(1) replace y_{t+1} with y_t
(2) replace y_t with a random token
(3) swap y_t and y_{t+1}.

Stochastically mix cost function

$$J(\theta) = -\sum_{l=0}^{L+1}\left(\alpha_l \sum_{t=1}^{T} \log p_\theta(y_t^*|\hat{Y}^{l-1}, X) + (1-\alpha_l)\sum_{t=1}^{T}\log p_\theta(y_t^*|\tilde{Y}, X)\right), \quad (4)$$

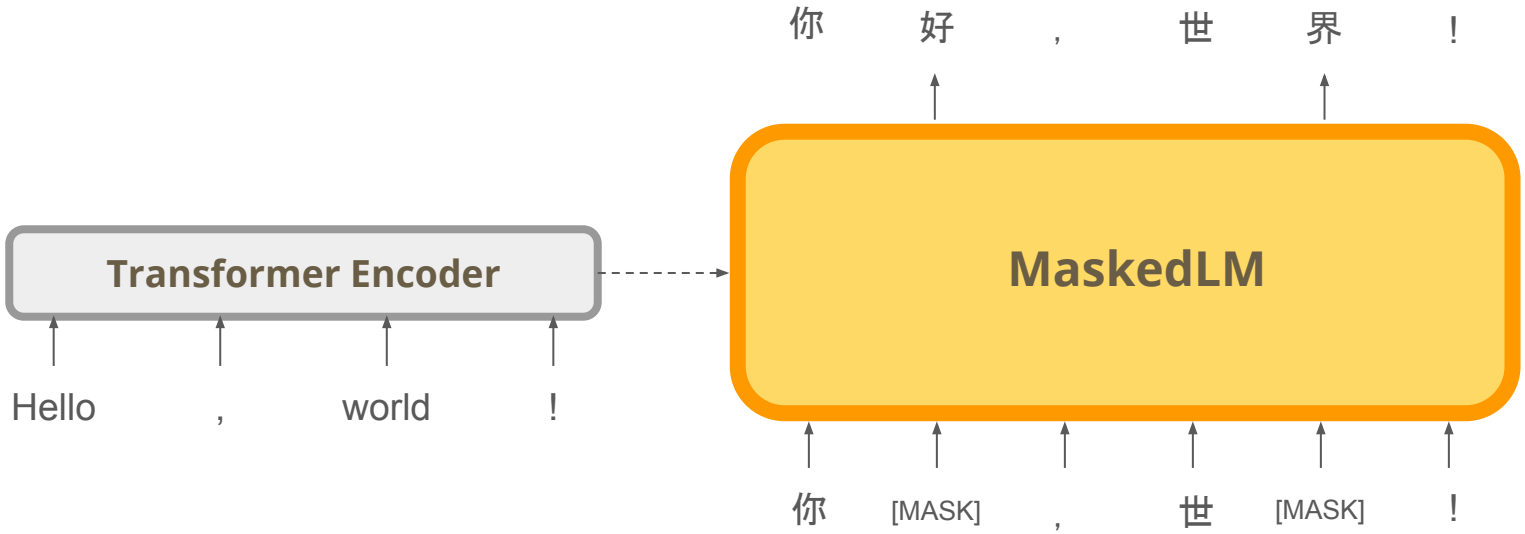
Results

		IWSLT'16 En-De				WMT'16 En-Ro				WMT'14 En-De				MS COCO		
		En→	De→	GPU	CPU	En→	Ro→	GPU	CPU	En→	De→	GPU	CPU	BLEU	GPU	CPU
AR	$b = 1$	28.64	34.11	70.3	32.2	31.93	31.55	55.6	15.7	23.77	28.15	54.0	15.8	23.47	4.3	2.1
	$b = 4$	28.98	34.81	63.8	14.6	32.40	32.06	43.3	7.3	24.57	28.47	44.9	7.0	24.78	3.6	1.0
NAT	FT	26.52	–	–	–	27.29	29.06	–	–	17.69	21.47	–	–	–	–	–
	FT+NPD	28.16	–	–	–	29.79	31.44	–	–	19.17	23.30	–	–	–	–	–
Our Model	$i_{\text{dec}} = 1$	22.20	27.68	573.0	213.2	24.45	25.73	694.2	98.6	13.91	16.77	511.4	83.3	20.12	17.1	8.9
	$i_{\text{dec}} = 2$	24.82	30.23	423.8	110.9	27.10	28.15	332.7	62.8	16.95	20.39	393.6	49.6	20.88	12.0	5.7
	$i_{\text{dec}} = 5$	26.58	31.85	189.7	52.8	28.86	29.72	194.4	29.0	20.26	23.86	139.7	23.1	21.12	6.2	2.8
	$i_{\text{dec}} = 10$	27.11	32.31	98.8	24.1	29.32	30.19	93.1	14.8	21.61	25.48	90.4	12.3	21.24	2.0	1.2
	Adaptive	27.01	32.43	125.9	29.3	29.66	30.30	118.3	16.5	21.54	25.43	107.2	20.3	21.12	10.8	4.8

Mask-Predict

[1904.09324] Mask-Predict: Parallel Decoding of Conditional Masked Language Models, Ghazvininejad et al., EMNLP 2019

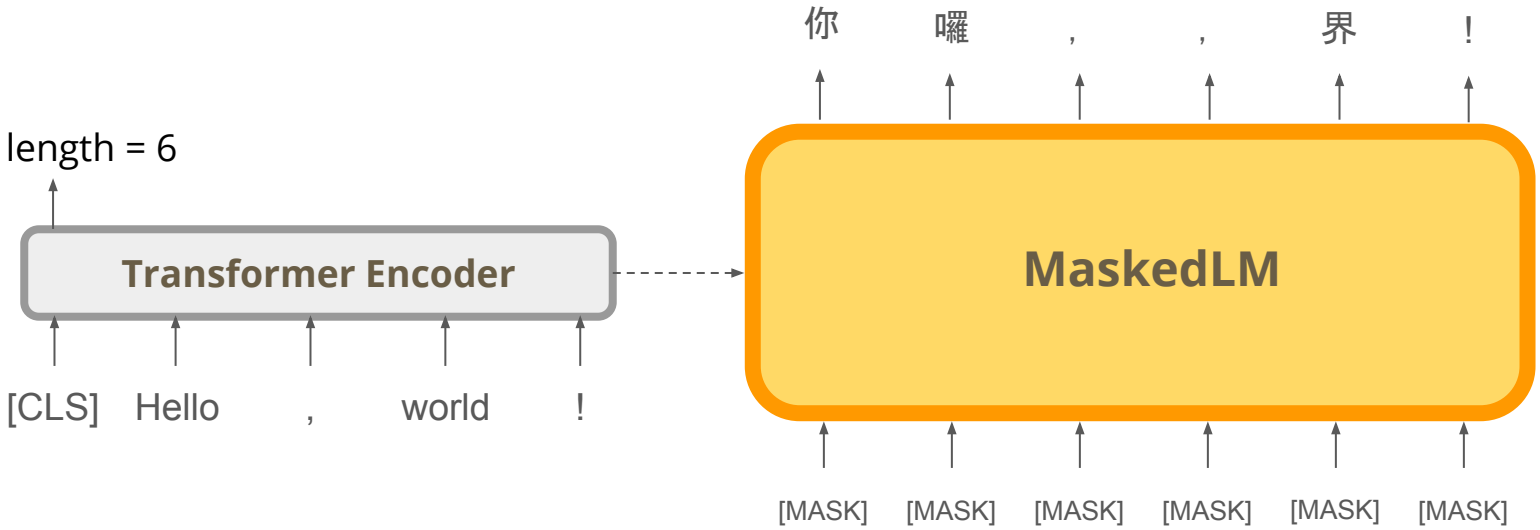
Conditional Masked Language Models (CMLM):



Mask-Predict

[1904.09324] Mask-Predict: Parallel Decoding of Conditional Masked Language Models, Ghazvininejad et al., EMNLP 2019

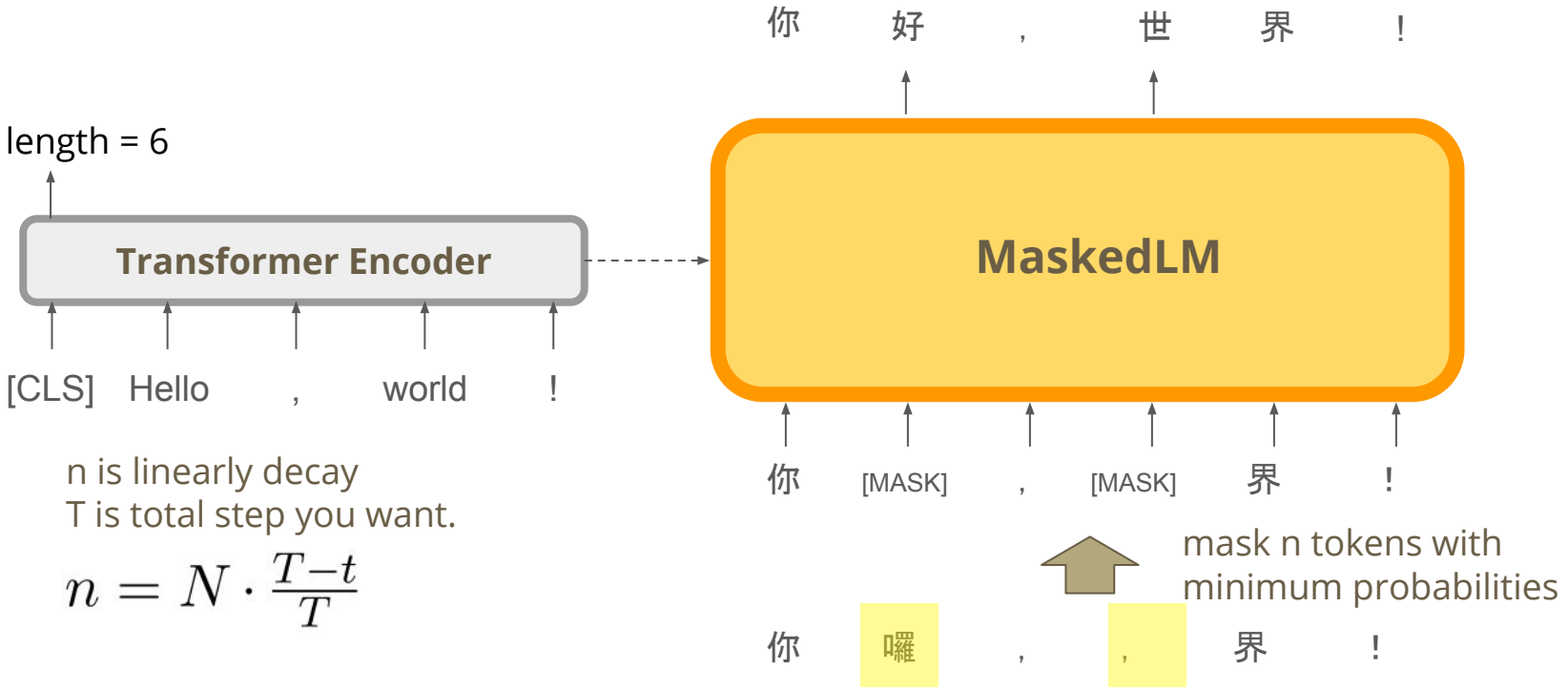
t = 0



Mask-Predict

[1904.09324] Mask-Predict: Parallel Decoding of Conditional Masked Language Models, Ghazvininejad et al., EMNLP 2019

t = 1



n is linearly decay
T is total step you want.

$$n = N \cdot \frac{T-t}{T}$$

Mask-Predict

Example: (target length is fixed)

src	Der Abzug der franzsischen Kampftruppen wurde am 20. November abgeschlossen .
$t = 0$	The departure of the French combat completed completed on 20 November .
$t = 1$	The departure of French combat troops was completed on 20 November .
$t = 2$	The withdrawal of French combat troops was completed on November 20th .

