Neural Machine Translation with Non-Autoregressive Sequence Generation

CPSC 677 presentation

Linyong Nan

Sept 15, 2020

Introduction to MT

Source of image

Goal: translate a source text sequence **X** in one language into a target text sequence **Y** in another language.

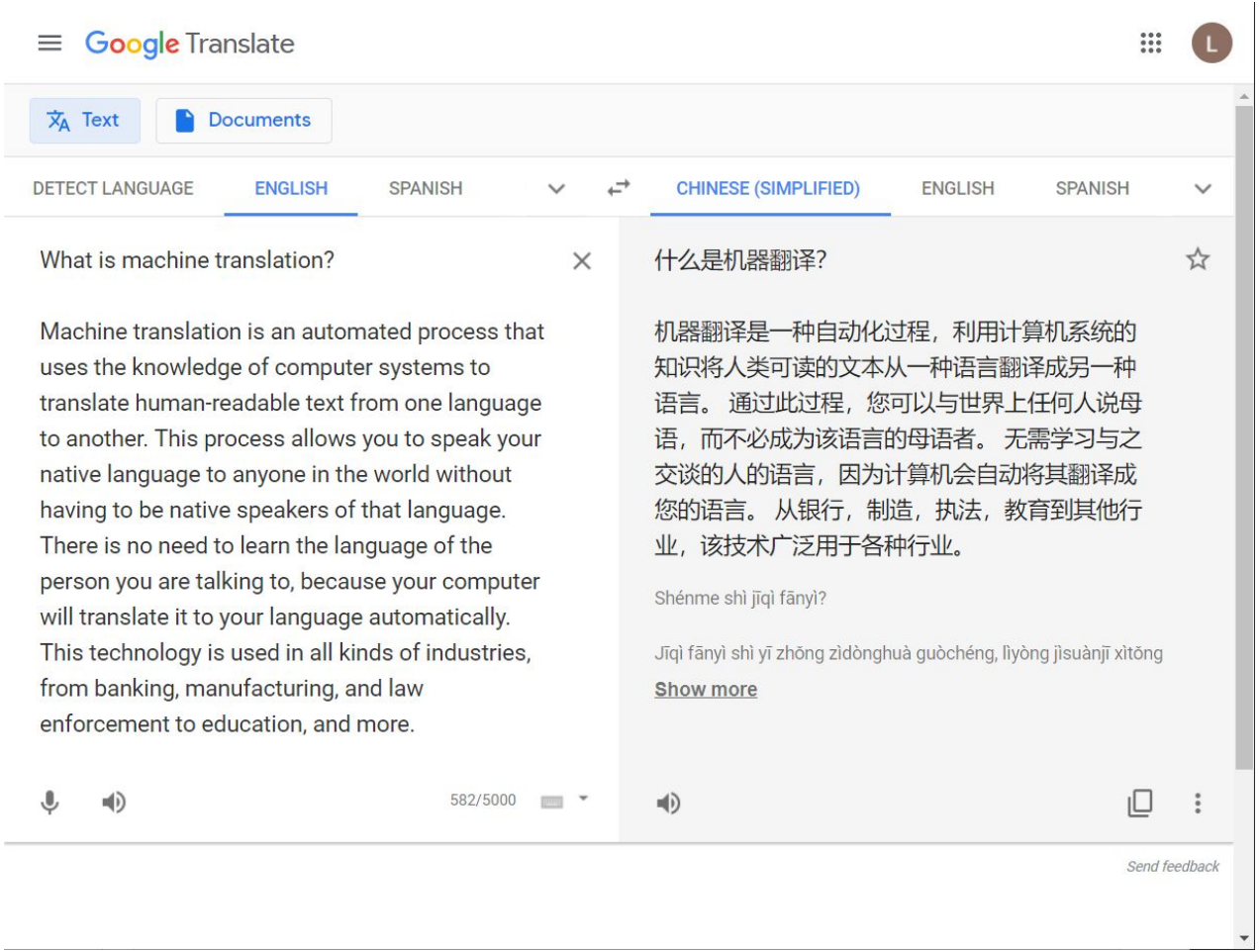
Example: say we have a source text sequence written by GPT-2...

What is machine translation?

Machine translation is an automated process that uses the knowledge of computer systems to translate human-readable text from one language to another. This process allows you to speak your native language to anyone in the world without having to be native speakers of that language . There is no need to learn the language of the person you are talking to , because your computer will translate it to your language automatically . This technology is used in all kinds of industries , from banking , manufacturing, and law enforcement to education, and more.

Written by Transformer · transformer.huggingface.co

Introduction to MT



History of MT

Source of image

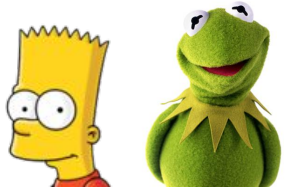
- 1950s: **Rule-based MT**
  - Bilingual dictionary and handcrafted rules
  - Georgetown-IBM demo: automatically translated 60 Russian sentences to English.
- 1990s: **Statistical MT**
  - Noisy-Channel approach: $Y^* = \underset{Y}{\operatorname{argmax}}\, p(Y|X) = \underset{Y}{\operatorname{argmax}}\, p(X|Y)p(Y)$
    - Translation model and language model
    - IBM models: use alignments in the translation model
  - Complicated, modularized, language dependent, heavy feature engineering
- 2010s: **Neural MT**
  - Single end-to-end neural network with large parallel corpus
  - Seq2seq with attention, RNN, Transformer

More recent

Source of image

- 2010-: Neural MT (covered in 2018 presentations)
  - Leverage large monolingual corpora, pre-training
    - Unsupervised MT
    - Semi-supervised MT
  - Multilingual MT
    - Want models that are language independent
    - Improve MT systems for low-resource languages
  - **Non-Autoregressive MT**
- Lots of challenges (from this review):
  - Document-level MT: not as good as translating a sentence
  - Low-resource MT: Neural methods work well only when lots of resources exist
  - Multimodal MT: translate with extra contexts (pragmatics, world knowledge, etc.)
  - **Non-autoregressive decoding**: slow inference

Text Sequence Modeling Formalism

- Directed
  - Unidirectional, Monotonic, Causal, Autoregressive (AR): N-gram LM, Neural LM
    - +large scale pre-training → GPT-n
    - Straightforward to sample from (can read and write)
  - Bidirectional: ELMo 
- Undirected
  - Autoencoding (AE): Masked LM (denoising AE) 
    - +large scale pre-training → BERT
  - Great job in NLU, but can we decode from it to improve NLG? (can it write better?)
- Combined
  - BART, UniLM, KERMIT, etc. (can read and write well) 
- XLM, XLNet, MASS, ERNIE, and more.

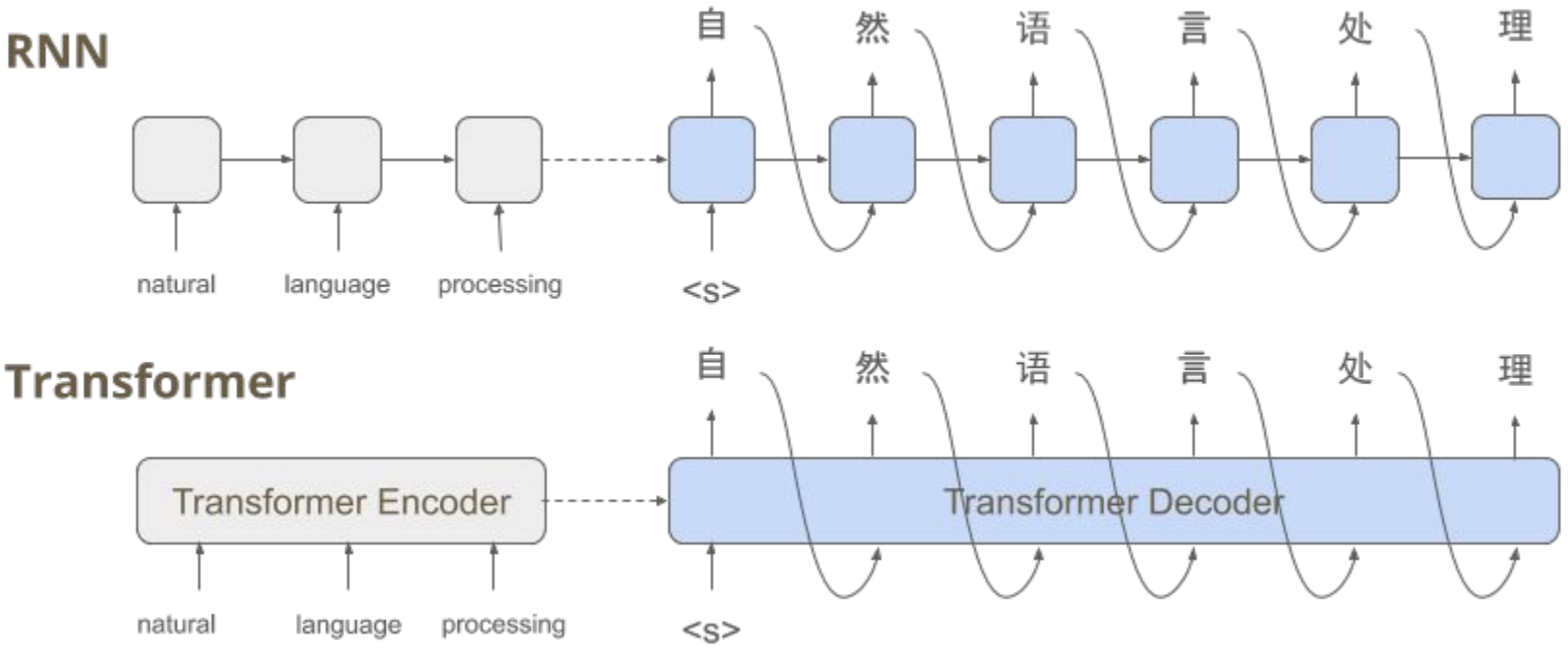
Machine translation as modeling text sequence pairs (X, Y)

Given source and target sentences: $X = (x_1, x_2, \ldots, x_{T'}), Y = (y_1, y_2, \ldots, y_T)$
Want to model: $p(Y|X)$

- Seq2Seq
- Concatenate [X, <SEP>, Y] and model it as a sequence
  - BERT (next sentence prediction)
  - XLM (translation LM)
  - KERMIT

Autoregressive Modeling

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

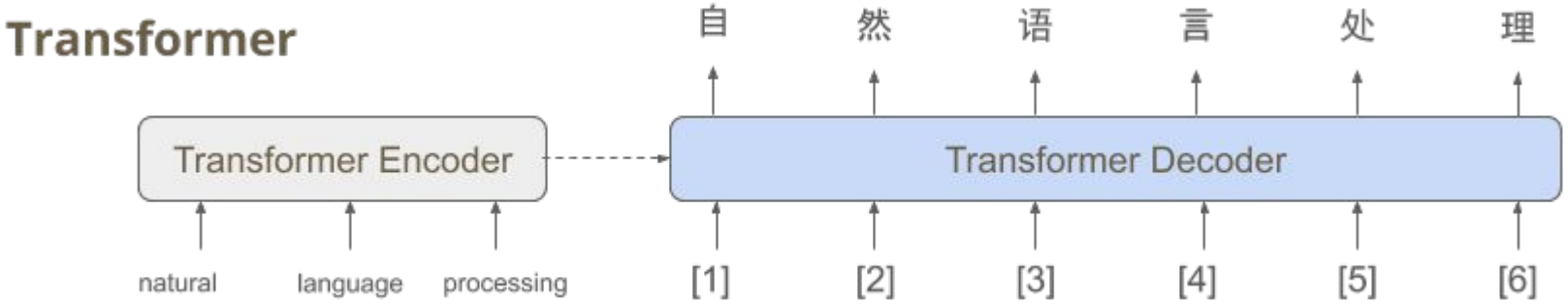


Decode in serial, but can train in parallel with teaching forcing

Source of image

Non-Autoregressive Modeling (Naive)

$$p_{\mathcal{NA}}(Y|X) = p_L(T|X) \cdot \prod_{t=1}^{T} p(y_t|X)$$



- Need to predict output length and feed position embedding
- Could potentially decode in parallel

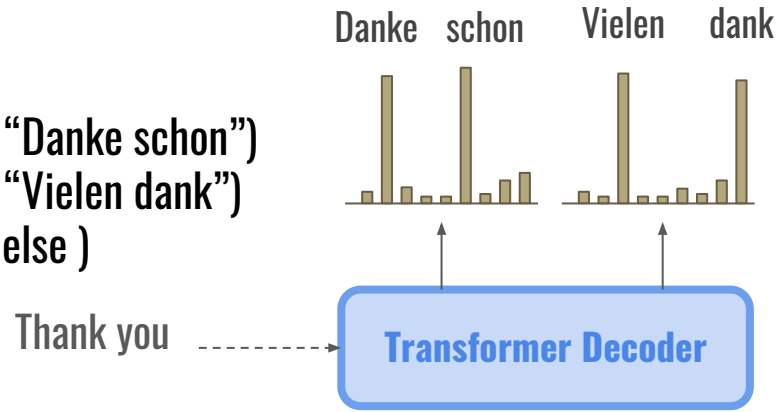
Source of image

Problems with non-autoregressive modeling

$$p_{\mathcal{NA}}(Y|X) = p_L(T|X) \cdot \prod_{t=1}^{T} p(y_t|X)$$

- **Multi-modality problem**: language is multimodal, there are multiple valid translations for a sentence.
- **Conditional independence assumption**: each y_t depends only on X, breaking the dependence across time (t=1,...,T)

In EN→DE dataset,
45% (“Thank you”, “Danke schon”)
45% (“Thank you”, “Vielen dank”)
10% (“Thank you”, else)