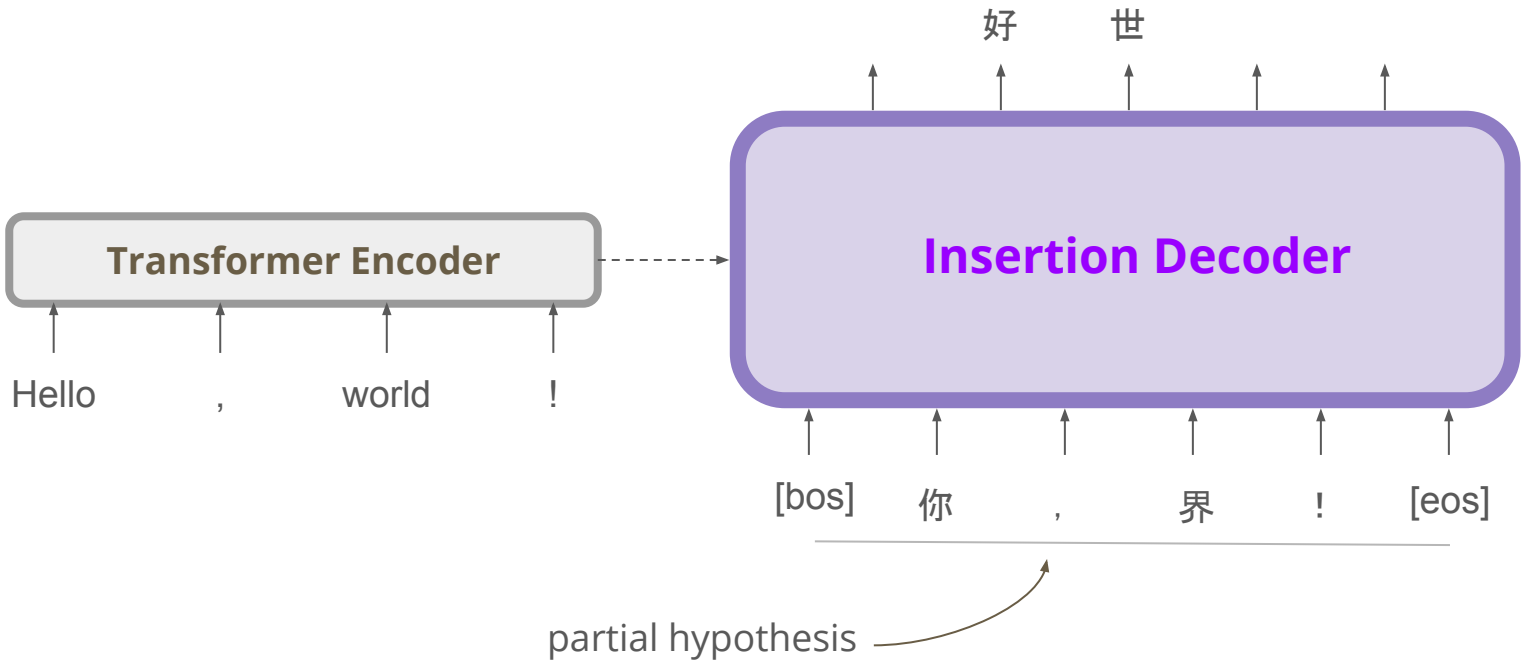
Results

Model	Dimensions (Model/Hidden)	Iterations	WMT'14 EN-DE	WMT'14 DE-EN	WMT'16 EN-RO	WMT'16 RO-EN
NAT w/ Fertility (Gu et al., 2018)	512/512	1	19.17	23.20	29.79	31.44
CTC Loss (Libovický and Helcl, 2018)	512/4096	1	17.68	19.80	19.93	24.71
Iterative Refinement (Lee et al., 2018)	512/512	1	13.91	16.77	24.45	25.73
	512/512	10	21.61	25.48	29.32	30.19
(Dynamic #Iterations)	512/512	?	21.54	25.43	29.66	30.30
Small CMLM with Mask-Predict	512/512	1	15.06	19.26	20.12	20.36
	512/512	4	**24.17**	**28.55**	**30.00**	30.43
	512/512	10	**25.51**	**29.47**	**31.65**	**32.27**
Base CMLM with Mask-Predict	512/2048	1	18.05	21.83	27.32	28.20
	512/2048	4	**25.94**	**29.90**	**32.53**	**33.23**
	512/2048	10	**27.03**	**30.53**	**33.08**	**33.31**
Base Transformer (Vaswani et al., 2017)	512/2048	N	27.30	— —	— —	— —
Base Transformer (Our Implementation)	512/2048	N	27.74	31.09	34.28	33.99
Base Transformer (+Distillation)	512/2048	N	27.86	31.07	— —	— —
Large Transformer (Vaswani et al., 2017)	1024/4096	N	28.40	— —	— —	— —
Large Transformer (Our Implementation)	1024/4096	N	28.60	31.71	— —	— —

[1902.03249] Insertion Transformer: Flexible Sequence Generation via Insertion Operations, Stern et al., ICML 2019.

Insertion Transformer

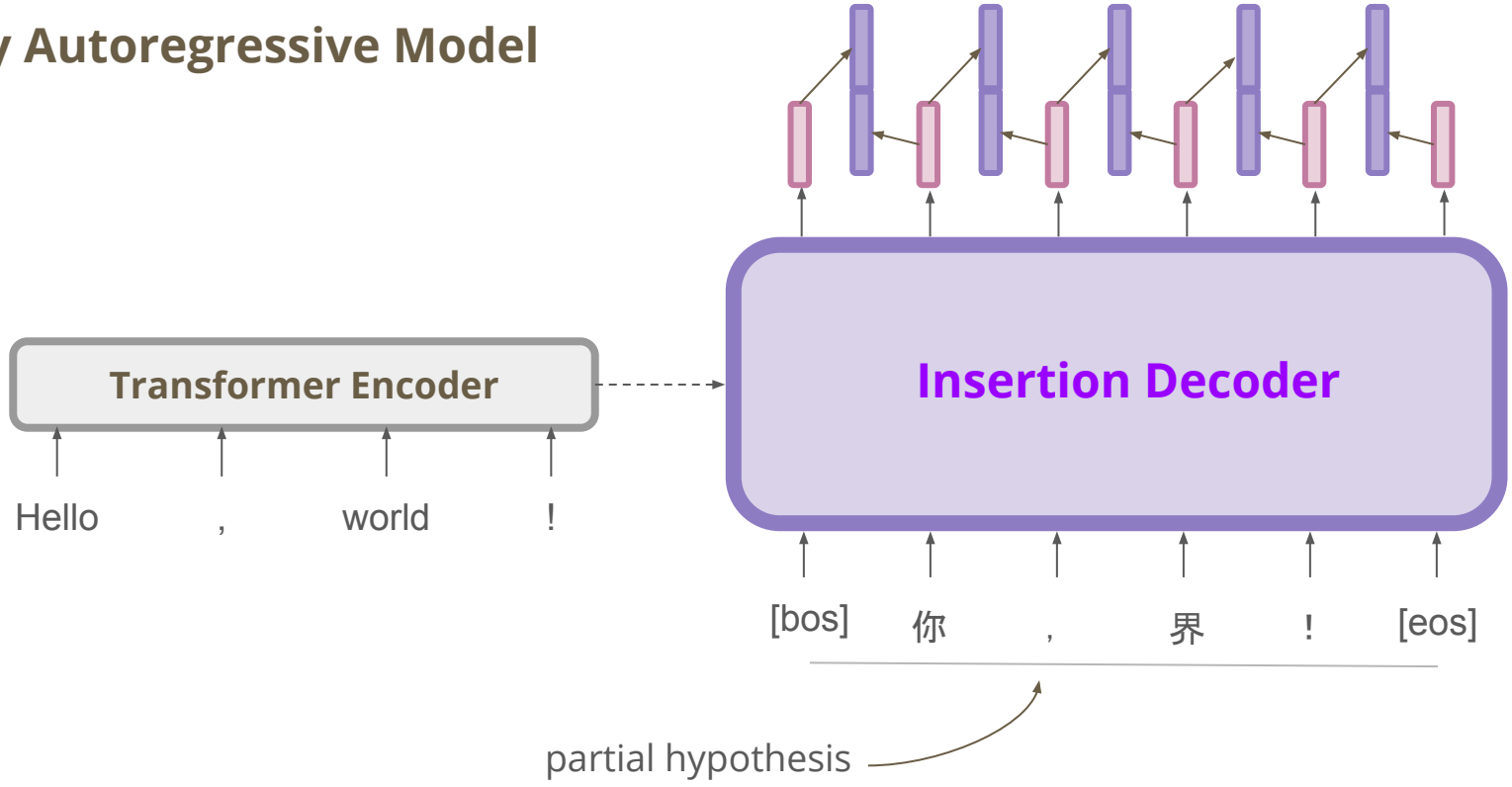
Partially Autoregressive Model



[1902.03249] Insertion Transformer: Flexible Sequence Generation via Insertion Operations, Stern et al., ICML 2019.

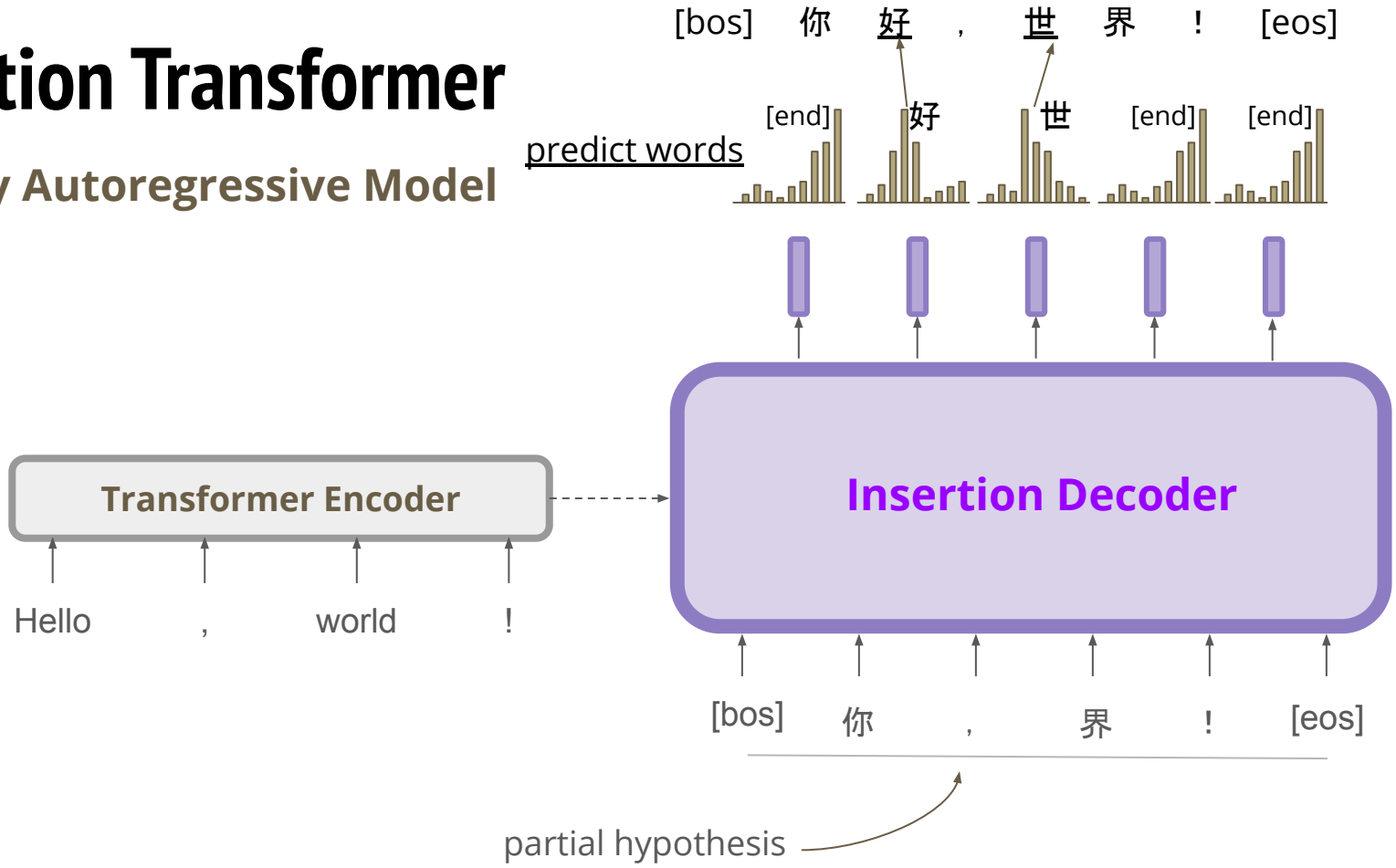
Insertion Transformer

Partially Autoregressive Model

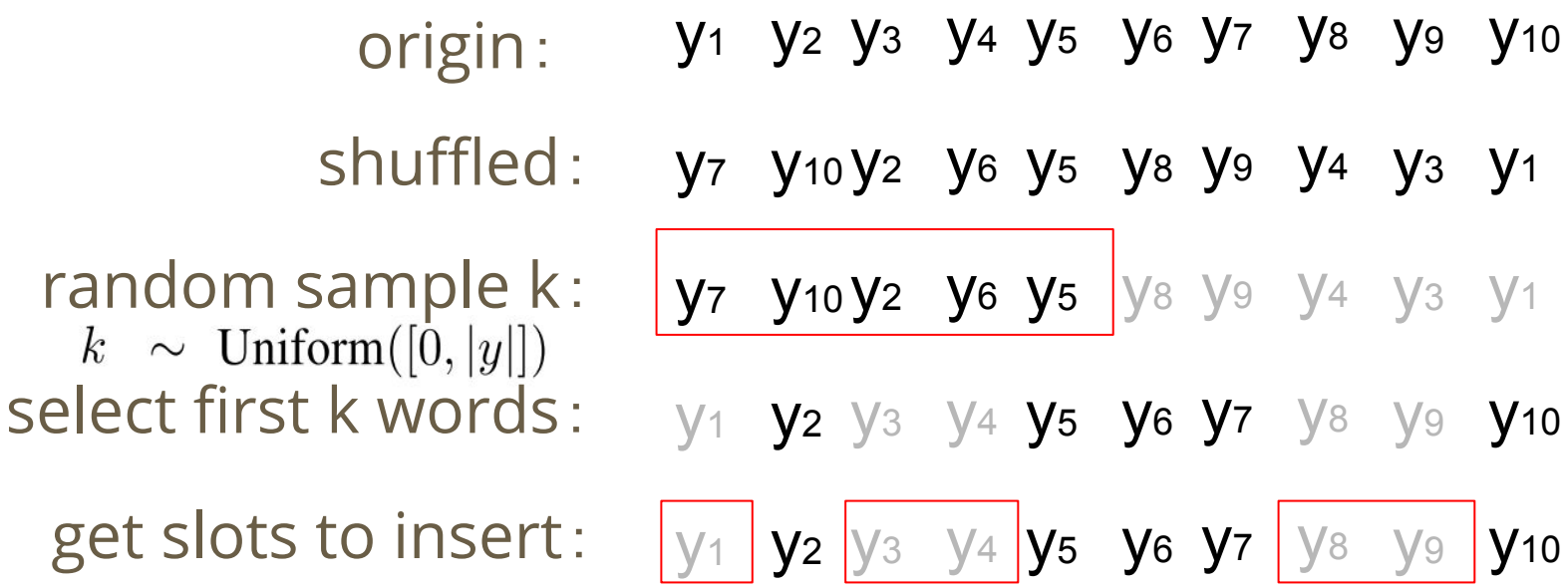


Insertion Transformer

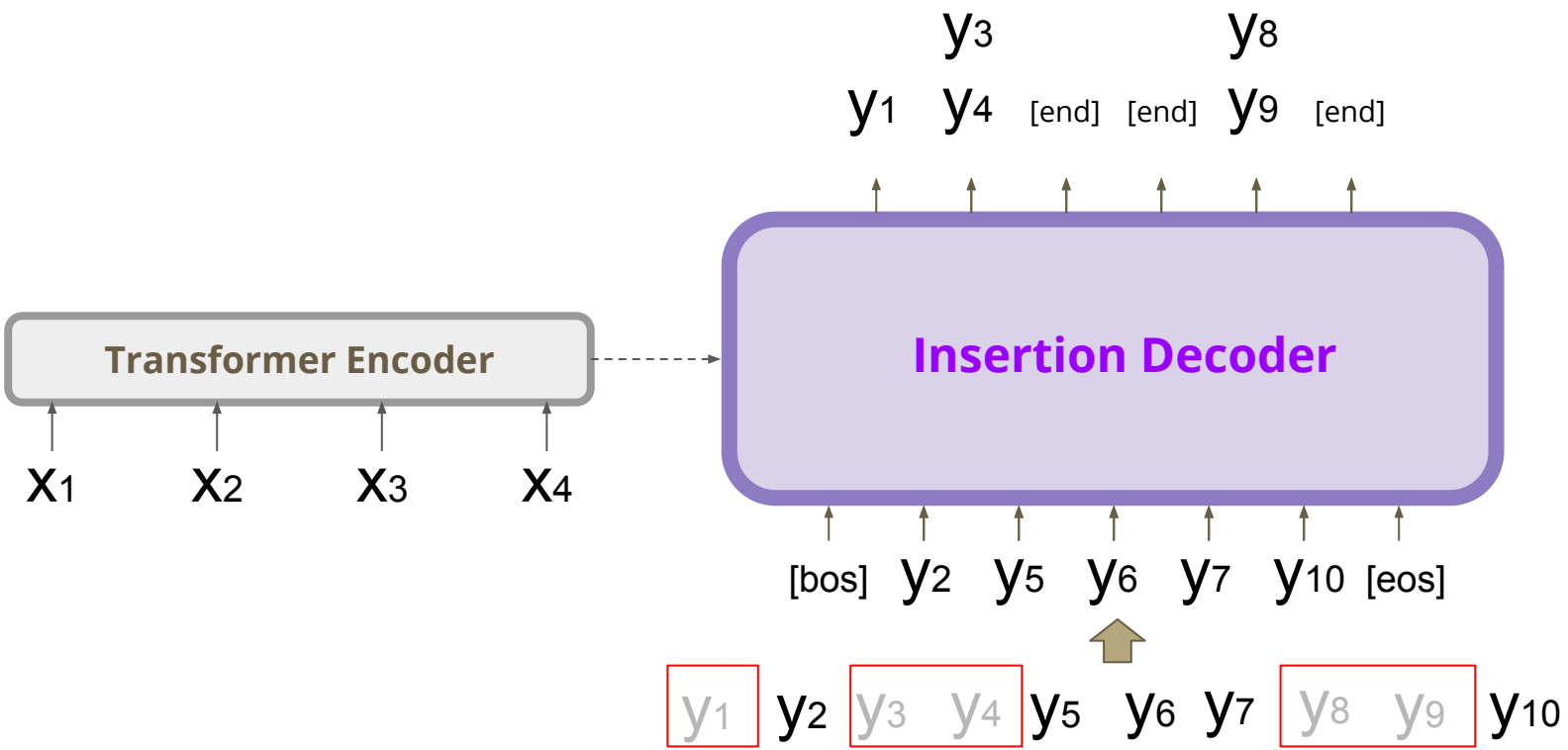
Partially Autoregressive Model



Training Examples

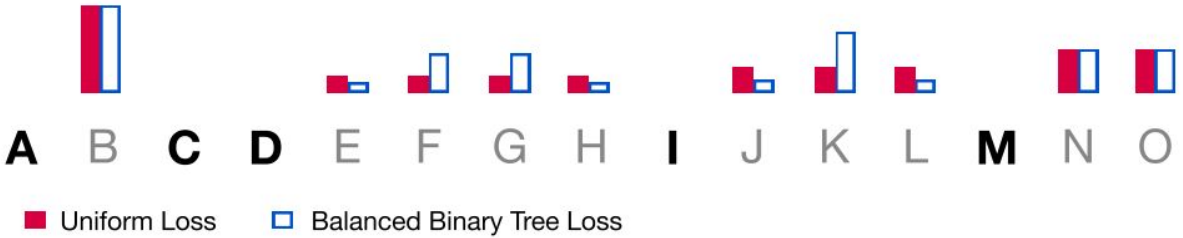


Training Examples



$$[] \rightarrow [D] \rightarrow [B, D, F] \rightarrow [A, B, C, D, E, F, G]$$

Multiple target words to predict?



Uniform Policy

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

Balanced Binary Tree Policy

$$w_l(i) = \frac{\exp(-d_l(i)/\tau)}{\sum_{i'=i_l}^{j_l} \exp(-d_l(i')/\tau)} \Rightarrow \text{slot-loss}(x, \hat{y}, l) = \sum_{i=i_l}^{j_l} -\log p(y_i, l \mid x, \hat{y}) \cdot w_l(i).$$

softmax weighting policy

Middle word has priority!!!

Example

Input: But on the other side of the state, that is not the impression many people have of their former governor.

Output: Aber auf der anderen Seite des Staates ist das nicht der Eindruck, den viele von ihrem ehemaligen Gouverneur haben.

Parallel decode (binary tree loss):

Aber_ auf_ der_ anderen_ Seite_ des_ Staates_ ist_ das_ nicht_ der_ Eindruck_ , _ den_ viele_ von_ ihrem_ ehemaligen_ Gouverneur _ haben_ ._

Aber_ auf_ der_ anderen_ Seite_ des_ Staates_ ist_ das_ nicht_ der_ Eindruck_ , _ den_ viele_ von_ ihrem_ ehemaligen_ Gouverneur _ haben_ ._

Aber_ auf_ der_ anderen_ Seite_ des_ Staates_ ist_ das_ nicht_ der_ Eindruck_ , _ den_ viele_ von_ ihrem_ ehemaligen_ Gouverneur _ haben_ ._

Aber_ auf_ der_ anderen_ Seite_ des_ Staates_ ist_ das_ nicht_ der_ Eindruck_ , _ den_ viele_ von_ ihrem_ ehemaligen_ Gouverneur _ haben_ ._

Aber_ auf_ der_ anderen_ Seite_ des_ Staates_ ist_ das_ nicht_ der_ Eindruck_ , _ den_ viele_ von_ ihrem_ ehemaligen_ Gouverneur _ haben_ ._

Input: They want to create a post on the college's equal opportunities committee to ensure that their opinions can be aired freely.

Output: Sie wollen einen Posten im Ausschuss für Chancengleichheit des Kollegiums einrichten, um sicherzustellen, dass ihre Meinungen frei zur Sprache gebracht werden können.