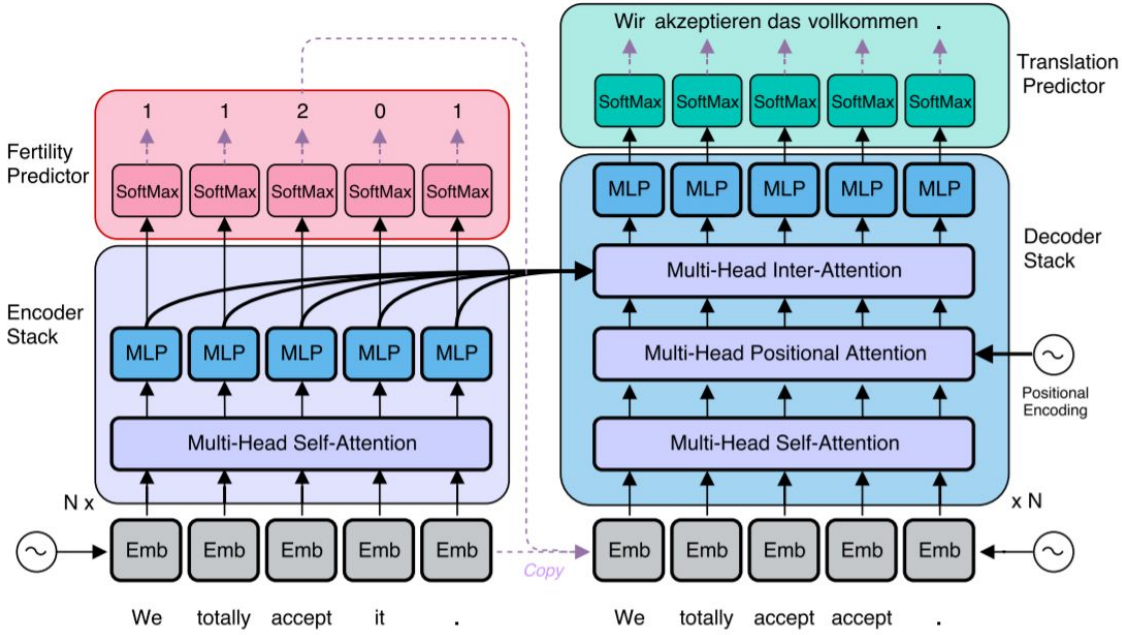
Danke schon ✓
Vielen dank ✓
Danke dank ✗
Vielen schon ✗

Source of image

Paper 1: Non-Autoregressive Neural Machine Translation, Gu et al., ICLR 2018

Vanilla Non-Autoregressive Translation

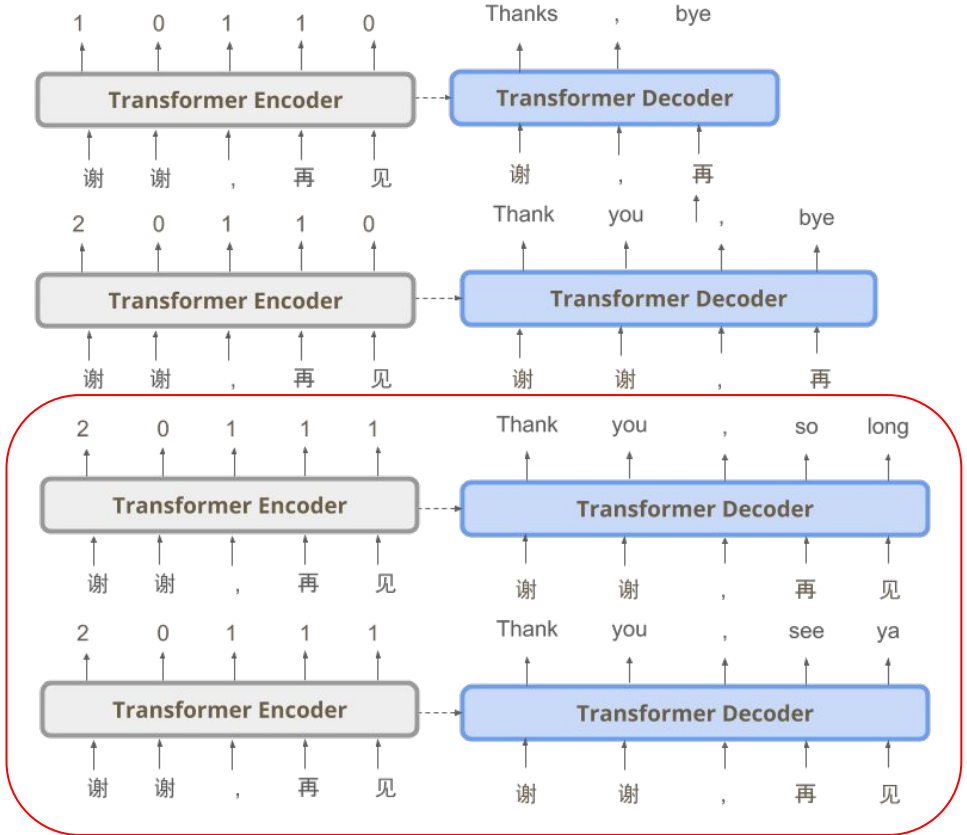
- Predict **fertility** as latent variable and copy inputs
  - Number of foreign words each native word produces
  - Sequence of integers with range [0, 50]
  - **[‘we’, ‘totally’, ‘accept’, ‘it’, ‘.’] → [1, 1, 2, 0, 1] → [‘we’, ‘totally’, ‘accept’, ‘accept’, ‘.’]**
  - represents sentence-level **plan/scaffold** that can be used to generate entire Y at once



Paper 1: Non-Autoregressive Neural Machine Translation, Gu et al., ICLR 2018

How does adding fertility help with solving multimodal problem?

- Provides natural factorization of the output space that drastically reduces the mode space.
- Global mode choice turns into local mode choices
- Don't need to explicitly model the length, which is the sum of the fertility sequence.
- Fixed targets: labels come from external aligner (fast-align)



Modified P(Y|X)

$$p_{\mathcal{NA}}(Y|X) = \sum_{f_1,\ldots,f_{T'} \in \mathcal{F}} \left(\prod_{t'=1}^{T'} p_{\mathcal{F}}(f_{t'}|X) \cdot \prod_{t=1}^{T} p(y_t|x_1\{f_1\},\ldots,x_{T'}\{f_{T'}\}) \right)$$

where $\mathcal{F} = \{f_1,\ldots,f_{T'}|\sum_{t'=1}^{T'} f_{t'} = T, f_{t'} \in \mathbb{Z}^*\}$ is the set of all fertility sequence, and $x\{f\}$ denotes the token x repeated f times.

Noisy Parallel Decoding (NPD)

1. Sample multiple fertility sequences
2. Compute the best translation for each fertility sequence
3. Identify the best one by scoring them with an autoregressive teacher

- For an autoregressive model, each sample takes **O(1)** to process during training (with teacher forcing) and **O(n)** during inference (n passes).
- Scoring a candidate translation takes **O(1)** time, and all candidate translations can be scored in parallel. This process only double the latency.