Parallel decode (uniform loss):

Sie_ wollen_ einen_ Posten_ im_ Ausschuss_ für_ Chancengleichheit_ des_ Koll egi ums_ einrichten_ , _ um_ sicherzustellen_ , _ dass_ ihre_ Mein ungen_ frei_ zur_ Sprache_ gebracht_ werden_ können_ ._

Sie_ wollen_ einen_ Posten_ im_ Ausschuss_ für_ Chancengleichheit_ des_ Koll egi ums_ einrichten_ , _ um_ sicherzustellen_ , _ dass_ ihre_ Mein ungen_ frei_ zur_ Sprache_ gebracht_ werden_ können_ ._

Sie_ wollen_ einen_ Posten_ im_ Ausschuss_ für_ Chancengleichheit_ des_ Koll egi ums_ einrichten_ , _ um_ sicherzustellen_ , _ dass_ ihre_ Mein ungen_ frei_ zur_ Sprache_ gebracht_ werden_ können_ ._

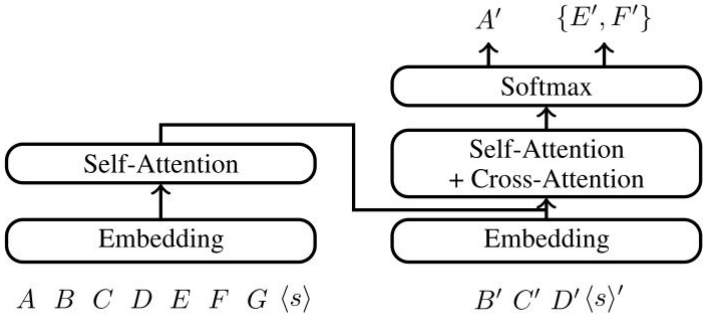
Sie_ wollen_ einen_ Posten_ im_ Ausschuss_ für_ Chancengleichheit_ des_ Koll egi ums_ einrichten_ , _ um_ sicherzustellen_ , _ dass_ ihre_ Mein ungen_ frei_ zur_ Sprache_ gebracht_ werden_ können_ ._

Sie_ wollen_ einen_ Posten_ im_ Ausschuss_ für_ Chancengleichheit_ des_ Koll egi ums_ einrichten_ , _ um_ sicherzustellen_ , _ dass_ ihre_ Mein ungen_ frei_ zur_ Sprache_ gebracht_ werden_ können_ ._

Sie_ wollen_ einen_ Posten_ im_ Ausschuss_ für_ Chancengleichheit_ des_ Koll egi ums_ einrichten_ , _ um_ sicherzustellen_ , _ dass_ ihre_ Mein ungen_ frei_ zur_ Sprache_ gebracht_ werden_ können_ ._

Sie_ wollen_ einen_ Posten_ im_ Ausschuss_ für_ Chancengleichheit_ des_ Koll egi ums_ einrichten_ , _ um_ sicherzustellen_ , _ dass_ ihre_ Mein ungen_ frei_ zur_ Sprache_ gebracht_ werden_ können_ ._

Decode stage



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)

n operations

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)

$\lfloor \log_2 n \rfloor + 1$ operations

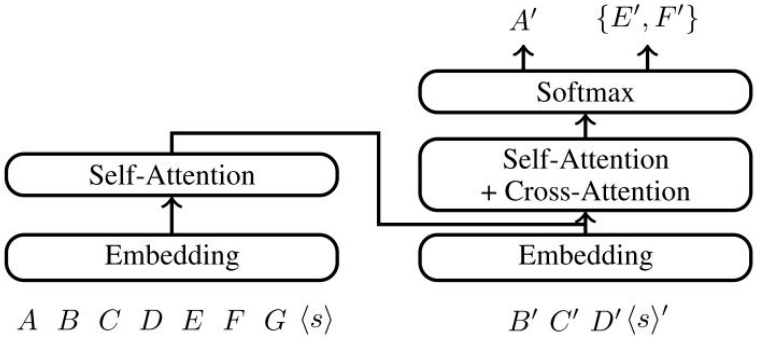
Comparison

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	$\approx \log_2 n$

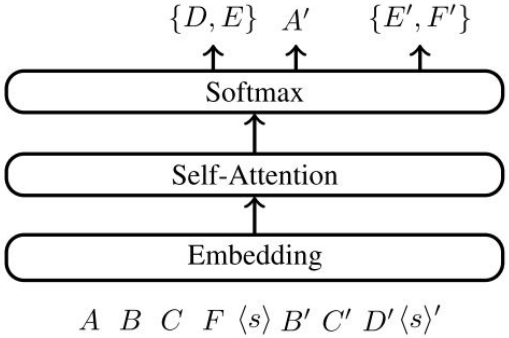
[1906.01604] KERMIT: Generative Insertion-Based Modeling for Sequences, Chan et al., arXiv 2019.

From P(y|x) to P(x, y), P(x), P(y), P(x|y), P(y|x)

KERMIT: Kontextuell Encoder Representations Made by Insertion Transformations



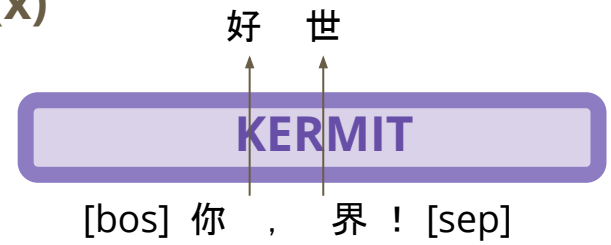
(c) Insertion Transformer



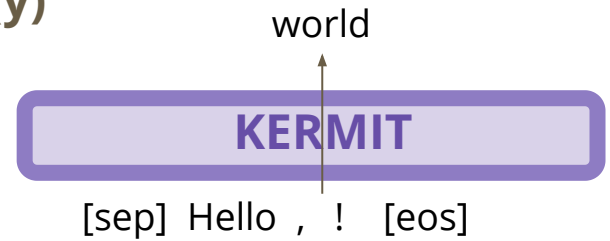
(d) KERMIT

Advantages

P(x)



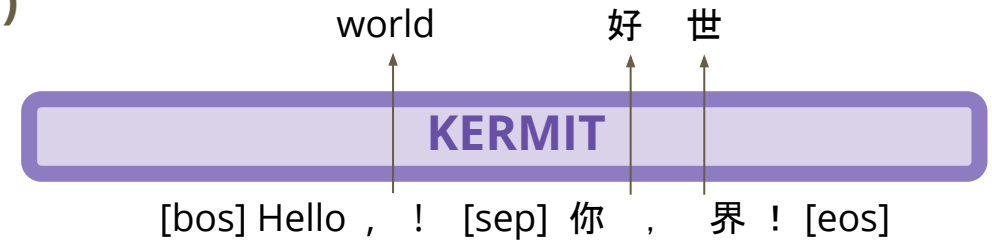
P(y)



P(y|x)

P(x|y)

P(x, y)



Results

Model	↔	En → De	De → En	Iterations
Autoregressive				
Transformer (Vaswani et al., 2017)	✗	27.3		n
Transformer (Our Implementation)	✗	27.8	31.2	n
Non-Autoregressive				
NAT (Gu et al., 2018)	✗	17.7	21.5	1
Iterative Refinement (Lee et al., 2018)	✗	21.6	25.5	10
Blockwise Parallel (Stern et al., 2018)	✗	27.4		$\approx n/5$
Insertion Transformer (Stern et al., 2019)	✗	27.4		$\approx \log_2 n \ll 10$
KERMIT				
Unidirectional ($p(y \mid x)$ or $p(x \mid y)$)	✗	27.8	30.7	$\approx \log_2 n \ll 10$
Bidirectional ($p(y \mid x)$ and $p(x \mid y)$)	✓	27.2	27.6	$\approx \log_2 n \ll 10$
Joint ($p(x, y)$)	✓	25.6	27.4	$\approx \log_2 n \ll 10$
+ Marginal Refining ($p(x)$ and $p(y)$)	✓	25.8	28.6	$\approx \log_2 n \ll 10$
↪ Unidirectional Finetuning	✗	28.7	31.4	$\approx \log_2 n \ll 10$
↪ Bidirectional Finetuning	✓	28.1	28.6	$\approx \log_2 n \ll 10$

Results

Model	↔	En → De	De → En	Iterations
Autoregressive				
Transformer (Vaswani et al., 2017)	✗	27.3		n
Transformer (Our Implementation)	✗	27.8	31.2	n
Non-Autoregressive				
NAT (Gu et al., 2018)	✗	17.7	21.5	1
Iterative Refinement (Lee et al., 2018)	✗	21.6	25.5	10
Blockwise Parallel (Stern et al., 2018)	✗	27.4		$\approx n/5$
Insertion Transformer (Stern et al., 2019)	✗	27.4		$\approx \log_2 n \ll 10$
KERMIT				
Unidirectional ($p(y \mid x)$ or $p(x \mid y)$)	✗	27.8	30.7	$\approx \log_2 n \ll 10$
Bidirectional ($p(y \mid x)$ and $p(x \mid y)$)	✓	27.2	27.6	$\approx \log_2 n \ll 10$
Joint ($p(x, y)$)	✓	25.6	27.4	$\approx \log_2 n \ll 10$
+ Marginal Refining ($p(x)$ and $p(y)$)	✓	25.8	28.6	$\approx \log_2 n \ll 10$
↪ Unidirectional Finetuning	✗	28.7	31.4	$\approx \log_2 n \ll 10$
↪ Bidirectional Finetuning	✓	28.1	28.6	$\approx \log_2 n \ll 10$

Results

Model	↔	En → De	De → En	Iterations
Autoregressive				
Transformer (Vaswani et al., 2017)	✗	27.3		n
Transformer (Our Implementation)	✗	27.8	31.2	n
Non-Autoregressive				
NAT (Gu et al., 2018)	✗	17.7	21.5	1
Iterative Refinement (Lee et al., 2018)	✗	21.6	25.5	10
Blockwise Parallel (Stern et al., 2018)	✗	27.4		$\approx n/5$
Insertion Transformer (Stern et al., 2019)	✗	27.4		$\approx \log_2 n \ll 10$
KERMIT				
Unidirectional ($p(y \mid x)$ or $p(x \mid y)$)	✗	27.8	30.7	$\approx \log_2 n \ll 10$
Bidirectional ($p(y \mid x)$ and $p(x \mid y)$)	✓	27.2	27.6	$\approx \log_2 n \ll 10$
Joint ($p(x, y)$)	✓	25.6	27.4	$\approx \log_2 n \ll 10$
+ Marginal Refining ($p(x)$ and $p(y)$)	✓	25.8	28.6	$\approx \log_2 n \ll 10$
↪ Unidirectional Finetuning	✗	28.7	31.4	$\approx \log_2 n \ll 10$
↪ Bidirectional Finetuning	✓	28.1	28.6	$\approx \log_2 n \ll 10$

Results

Model	↔	En → De	De → En	Iterations
Autoregressive				
Transformer (Vaswani et al., 2017)	✗	27.3		n
Transformer (Our Implementation)	✗	27.8	31.2	n
Non-Autoregressive				
NAT (Gu et al., 2018)	✗	17.7	21.5	1
Iterative Refinement (Lee et al., 2018)	✗	21.6	25.5	10
Blockwise Parallel (Stern et al., 2018)	✗	27.4		$\approx n/5$
Insertion Transformer (Stern et al., 2019)	✗	27.4		$\approx \log_2 n \ll 10$
KERMIT				
Unidirectional ($p(y \mid x)$ or $p(x \mid y)$)	✗	27.8	30.7	$\approx \log_2 n \ll 10$
Bidirectional ($p(y \mid x)$ and $p(x \mid y)$)	✓	27.2	27.6	$\approx \log_2 n \ll 10$
Joint ($p(x, y)$)	✓	25.6	27.4	$\approx \log_2 n \ll 10$
+ Marginal Refining ($p(x)$ and $p(y)$)	✓	25.8	28.6	$\approx \log_2 n \ll 10$
↪ Unidirectional Finetuning	✗	28.7	31.4	$\approx \log_2 n \ll 10$
↪ Bidirectional Finetuning	✓	28.1	28.6	$\approx \log_2 n \ll 10$

Results

Model	↔	En → De	De → En	Iterations
Autoregressive				
Transformer (Vaswani et al., 2017)	✗	27.3		n
Transformer (Our Implementation)	✗	27.8	31.2	n
Non-Autoregressive				
NAT (Gu et al., 2018)	✗	17.7	21.5	1
Iterative Refinement (Lee et al., 2018)	✗	21.6	25.5	10
Blockwise Parallel (Stern et al., 2018)	✗	27.4		$\approx n/5$
Insertion Transformer (Stern et al., 2019)	✗	27.4		$\approx \log_2 n \ll 10$
KERMIT				
Unidirectional ($p(y \mid x)$ or $p(x \mid y)$)	✗	27.8	30.7	$\approx \log_2 n \ll 10$
Bidirectional ($p(y \mid x)$ and $p(x \mid y)$)	✓	27.2	27.6	$\approx \log_2 n \ll 10$
Joint ($p(x, y)$)	✓	25.6	27.4	$\approx \log_2 n \ll 10$
+ Marginal Refining ($p(x)$ and $p(y)$)	✓	25.8	28.6	$\approx \log_2 n \ll 10$
↪ Unidirectional Finetuning	✗	28.7	31.4	$\approx \log_2 n \ll 10$
↪ Bidirectional Finetuning	✓	28.1	28.6	$\approx \log_2 n \ll 10$

GLUE

Model	Generative?	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-(m/mm)	QNLI	RTE	WNLI	AX	Score
GPT (Radford et al., 2018)	✓	45.4	91.3	82.3/75.7	82.0/80.0	70.3/88.5	82.1/81.4	87.4	56.0	53.4	29.8	72.8
BERT (Devlin et al., 2019)	✗	60.5	94.9	89.3/85.4	87.6/86.5	72.1/89.3	86.7/85.9	92.7	70.1	65.1	39.6	80.5
KERMIT	✓	60.0	94.2	88.6/84.3	86.6/85.6	71.7/89.0	85.6/85.2	92.0	68.4	65.1	37.6	79.8

Zero-shot ClozeQA

Model	Exact Match	F1
GPT-2	10.9	16.6
+ Oracle Length	12.2	18.2
BERT	12.3	18.9
+ Oracle Length	16.2	23.1
KERMIT	**20.9**	**30.3**

Table 4: SQuAD zero-shot cloze question answering.

Plymouth has a post-war shopping area in the city centre with substantial pedestrianisation. At the west end of the zone inside a grade II listed building is the Pannier Market that was completed in 1959 – pannier meaning "basket" from French, so it translates as "basket market". In terms of retail floorspace, Plymouth is ranked in the top five in the South West, and 29th nationally. Plymouth was one of the first ten British cities to trial the new Business Improvement District initiative. The Tinside Pool is situated at the foot of the Hoe and became a grade II listed building in 1998 before being restored to its 1930s look for £3.4 million. **What notable location was named a grade II listed building in 1998? ___ was named a grade II listed building in 1998 .**

Model	Answer
GPT-2	→ "Plymouth"
+ Oracle Length	→ "A listed building"
BERT	→ "plymouth"
+ Oracle Length	→ ": the pool"
KERMIT	→ **"the tinside pool"**
Correct	→ "Tinside Pool"

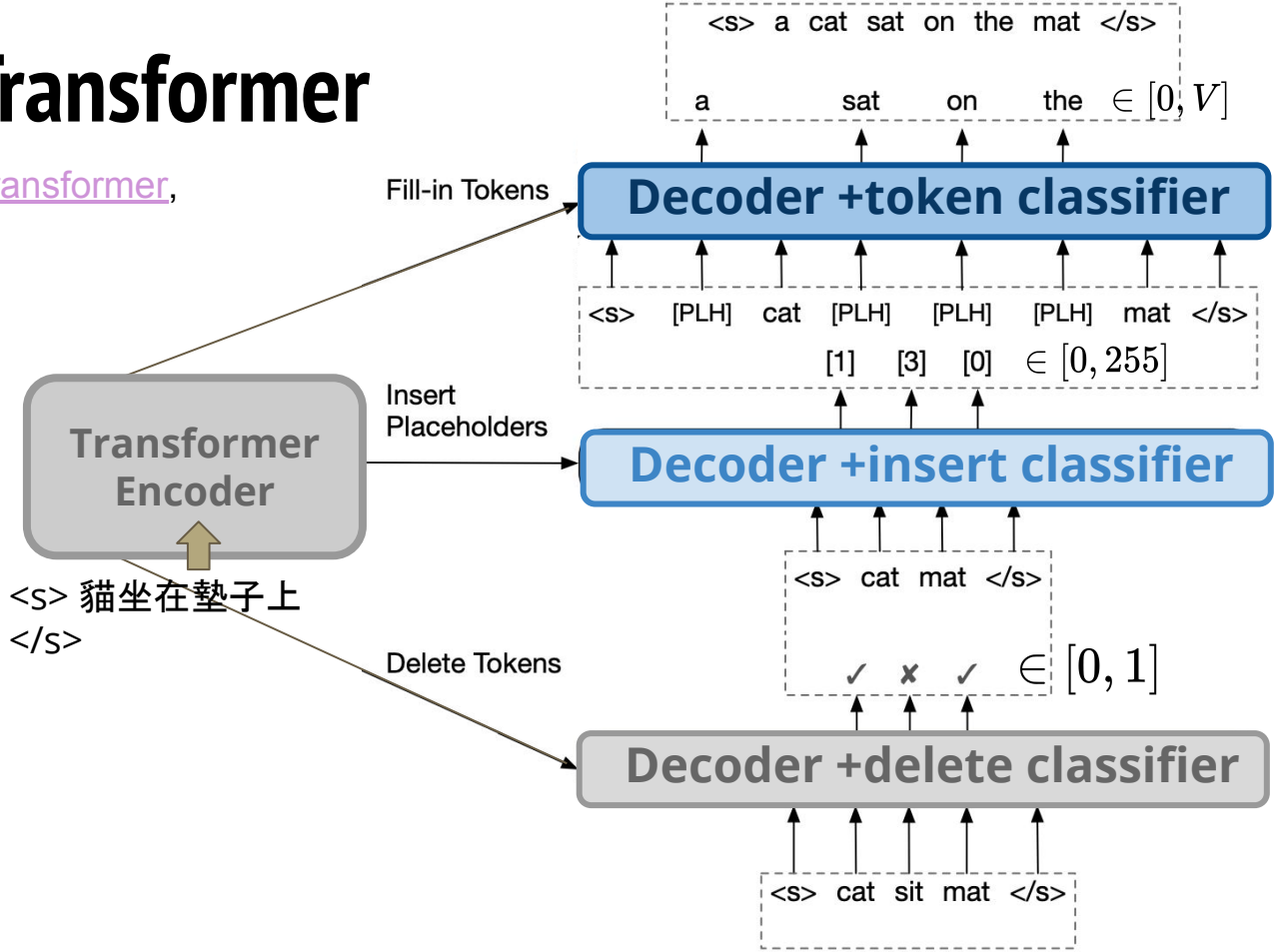
Multilingual KERMIT: It's Not Easy Being Generative, Chan et al., Perception as Generative Reasoning Workshop @NeurIPS 2019.

Multilingual KERMIT: It's Not Easy Being Generative

KERMIT can be trained by feeding it N way parallel-data, bilingual data, or monolingual data. At inference, KERMIT can generate translations for a particular target language, or up to N −1 languages in parallel.

Levenshtein Transformer

[1905.11006] Levenshtein Transformer,
Gu et al., NeurIPS 2019.



How to train?

Imitation Learning

Learn from expert: Levenshtein Distance Algorithm

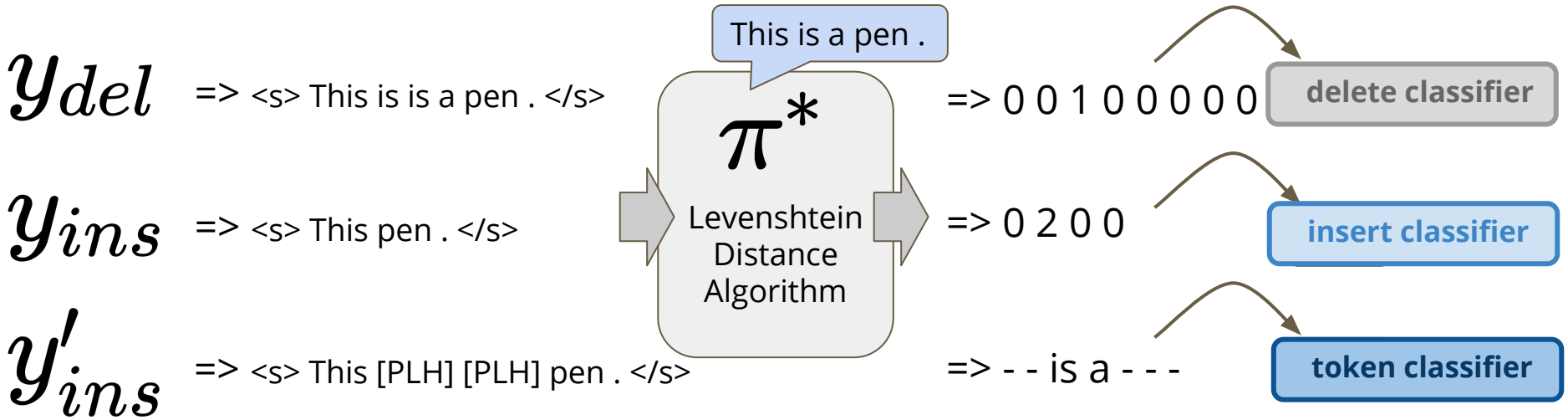
```
>>> import Levenshtein
>>> Levenshtein.distance("ABCEFGHJJ", "ABCDEFGHI")
3
>>> Levenshtein.editops("ABCEFGHJJ", "ABCDEFGHI")
[('insert', 3, 3), ('delete', 7, 8), ('replace', 8, 8)]
```

In this paper we decompose ‘replace’ into delete+insert.

Imitation Learning

y_{del} sentences that need to delete words

y_{ins} sentences that need to insert words



Examples

(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 >>	[__][回転][回転][すぎ][ると][逆][変形][が生じる][。]
(iteration 2)	*delete >>*	[__][~~回転~~][回転][すぎ][ると][逆][変形][が生じる][。]
	insert >>	[__][回転][速度が][すぎ][ると][，][逆][変形][が生じる][。]
(iteration 3)	*nothing to delete >>*	[__][回転][速度が][すぎ][ると][，][逆][変形][が生じる][。]
	insert >>	[__][回転][速度が][高][すぎ][ると][，][逆][変形][が生じる][。]
nothing to delete, nothing to insert >>		**[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][回路][を提案し][，][検証した][。]
(iteration 2)	*delete >>*	[__][可能な][~~構造~~][~~回路~~][回路][を提案し][，][検証した][。]
	insert >>	[__][いくつかの][可能な][構造と][回路][を提案し][，][検証した][。]
nothing to delete, nothing to insert >>		**[Terminate]**

Results

	Dataset	Metric	Transformer greedy	Transformer beam4	Levenshtein Transformer oracle	Levenshtein Transformer teacher
Quality ↑	Ro-En	BLEU	31.67	32.30	**33.02**	–
	En-De	BLEU	26.02	26.56	24.43	**26.67**
	En-Ja	BLEU	42.86	**43.68**	42.36	43.17
	Gigaword	ROUGE-1	34.91	35.19	35.57	**36.08**
		ROUGE-2	17.05	17.58	17.11	**18.33**
		ROUGE-L	32.66	32.98	33.55	**33.81**
Speed ↓	Ro-En	Latency (ms) / I_{DEC}	326 / 27.1	349 / 27.1	**97 / 2.19**	–
	En-De	Latency (ms) / I_{DEC}	343 / 28.1	369 / 28.1	126 / 2.88	**92 / 2.05**
	En-Ja	Latency (ms) / I_{DEC}	261 / 22.6	306 / 22.6	112 / 2.61	**106 / 1.97**
	Gigaword	Latency (ms) / I_{DEC}	116 / 10.1	149 / 10.1	98 / 2.32	**84 / 1.73**

Table 1: Generation quality (BLEU ↑, ROUGE-1/2/L ↑) and latency (ms ↓) as well as the average number of decoder iterations (I_{DEC}) on the standard test sets for LevT and the autoregressive baseline (with both greedy and beam-search outputs). We show the results of LevT trained from both oracle and the autoregressive teacher model.