Training

$$\mathcal{L}_{\text{ML}} = \log p_{\mathcal{NA}}(Y|X;\theta) = \log \sum_{f_{1:T'} \in \mathcal{F}} p_F(f_{1:T'}|x_{1:T'};\theta) \cdot p(y_{1:T}|x_{1:T'}, f_{1:T'};\theta)$$

$$\geq \mathop{\mathbb{E}}_{f_{1:T'} \sim q} \left(\underbrace{\sum_{t=1}^{T} \log p(y_t|x_1\{f_1\}, .., x_{T'}\{f_{T'}\};\theta)}_{\text{Translation Loss}} + \underbrace{\sum_{t'=1}^{T'} \log p_F(f_{t'}|x_{1:T'};\theta)}_{\text{Fertility Loss}} \right) + \mathcal{H}(q)$$

- The proposal *q* can be defined by an 1. External, fixed fertility model, 2. Attention weights used in the fixed autoregressive teacher
- This loss allows us to train the translation model *p* and the fertility model *p_F* in a supervised fashion.

Sequence-level knowledge distillation

- [“Thank”, “you”, “.”] → [2, 0, 1] → [“Danke”, “schon”, “.”] or [“Vielen”, “Dank”, “.”]
- We still have some nondeterminism in the training data.

One solution: construct a new corpus that is less noisy

- Use autoregressive teacher’s greedy outputs to replace the human translations
- Train non-autoregressive student with teacher’s outputs as the targets
- This lowers the quality of the translations, but makes the student to consistently translate “Thank you.” to “Danke schon.” instead of “Danke dank.”

Fine-tuning

- Rely heavily on the outputs of the external alignment system, still want to train the entire model end-to-end
- After training NAT to convergence, fine-tune with additional loss that consists of the reverse KL divergence with the autoregressive teacher output distribution using REINFORCE (word-level knowledge distillation), together with the original sequence-level distillation loss.
- Not much boost in performance, as shown in ablation study.

Paper 1: Non-Autoregressive Neural Machine Translation, Gu et al., ICLR 2018

Experiments, Results, Ablation Study

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×

- Inference time X10 faster than autoregressive greedy decoding, or X15 faster than beam search with size 4
- Worse BLEU score in general (tradeoff)

Paper 1: Non-Autoregressive Neural Machine Translation, Gu et al., ICLR 2018

Experiments, Results, Ablation Study

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**

Different Approaches to NAT

- Vanilla NAT
  - **Gu et al., 2018[paper 1]**
- Iterative Refinement
  - Lee et al., **Ghazvininejad et al. (Mask-Predict)[paper 2]**, **Mansimov et al.[paper 3]**
- Insertion-based
  - Stern et al. (Insertion Transformer), Chan et al. (KERMIT)., **Chan et al.[paper 4]**
- Insert+Delete
  - Gu et al. (Levenshtein Transformer),
- CTC-based
  - Libovicky and Helcl, Saharia et al.
- Flow-based
  - Ma et al.
- Bidirectional Decoding
  - Zhou et al., Zhang et al.

NAT with Iterative Refinement

Multiple passes of decoding, each pass refines the previously generated seq

(could be one decoder doing multiple passes, or multiple one-pass decoders)

$$X \to Y_0 \to Y_1 \to Y_2 \to \cdots \to Y_T$$

Each sample takes constant time (T) to process, so **O(1)** time during inference

Early works that explore multiple passes of decoding using AR model:

- Deliberation Networks (Xia et al., NeurIPS 2017)
- (Will add more later...)

Paper 2: Mask-Predict: Parallel Decoding of Conditional Masked Language Models, Ghazvininejad et al., EMNLP 2019

Mask-Predict

- Conditional Masked Language Models (CMLM):
  - Transformer encoder + BERT as decoder
  - Predicts a set of target tokens Y_{mask} given a source X and remaining target tokens Y_{obs}
  - Tokens in Y_{mask} are conditionally independent of each other
- Predict target sequence length in encoder with [LENGTH]
  - Similar to [CLS] in BERT
- In each pass, refine words that are least confident
  - Mask tokens that were predicted with lowest probability in the lass past, predict again

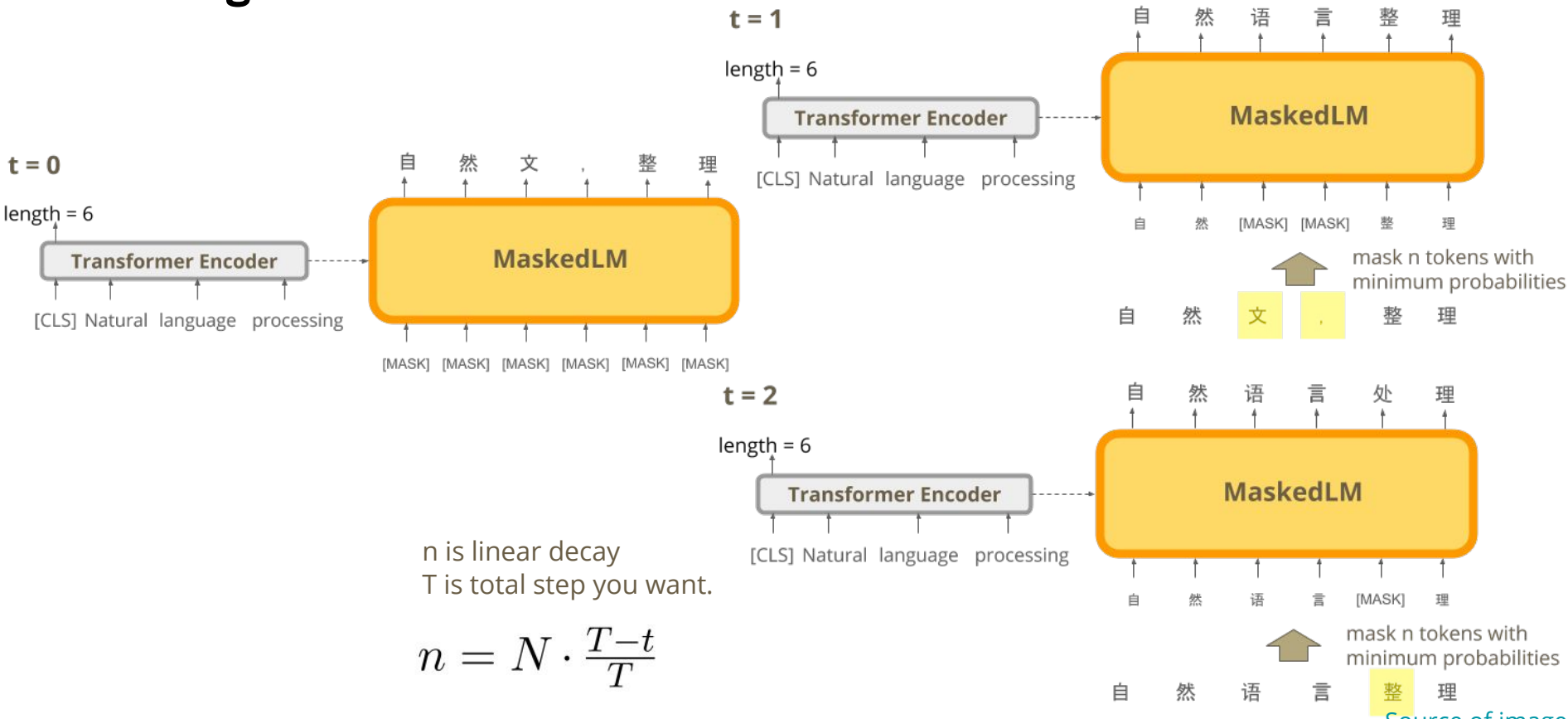
Paper 2: Mask-Predict: Parallel Decoding of Conditional Masked Language Models, Ghazvininejad et al., EMNLP 2019

Training

- Objective similar to BERT: cross-entropy loss over every token in Y_{mask}
- Length of inputs to the decoder is predicted by the encoder
- Sequence-level knowledge distillation, similar to paper 1

Paper 2: [Mask-Predict: Parallel Decoding of Conditional Masked Language Models](), Ghazvininejad et al., EMNLP 2019

Decoding: Mask-Predict



n is linear decay
T is total step you want.

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

Source of image

Paper 2: Mask-Predict: Parallel Decoding of Conditional Masked Language Models, Ghazvininejad et al., EMNLP 2019

Example of iterative refinement

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 .

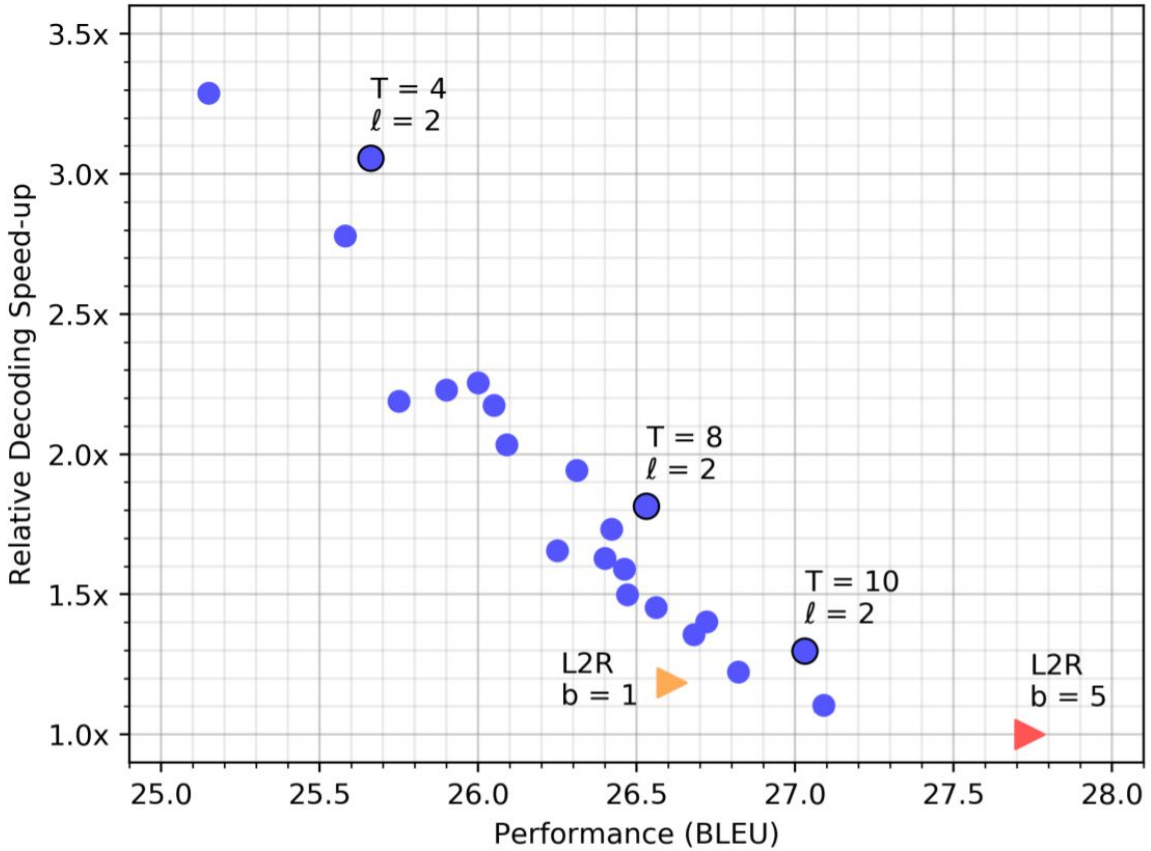
Paper 2: Mask-Predict: Parallel Decoding of Conditional Masked Language Models, Ghazvininejad et al., EMNLP 2019

Experiments and 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	— —	— —

Paper 2: [Mask-Predict: Parallel Decoding of Conditional Masked Language Models](), Ghazvininejad et al., EMNLP 2019

Trade-off between decoding speed and performance



Analysis

- Why multiple iterations?
  - To alleviate the multi-modal problem: condition only on parts of the input, reduce the mode space
- Do longer sequences need more iterations?
  - Yes, but with diminishing returns
- Do more length candidates help?
  - Not necessarily, adding too many can degrade performance
- Is distillation necessary?
  - Absolutely.

	$T = 4$	$T = 10$	$T = N$
$1 \leq N < 10$	21.8	22.4	22.4
$10 \leq N < 20$	24.6	25.9	26.0
$20 \leq N < 30$	24.9	26.7	27.1
$30 \leq N < 40$	24.9	26.7	27.6
$40 \leq N$	25.0	27.5	28.1

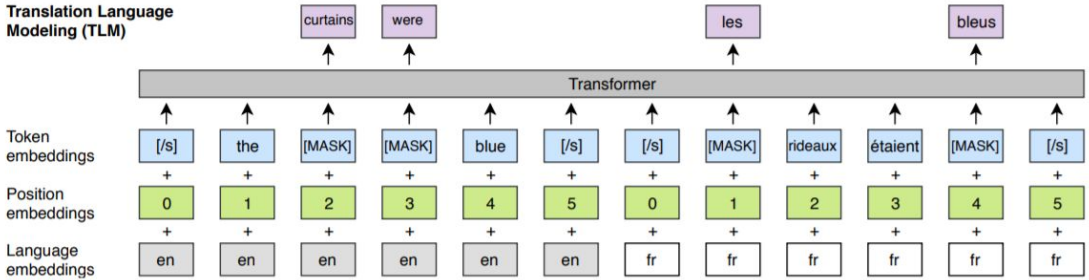
Iterations	WMT'14 EN-DE		WMT'16 EN-RO	
	Raw	Dist	Raw	Dist
$T = 1$	10.64	**18.05**	21.22	**27.32**
$T = 4$	22.25	**25.94**	31.40	**32.53**
$T = 10$	24.61	**27.03**	32.86	**33.08**

Table 6: The performance (BLEU) of base CMLM, trained with either raw data (Raw) or knowledge distillation from an autoregressive model (Dist).