Results

	Dataset	Metric	Transformer greedy	Transformer beam4	Levenshtein Transformer oracle	Levenshtein Transformer teacher
Quality ↑	Ro-En	BLEU	31.67	32.30	**33.02**	–
	En-De	BLEU	26.02	26.56	24.43	**26.67**
	En-Ja	BLEU	42.86	**43.68**	42.36	43.17
	Gigaword	ROUGE-1	34.91	35.19	35.57	**36.08**
		ROUGE-2	17.05	17.58	17.11	**18.33**
		ROUGE-L	32.66	32.98	33.55	**33.81**
Speed ↓	Ro-En	Latency (ms) / I_{DEC}	326 / 27.1	349 / 27.1	**97 / 2.19**	–
	En-De	Latency (ms) / I_{DEC}	343 / 28.1	369 / 28.1	126 / 2.88	**92 / 2.05**
	En-Ja	Latency (ms) / I_{DEC}	261 / 22.6	306 / 22.6	112 / 2.61	**106 / 1.97**
	Gigaword	Latency (ms) / I_{DEC}	116 / 10.1	149 / 10.1	98 / 2.32	**84 / 1.73**

Table 1: Generation quality (BLEU ↑, ROUGE-1/2/L ↑) and latency (ms ↓) as well as the average number of decoder iterations (I_{DEC}) on the standard test sets for LevT and the autoregressive baseline (with both greedy and beam-search outputs). We show the results of LevT trained from both oracle and the autoregressive teacher model.

Results

	Dataset	Metric	Transformer greedy	Transformer beam4	Levenshtein Transformer oracle	Levenshtein Transformer teacher
Quality ↑	Ro-En	BLEU	31.67	32.30	**33.02**	–
	En-De	BLEU	26.02	26.56	24.43	**26.67**
	En-Ja	BLEU	42.86	**43.68**	42.36	43.17
	Gigaword	ROUGE-1	34.91	35.19	35.57	**36.08**
		ROUGE-2	17.05	17.58	17.11	**18.33**
		ROUGE-L	32.66	32.98	33.55	**33.81**
Speed ↓	Ro-En	Latency (ms) / I_{DEC}	326 / 27.1	349 / 27.1	**97 / 2.19**	–
	En-De	Latency (ms) / I_{DEC}	343 / 28.1	369 / 28.1	126 / 2.88	**92 / 2.05**
	En-Ja	Latency (ms) / I_{DEC}	261 / 22.6	306 / 22.6	112 / 2.61	**106 / 1.97**
	Gigaword	Latency (ms) / I_{DEC}	116 / 10.1	149 / 10.1	98 / 2.32	**84 / 1.73**

Table 1: Generation quality (BLEU ↑, ROUGE-1/2/L ↑) and latency (ms ↓) as well as the average number of decoder iterations (I_{DEC}) on the standard test sets for LevT and the autoregressive baseline (with both greedy and beam-search outputs). We show the results of LevT trained from both oracle and the autoregressive teacher model.

Results

	Dataset	Metric	Transformer greedy	Transformer beam4	Levenshtein Transformer oracle	Levenshtein Transformer teacher
Quality ↑	Ro-En	BLEU	31.67	32.30	**33.02**	–
	En-De	BLEU	26.02	26.56	24.43	**26.67**
	En-Ja	BLEU	42.86	**43.68**	42.36	43.17
	Gigaword	ROUGE-1	34.91	35.19	35.57	**36.08**
		ROUGE-2	17.05	17.58	17.11	**18.33**
		ROUGE-L	32.66	32.98	33.55	**33.81**
Speed ↓	Ro-En	Latency (ms) / I_{DEC}	326 / 27.1	349 / 27.1	**97** / **2.19**	–
	En-De	Latency (ms) / I_{DEC}	343 / 28.1	369 / 28.1	126 / 2.88	**92** / **2.05**
	En-Ja	Latency (ms) / I_{DEC}	261 / 22.6	306 / 22.6	112 / 2.61	**106** / **1.97**
	Gigaword	Latency (ms) / I_{DEC}	116 / 10.1	149 / 10.1	98 / 2.32	**84** / **1.73**

Table 1: Generation quality (BLEU ↑, ROUGE-1/2/L ↑) and latency (ms ↓) as well as the average number of decoder iterations (I_{DEC}) on the standard test sets for LevT and the autoregressive baseline (with both greedy and beam-search outputs). We show the results of LevT trained from both oracle and the autoregressive teacher model.

Results

	Dataset	Metric	Transformer greedy	Transformer beam4	Levenshtein Transformer oracle	Levenshtein Transformer teacher
Quality ↑	Ro-En	BLEU	31.67	32.30	**33.02**	–
	En-De	BLEU	26.02	26.56	24.43	**26.67**
	En-Ja	BLEU	42.86	**43.68**	42.36	43.17
	Gigaword	ROUGE-1	34.91	35.19	35.57	**36.08**
		ROUGE-2	17.05	17.58	17.11	**18.33**
		ROUGE-L	32.66	32.98	33.55	**33.81**
Speed ↓	Ro-En	Latency (ms) / I_{DEC}	326 / 27.1	349 / 27.1	**97 / 2.19**	–
	En-De	Latency (ms) / I_{DEC}	343 / 28.1	369 / 28.1	126 / 2.88	**92 / 2.05**
	En-Ja	Latency (ms) / I_{DEC}	261 / 22.6	306 / 22.6	112 / 2.61	**106 / 1.97**
	Gigaword	Latency (ms) / I_{DEC}	116 / 10.1	149 / 10.1	98 / 2.32	**84 / 1.73**

Table 1: Generation quality (BLEU ↑, ROUGE-1/2/L ↑) and latency (ms ↓) as well as the average number of decoder iterations (I_{DEC}) on the standard test sets for LevT and the autoregressive baseline (with both greedy and beam-search outputs). We show the results of LevT trained from both oracle and the autoregressive teacher model.

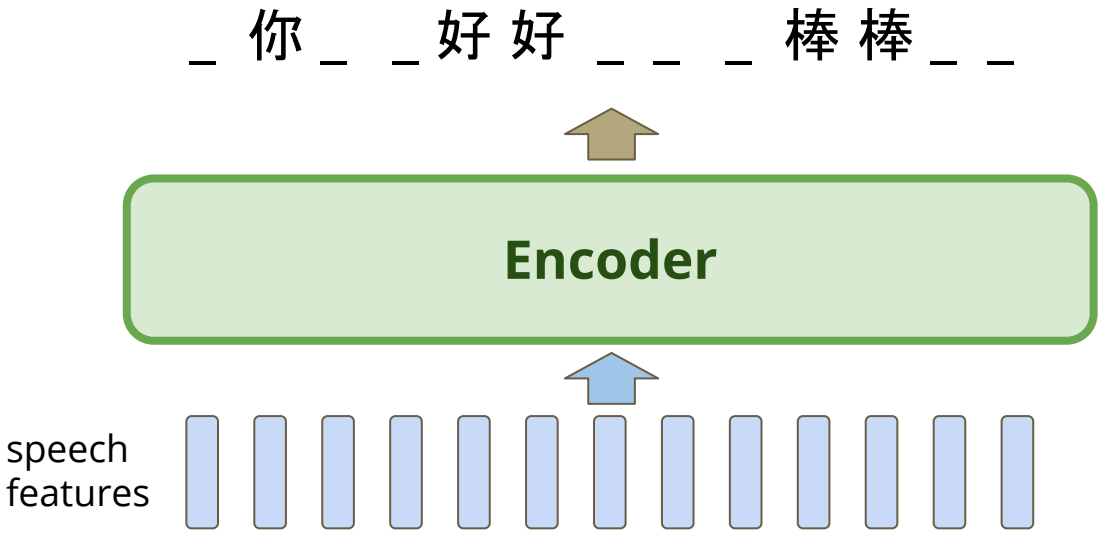
CTC

Connectionist Temporal Classification : Labelling Unsegmented Sequence Data with Recurrent Neural Networks, Graves et al., ICML 2006

- also non-autoregressive model (for speech recognition)
- speech data doesn't have serious multi-modality problem!

shortcoming:

1. Falls behind LAS
2. Can't be refined



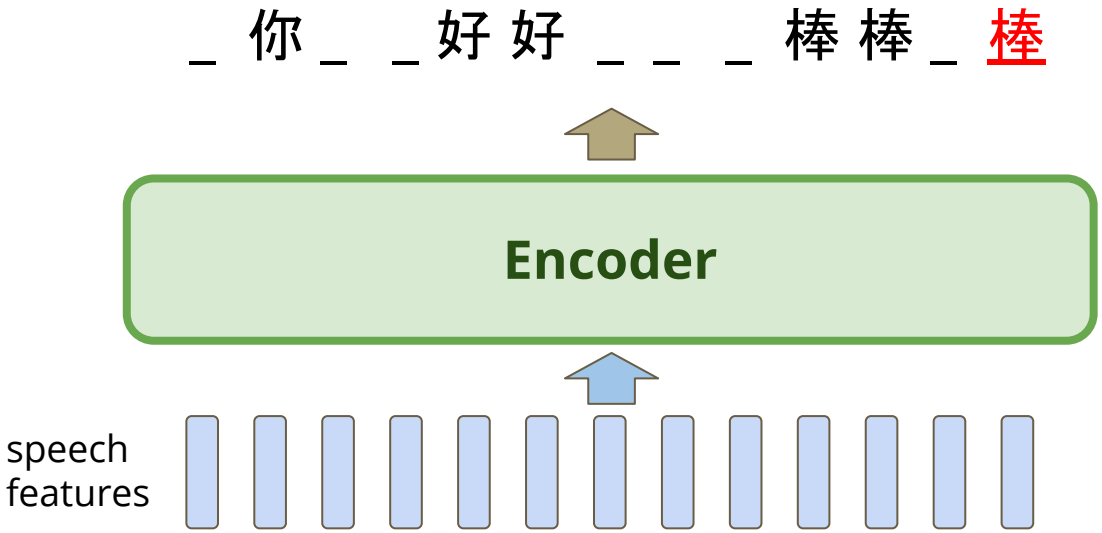
CTC

Connectionist Temporal Classification : Labelling Unsegmented Sequence Data with Recurrent Neural Networks, Graves et al., ICML 2006

- also non-autoregressive model (for speech recognition)
- speech data doesn't have serious multi-modality problem!

shortcoming:

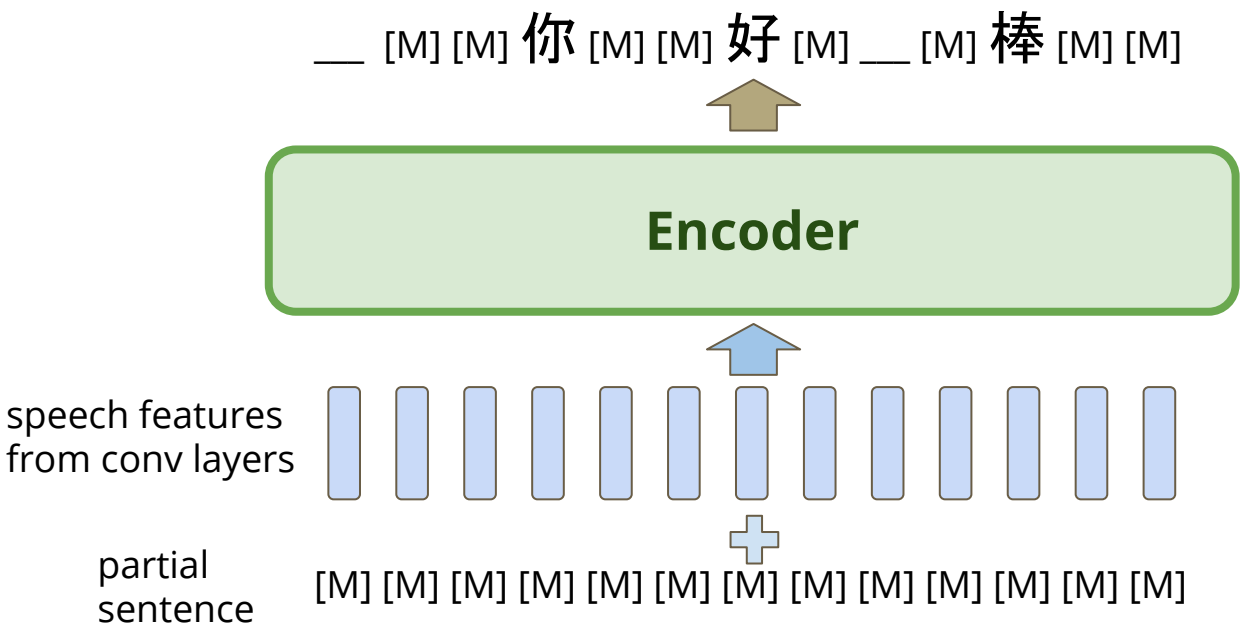
1. Falls behind LAS
2. Can't be refined



[2002.08926] Imputer: Sequence Modelling via Imputation and Dynamic Programming, Chan et al., arXiv 2020

Imputer (CTC+Mask-Predict)

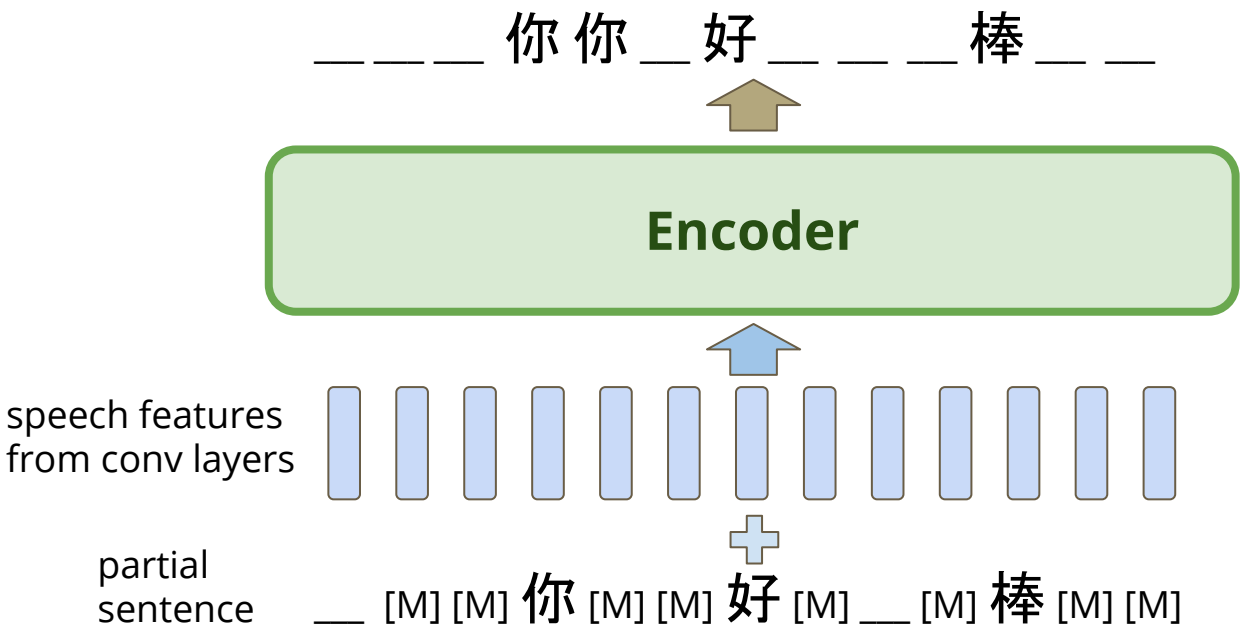
t = 0



[2002.08926] Imputer: Sequence Modelling via Imputation and Dynamic Programming, Chan et al., arXiv 2020

Imputer (CTC+Mask-Predict)

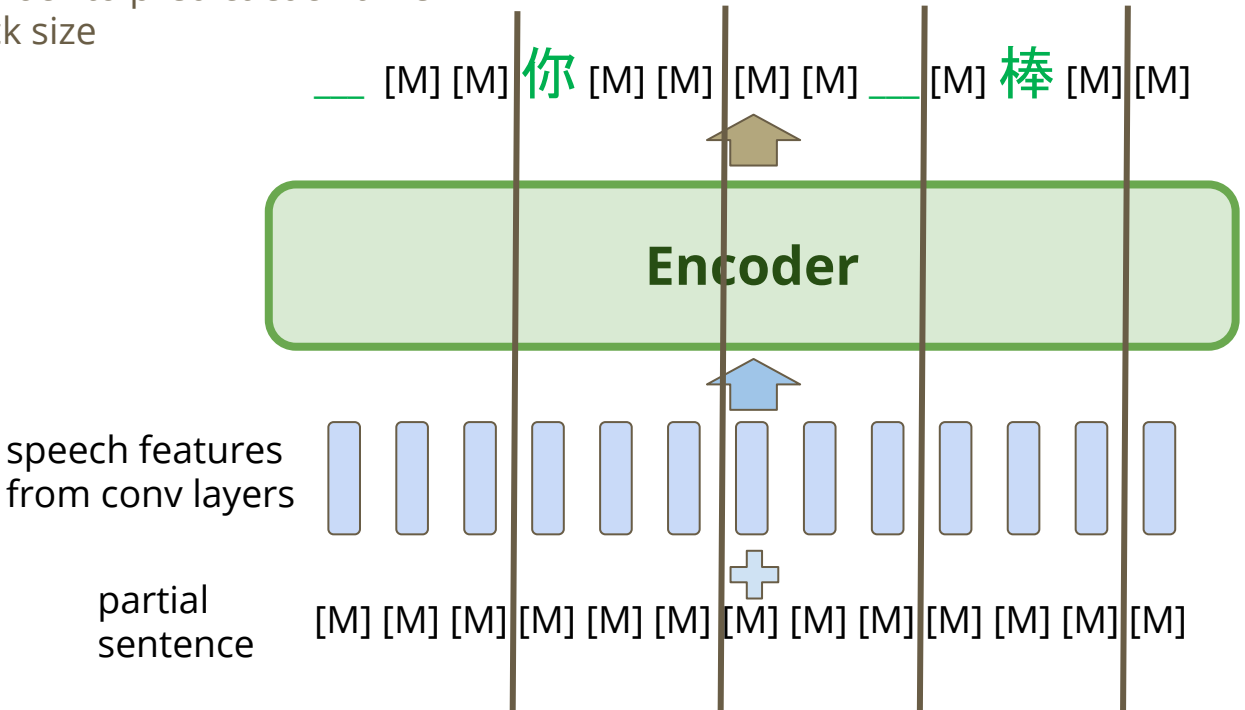
t = 1



[2002.08926] Imputer: Sequence Modelling via Imputation and Dynamic Programming, Chan et al., arXiv 2020

Block Decoding

ex: Decode Block Size = 3
pick one position in a block to predict each time
=> max iteration = block size



Example

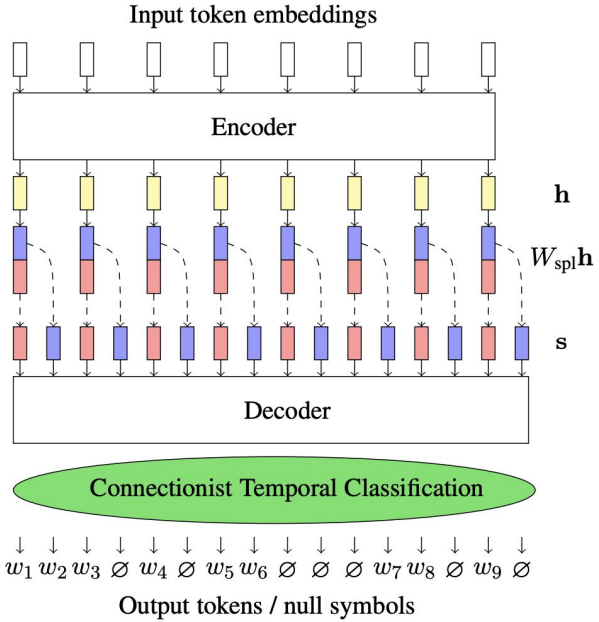
b	Alignment																															
0	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅
1	∅	∅	∅	[HAVE	∅	∅	∅	∅	∅	[L	∅	∅	∅	∅	∅	∅	∅	∅	∅	Y	∅	∅	∅	∅	∅	∅	L	∅	∅	∅	∅	∅
2	∅	∅	∅	[HAVE	∅	∅	[TO	∅	∅	[L	∅	∅	IVE	∅	∅	∅	∅	∅	∅	Y	∅	∅	[SE	∅	∅	∅	L	∅	F	∅	∅	∅
3	∅	∅	_	[HAVE	∅	∅	[TO	∅	_	[L	∅	∅	IVE	∅	∅	∅	∅	[M	∅	Y	∅	∅	[SE	∅	_	∅	L	∅	F	∅	∅	∅
4	∅	∅	_	[HAVE	∅	∅	[TO	_	_	[L	∅	∅	IVE	_	∅	∅	∅	[M	∅	Y	∅	∅	[SE	_	_	∅	L	∅	F	∅	∅	_
5	_	∅	_	[HAVE	∅	∅	[TO	_	_	[L	∅	_	IVE	_	∅	∅	∅	[M	_	Y	∅	∅	[SE	_	_	_	L	∅	F	∅	∅	_
6	_	∅	_	[HAVE	∅	_	[TO	_	_	[L	∅	_	IVE	_	[WITH	∅	∅	[M	_	Y	∅	_	[SE	_	_	_	L	∅	F	∅	_	_
7	_	_	_	[HAVE	∅	_	[TO	_	_	[L	_	_	IVE	_	[WITH	∅	_	[M	_	Y	∅	_	[SE	_	_	_	L	_	F	∅	_	_
8	_	_	_	[HAVE	_	_	[TO	_	_	[L	_	_	IVE	_	[WITH	_	_	[M	_	Y	_	_	[SE	_	_	_	L	_	F	_	_	_

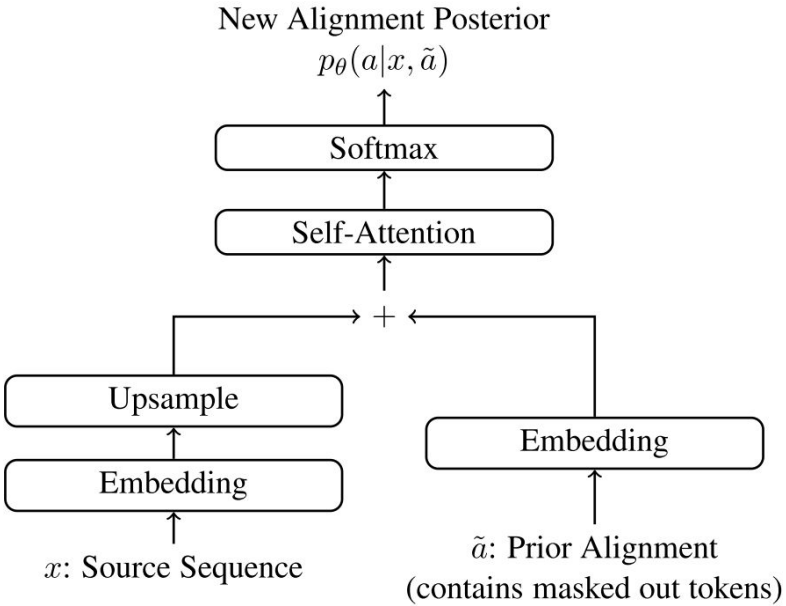
Results

Table 1. Wall Street Journal Character Error Rate (CER) and Word Error Rate (WER).