Paper 3: A Generalized Framework of Sequence Generation with Application to Undirected Sequence Models, Mansimov et al., [ARXIV 2020]

General Framework of Generation from Masked LM

- Propose a framework that models the process of sequence generation
  - After determining the length of the sequence, iteratively alternating between:
    - Select sequence positions to mask
    - Generate new symbols for those masked positions
- Unify decoding process in directed and undirected models
  - BERT has a mouth and it can speak
  - Use Gibbs sampling to collect unbiased samples from BERT-like models
- Validate the proposal on MT using XLM with translation LM objective:
  - Start from a pre-trained XLM and fine-tune using translation LM objective



Paper 3: A Generalized Framework of Sequence Generation with Application to Undirected Sequence Models, Mansimov et al., [ARXIV 2020]

Decoding Framework

Generation sequence G contains T pairs of:

- an intermediate sequence $Y^t = (y_1^t, \ldots, y_L^t)$
- the coordinate sequence $Z^t = (z_1^t, \ldots, z_L^t)$

Where V is a vocabulary, L is the length of a sequence, T is the number of generation steps (number of refinements), $y_i^t \in V, z_i^t \in \{0, 1\}$

The coordinate seq indicates which symbol(s) of the current intermediate seq are to be replaced:

$$y_i^{t+1} = (1 - z_i^{t+1}) y_i^t + z_i^{t+1} \tilde{y}_i^{t+1}, \text{where } \tilde{y}_i^{t+1} \in V \text{ is a new symbol}, i \in [1, L]$$

Z as edit operations

Paper 3: A Generalized Framework of Sequence Generation with Application to Undirected Sequence Models, Mansimov et al., [ARXIV 2020]

Decoding Framework

$$G = ((Y^1, Z^1), (Y^2, Z^2), \ldots, (Y^T, Z^T))$$

$$Y^1 = (\langle mask \rangle, \ldots, \langle mask \rangle)$$

$$Z^1 = (0, \ldots, 0)$$

Iteratively fills/edits the tokens and terminates after T steps

$$P(G|X) = P(L|X) \prod_{t=1}^{T} \prod_{i=1}^{L} p(z_i^{t+1}|Y^{\leq t}, Z^t, X) p(y_i^{t+1}|Y^{\leq t}, X)^{z_i^{t+1}}$$

- Predict how long the target sentence will be (Length prediction)
- Select which symbols to be replaced to form Z^{t+1}(Coordinate selection)
  - Select which tokens to mask
- Select among target vocab what new symbols to use (Symbol selection), and replace with new symbol if the corresponding coordinate is 1 (Replacement)
  - Predict masked tokens

Paper 3: A Generalized Framework of Sequence Generation with Application to Undirected Sequence Models, Mansimov et al., [ARXIV 2020]

Special decoding cases under this framework

- Autoregressive sequence model

$$P(L|X) \propto \sum_{y_{1:L-1}} \prod_{l=1}^{L-1} p(y_{l+1}^{l+1} = \langle \text{eos} \rangle | y_{\leq l}^{\leq l}, X)$$

$$p(z_{i+1}^{t+1} = 1 | Y^{\leq t}, Z^t, X) = \mathbb{1}(z_i^t = 1), z_1^1 = 1$$

$$p(y_{i+1}^{t+1} | Y^{\leq t}, X) = p(y_{i+1}^{t+1} | y_1^t, y_2^t, , y_i^t, X), \text{ for } z_{i+1}^{t+1} = 1$$

$$P(G|X) = \prod_{i=1}^{L} p(y_i | y_{<i}, X)$$

- Non-Autoregressive sequence model by iterative refinement (Lee et al., 2018)
  - Length prediction and symbol replacement same as the original modeling
  - Coordinate selection: replace the symbols in all positions - $Z^t = (1, 1, \ldots, 1), \forall t$
- Semi-Autoregressive sequence model
  - Same length prediction construction

$$p(z_{k(i+1)+j}^{t+1} = 1 | Y^{\leq t}, Z^t, X) = \begin{cases} 1, & \text{if } z_{ki+j}^t = 1, \forall j \in \{0, 1, \ldots, k\} \\ 0, & \text{otherwise,} \end{cases}$$

$$p(y_{k(i+1)+j}^{t+1} | Y^{\leq t}, X) = p(y_{k(i+1)+j}^{t+1} | y_{<k(i+1)}^t, X), \quad \forall j \in \{0, 1, \ldots, k\},$$

for $z_i^t = 1$. This naturally implies that $T = \lceil L/k \rceil$.

Paper 3: A Generalized Framework of Sequence Generation with Application to Undirected Sequence Models, Mansimov et al., [ARXIV 2020]

Select coordinates to mask with different Gibbs sampling strategies

- Assume $P(L|X)$ is estimated from the training data
- 2 handcrafted Gibbs sampling strategies:
  - **Uniform** coordinate selection
  - Non-uniform coordinate selection
    - Select based on a log-linear model with the features obtained from previous text seq ar coordinate seq
    - Autoregressive: **Left2Right**
    - Paper 2: **Least2Most**
    - **Easy-first**: Refer to the paper
- Learned Gibbs sampling strategy:
  - Learn coordinate selection policy that maximizes a reward function

$$p(z_i^{t+1} = 1|Y^{\leq t}, Z^t, X) = 1/L$$

$$p(z_i^{t+1} = 1|Y^{\leq t}, Z^t, X) \propto \exp\left\{\frac{1}{\tau}\sum_{i=1}^{L}\alpha_i\phi_i(Y^t, Z^t, X, i)\right\}$$

Paper 3: A Generalized Framework of Sequence Generation with Application to Undirected Sequence Models, Mansimov et al., [ARXIV 2020]

Optimistic Decoding and Beam Search

- Naive approximation of $\text{argmax}_Y p(Y|X)$ is to marginalize out the generation procedure G using Monte Carlo method.
  - High variance and non-deterministic
- An optimistic assumption: distribution of the sequence generated by following the most likely generation path is close to the above distribution
  - No need to sample different generation paths
  - Still need to marginize out the generation path

$$\underset{\substack{L,Y^1,\ldots,Y^T \\ Z^1,\ldots,Z^T}}{\text{argmax}} \log p(L|X) + \sum_{t=1}^{T}\sum_{i=1}^{L}\left(\log p(z_i^{t+1}|Y^{\leq t}, Z^t, X) + z_i^{t+1}\log p(y_i^{t+1}|Y^{\leq t}, X)\right)$$

- Heuristic search: Length-conditioned beam search
  - Given length L, at each timestep t, generate and pick top-K scored hypotheses.
  - K hypotheses has K′ position candidates and K″ symbol replacement candidates$\rightarrow (K \times K' \times K'')$
  - From the expanded $(K \times K' \times K'')$ candidates, pick top-K candidates again by score.

Paper 3: A Generalized Framework of Sequence Generation with Application to Undirected Sequence Models, Mansimov et al., [ARXIV 2020]

Experiments and Results

- Linear time decoding **O(n)**: T set to L or 2L
- Constant time decoding **O(1)**: T constant w.r.t L
  - Number of tokens masked at t-th iteration is either constant $\lceil L/T \rceil$ or annealed from L to 1 (as in paper 2)

Paper 3: A Generalized Framework of Sequence Generation with Application to Undirected Sequence Models, Mansimov et al., [ARXIV 2020]

Linear-time decoding

	b	T	Baseline Autoregressive	Decoding from an undirected sequence model Uniform	Left2Right	Least2Most	Easy-First	Learned
En→De	1	L	25.33	21.01	24.27	23.08	23.73	24.10
	4	L	26.84	22.16	25.15	23.81	24.13	24.87
	4	L*	–	22.74	**25.66**	24.42	24.69	**25.28**
	1	$2L$	–	21.16	24.45	23.32	23.87	24.15
	4	$2L$	–	21.99	25.14	23.81	24.14	24.86
De→En	1	L	29.83	26.01	28.34	28.85	29.00	28.47
	4	L	30.92	27.07	29.52	29.03	29.41	29.73
	4	L*	–	28.07	**30.46**	29.84	30.32	**30.58**
	1	$2L$	–	26.24	28.64	28.60	29.12	28.45
	4	$2L$	–	26.98	29.50	29.02	29.41	29.71

Table 1. Results (BLEU↑) on WMT'14 En↔De translation using various decoding algorithms and different settings of beam search width (b) and number of iterations (T) as a function of sentence length (L). For each sentence we use 4 most likely sentence lengths. * denotes rescoring generated hypotheses using autoregressive model instead of proposed model.

Paper 3: [A Generalized Framework of Sequence Generation with Application to Undirected Sequence Models](), Mansimov et al., [ARXIV 2020]

Generation Orders

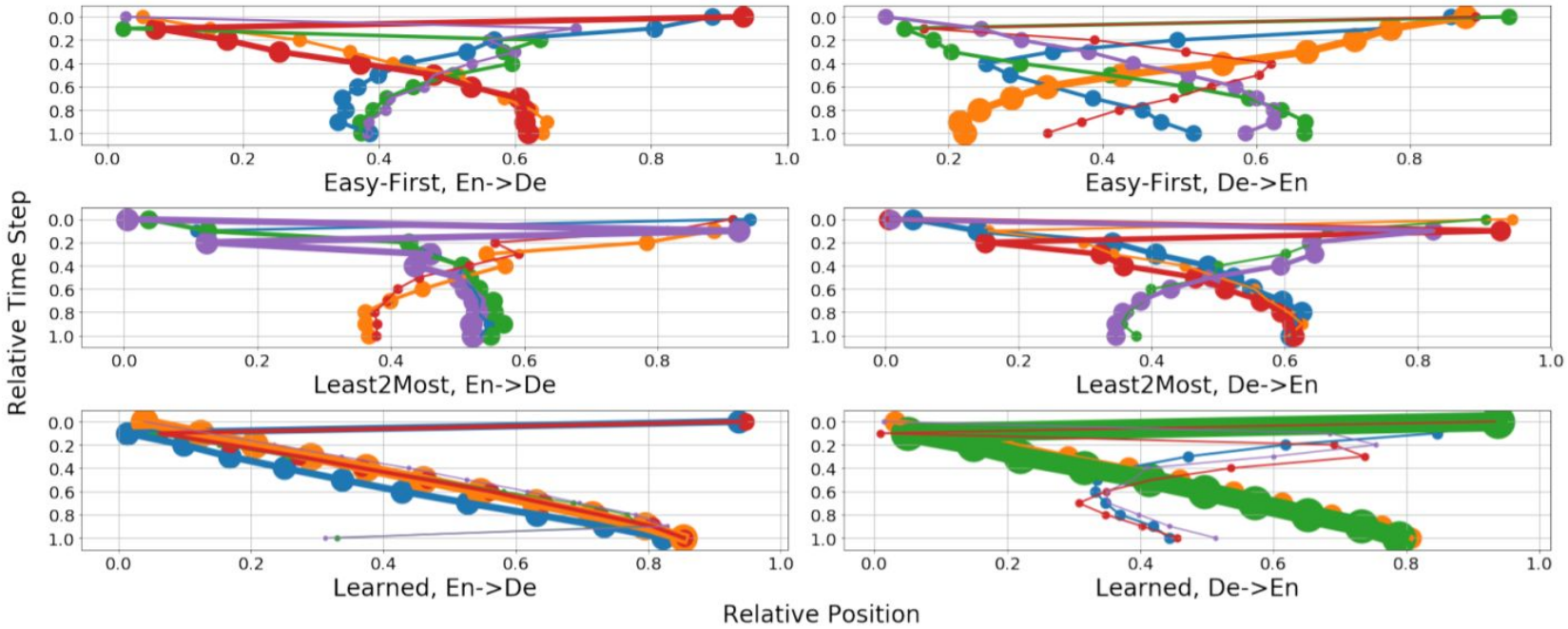


Figure 1. Generation orders given by **easy-first**, **least2most**, and **learned** coordinate selection. We use greedy search with L iterations on the development set. We group the orders into five clusters using and visualize cluster centers with normalized positions (x-axis) over normalized generation steps (y-axis). The thickness of a line is proportional to the number of examples in the corresponding cluster.

Paper 3: A Generalized Framework of Sequence Generation with Application to Undirected Sequence Models, Mansimov et al., [ARXIV 2020]

Constant-time decoding

T	o_t	Uniform	Left2Right	Least2Most	Easy-First	Hard-First	Learned
10	$L \rightarrow 1$	22.38	22.38	27.14	22.21	26.66	12.70
10	$L \rightarrow 1$*	23.64	23.64	**28.63**	23.79	**28.46**	13.18
10	$\lceil L/T \rceil$	22.43	21.92	24.69	25.16	23.46	26.47
20	$L \rightarrow 1$	26.01	26.01	28.54	22.24	28.32	12.85
20	$L \rightarrow 1$*	27.28	27.28	**30.13**	24.55	**29.82**	13.19
20	$\lceil L/T \rceil$	24.69	25.94	27.01	27.49	25.56	27.82

Table 3. Constant-time machine translation on WMT'14 De→En with different settings of the budget (T) and number of tokens predicted each iteration (o_t). * denotes rescoring generated hypotheses using autoregressive model instead of proposed model.

Paper 3: A Generalized Framework of Sequence Generation with Application to Undirected Sequence Models, Mansimov et al., [ARXIV 2020]

Analysis

- Non-uniform coordinate selection strategies are better than uniform (left2right, least2most, easy-first, learned)
- Beam search is beneficial
- Knowledge distillation from the autoregressive teacher is helpful
- Little improvement in refining a sequence beyond the first pass
- Adaptive generation order strategies (least2most, easy-first, learned) tends to generate in monotonic order (left2right, right2left) or outside-in.
- Found no other interesting orders

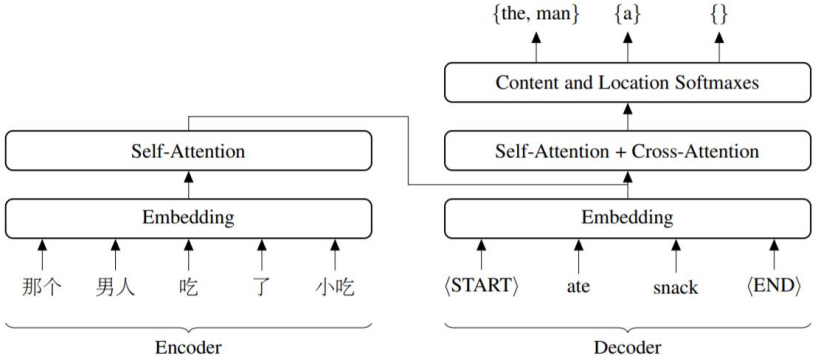
Paper 4: An Empirical Study of Generation Order for Machine Translation, Chan et al., [ARXIV 2019]

Insertion Transformer

- Seq2seq model whose output is formed by successively inserting one or more tokens at arbitrary locations into a partial hypothesis
- Joint distribution over tokens and slots

$$p(c, l \mid x, \hat{y}_t) = \text{InsertionTransformer}(x, \hat{y}_t),$$

where $c \in V$ is the content being selected from the vocabulary V and $0 \leq l \leq |\hat{y}_t|$ is the insertion location.