Model	CER	WER	Iterations
seq2seq			
Bahdanau et al. (2016a)	6.4	18.6	n
Bahdanau et al. (2016b)	5.9	18.0	n
Chorowski & Jaitly (2017)	-	10.6	n
Zhang et al. (2017)	-	10.5	n
Chan et al. (2017)	-	9.6	n
Kim et al. (2017)	7.4	-	n
Serdyuk et al. (2018)	6.2	-	n
Tjandra et al. (2018)	6.1	-	n
Sabour et al. (2019)	3.1	9.3	n
CTC			
Graves & Jaitly (2014)	8.4	27.3	1
Liu et al. (2017)	-	16.7	1
CTC (Our Work)	5.6	15.2	1
Imputer (IM)	6.2	16.5	8
Imputer (DP)	4.9	12.7	8

Applying CTC/Imputer on text generation





[1811.04719] End-to-End Non-Autoregressive Neural Machine Translation with Connectionist Temporal Classification, Libovický et al., EMNLP 2018
[2004.07437] Non-Autoregressive Machine Translation with Latent Alignments, Chan et al., arXiv 2020

Results

compared with single step models

Method	Iterations	WMT'14 En→De	WMT'14 De→En	WMT'16 En→Ro	WMT'16 Ro→En
Non-Autoregressive					
Iterative Refinement (Lee et al., 2018)	1	13.9	16.7	24.5	25.7
NAT with Fertility (Gu et al., 2018)	1	17.7	21.5	27.3	29.1
CTC† (Libovicky and Helcl, 2018)	1	17.7	19.8	19.9	24.7
Mask-Predict (Ghazvininejad et al., 2019)	1	18.0	19.3	27.3	28.2
SMART (Ghazvininejad et al., 2020b)	1	18.6	23.8	-	-
Auxiliary Regularization (Wang et al., 2019)	1	20.7	24.8	-	-
Bag-of-ngrams Loss (Shao et al., 2020)	1	20.9	24.6	28.3	29.3
Hint-based Training (Li et al., 2019)	1	21.1	25.2	-	-
FlowSeq (Ma et al., 2019)	1	21.5	26.2	29.3	30.4
Bigram CRF (Sun et al., 2019)	1	23.4	27.2	-	-
AXE CMLM (Ghazvininejad et al., 2020a)	1	23.5	27.9	30.8	31.5
Our Work					
CTC†	1	25.7	28.1	32.2	31.6
Imputer	1	**25.8**	**28.4**	**32.3**	**31.7**

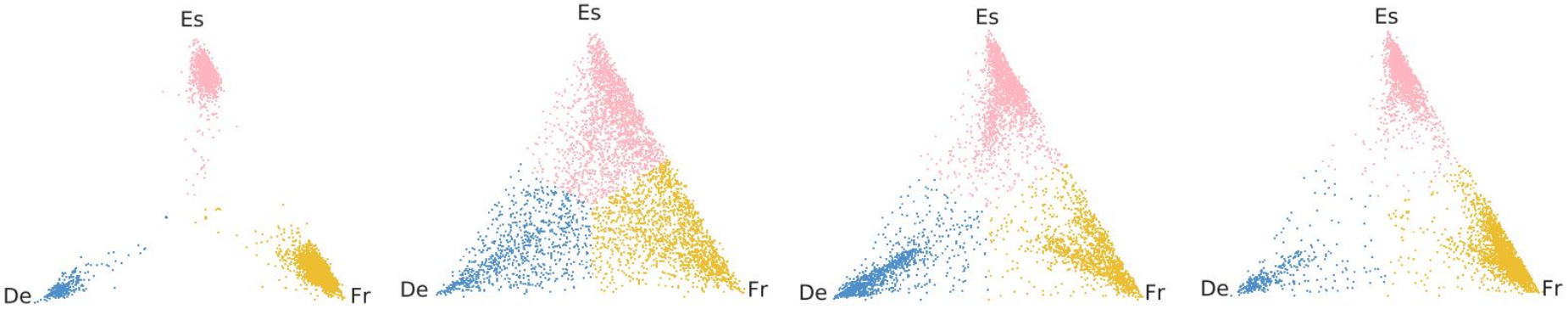
Results

compared with multiple refine steps models

Method	Iterations	WMT'14 En→De	WMT'14 De→En	WMT'16 En→Ro	WMT'16 Ro→En
Autoregressive					
Base Transformer[2]	n	27.8	31.2	34.3	34.0
Non-Autoregressive					
Insertion Transformer (Stern et al., 2019)	$\approx \log_2 n$	27.4	-	-	-
KERMIT (Chan et al., 2019b)	$\approx \log_2 n$	27.8	30.7	-	-
Iterative Refinement (Lee et al., 2018)	10	21.6	25.5	29.3	30.2
Mask-Predict (Ghazvininejad et al., 2019)	4	25.9	29.9	32.5	33.2
	10	27.0	30.5	33.1	33.3
SMART (Ghazvininejad et al., 2020b)	4	27.0	30.9	-	-
	10	27.7	**31.3**	-	-
Our Work					
Imputer	2	27.5	30.2	33.7	33.4
	4	28.0	31.0	34.3	34.0
	8	**28.2**	**31.3**	**34.4**	**34.1**

[1911.02727] Understanding Knowledge Distillation in Non-autoregressive Machine Translation, Gu et al., ICLR 2020

Knowledge Distillation in NAT



(a) AT Baseline (b) NAT Baseline (c) NAT Random Select (d) NAT Distill

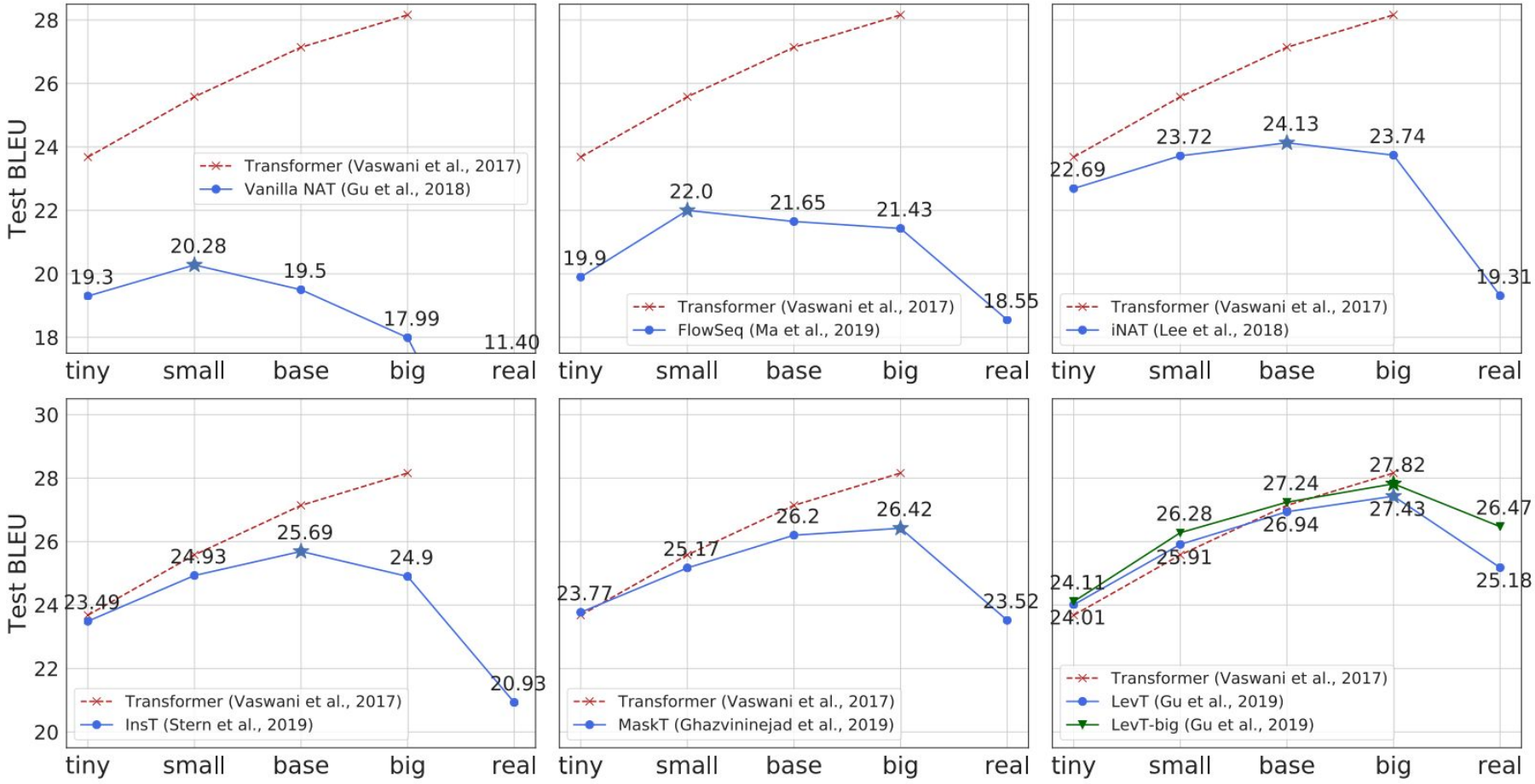


Figure 4: The performance of NAT models of varying capacity trained on both the real and the distilled data from tiny, small, base and big AT models.

Reference

Jiatao Gu

@Facebook AI Research

https://jiataogu.me/

1. **Vanilla NAT** (Gu et al., 2018)
   Non-Autoregressive Neural Machine Translation, ICLR 2018
2. **Levenshtein Transformer** (LevT, Gu et al., 2019)
   Levenshtein Transformer, NIPS 2019
3. **Knowledge Distillation in NAT** (Zhou et al., 2019)
   Understanding Knowledge Distillation in Non-autoregressive Machine Translation, ICLR 2020

Reference

William Chan @Google Brain
& Mitchell Stern

http://williamchan.ca/

4. **Insertion Transformer** (InsT, Stern et al., 2019)
Insertion Transformer: Flexible Sequence Generation via Insertion Operations, ICML 2019
5. **KERMIT** (Chan et al., 2019)
KERMIT: Generative Insertion-Based Modeling for Sequences, Preprint
6. **Multilingual KERMIT** (Chan et al., 2019)
Multilingual KERMIT: It's Not Easy Being Generative, PGR@NeurIPS 2019
7. **Imputer** (Chan et al., 2020)
Imputer: Sequence Modelling via Imputation and Dynamic Programming, Preprint
8. **NAT with Latent Alignments** (Saharia et al., 2020)
Non-Autoregressive Machine Translation with Latent Alignments, Preprint

(The first author of "Listen, Attend, Spell")

Reference

Others

9. **NAT with Iterative Refinement** (Lee et al., 2018)
   Deterministic Non-Autoregressive Neural Sequence Modeling by Iterative Refinement, EMNLP 2018
10. **MaskPredict** (Ghazvininejad et al., 2019)
    Mask-Predict: Parallel Decoding of Conditional Masked Language Models, EMNLP 2019
11. **CTC** (Graves et al., 2006)
    Connectionist Temporal Classification : Labelling Unsegmented Sequence Data with Recurrent Neural Networks, ICML 2006
12. **NAT with CTC** (Libovický et al., 2018)
    End-to-End Non-Autoregressive Neural Machine Translation with Connectionist Temporal Classification, EMNLP 2018