- Removes causal attention mask in the decoder
- Pads decoder input on both ends (length n+2), then concatenates adjacent pairs of output vectors to obtain n+1 slot representations.

Insertion Transformer

Modeling $p(c, l) = p(c|l)p(l)$:

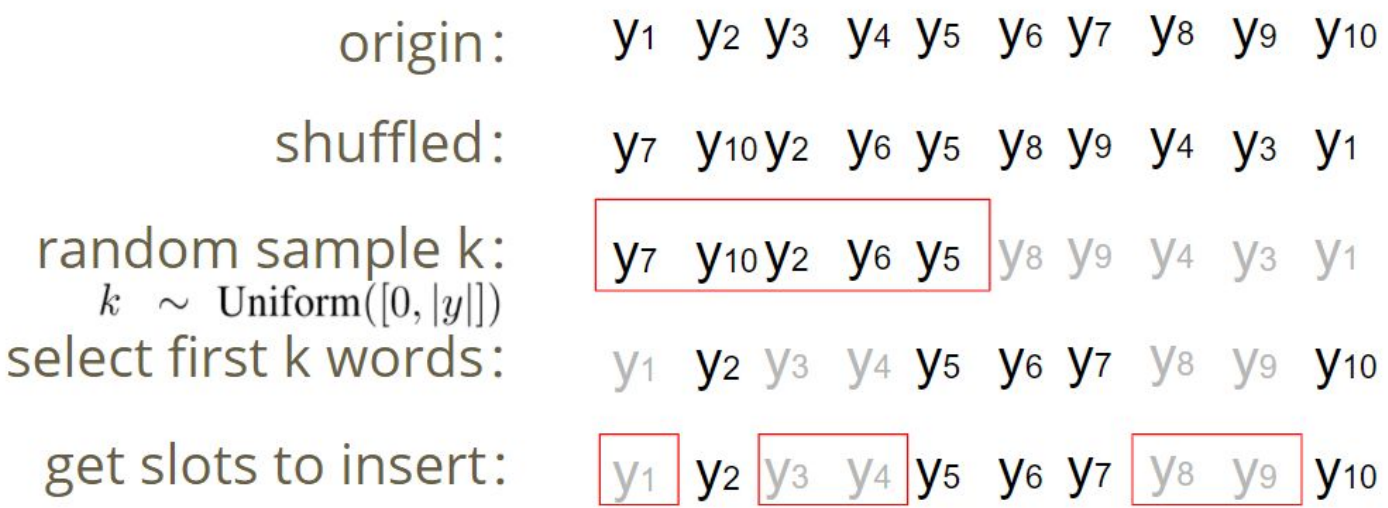
- matrix of slot representations $H \in \mathbb{R}^{(T+1)\times h}$, h is hidden size and T is the length of the current partial hypothesis.
- Transformer softmax projection matrix: $W \in \mathbb{R}^{h\times|\mathcal{C}|}$

$$p(c \mid l) = \text{softmax}(h_l W) \qquad p(l) = \text{softmax}(Hq)$$

- $h_l \in \mathbb{R}^h$ is the L-th row of H, $q \in \mathbb{R}^h$ is a learnable query vector.

Roll-in Policy

- Sample lots of partial hypotheses to form training examples
  - Increase diversity
  - Reduce exposure bias
  - Biased sampling procedure that gives uniform treatment to all lengths:



Source of image

Paper 4: An Empirical Study of Generation Order for Machine Translation, Chan et al., [ARXIV 2019]

Insertion Transformer - decoding

Serial generation:

Hypothesis	Insertion
[]	(ate, 0)
[ate]	(snack, 1)
[ate, snack]	(man, 0)
[man, ate, snack]	(the, 0)
[the, man, ate, snack]	(a, 3)
[the, man, ate, a, snack]	(⟨EOS⟩, 5)

Parallel generation:

Hypothesis	Insertions
[]	(ate, 0)
[ate]	(man, 0), (snack, 1)
[man, ate, snack]	(the, 0), (a, 2)
[the, man, ate, a, snack]	(⟨EOS⟩, 5)

Figure 2: Example decoding paths for serial and parallel generation using the Insertion Transformer.

$$(\hat{c}_t, \hat{l}_t) = \underset{c,l}{\operatorname{argmax}}\, p(c, l \mid x, \hat{y}_t)$$

Highest-scoring (tokens, location)

$$\hat{c}_{l,t} = \underset{c}{\operatorname{argmax}}\, p(c \mid l, x, \hat{y}_t)$$

Top-k highest-scoring tokens at corresponding locations

Paper 4: [An Empirical Study of Generation Order for Machine Translation](), Chan et al., [ARXIV 2019]

Order-reward Framework

$$R(a) = \begin{cases} -O(a) & \forall a \in A^* \\ -\infty & \forall a \notin A^* \end{cases}$$

$$q_{\text{oracle}}(a) = \frac{\exp(R(a)/\tau)}{\sum_{a' \in A^*} \exp(R(a')/\tau)}$$

$$\mathcal{L} = \text{KL}(q_{\text{oracle}} \,\|\, p)$$

This loss construction allows training seq2seq under any oracle policy (can induce any insertion order)

Main idea: design order functions to induce preferences of insertion order

- Action ***a***: [word ***w***, slot ***s*** within ***span(i,j)***]
- ***O(a)***: mapping actions to scores
- ***R(a)***: reward function
- $q_{\text{oracle}}(a)$: probability that the oracle policy assigns to *a*
- A^*: set of all valid actions
- $\tau \in (0, \infty)$: temperature controlling the sharpness of the distribution
- ***p(a)***: probability that model assigns to *a*
- slot loss ***L***: KL-divergence between oracle and model distributions
- Full loss: average of slot losses across all slots

Paper 4: An Empirical Study of Generation Order for Machine Translation, Chan et al., [ARXIV 2019]

Order Functions

Order	**Order Function** $O(a)$
Uniform	0
Balanced Binary Tree	$\lvert s - (i+j)/2 \rvert$
Random	$\text{rank}(\text{hash}(w))$
Sequential (L2R vs. R2L)	$\pm s$
Frequency (Common vs. Rare)	$\pm\text{rank}(\text{frequency}(w))$
Length (Short vs. Long)	$\pm\text{rank}(\text{length}(w))$
Alphabetical (A $\rightarrow$ z vs. z $\rightarrow$ A)	$\pm\text{rank}(w)$
Adaptive (Easy vs. Hard)	$\pm \log p(a)$

Table 1: Order functions for an action a corresponding to the insertion of word w into slot s within span (i, j). The rank terms are computed with respect to the set of words from the valid action set A^*.

- **Uninformed** (uniform, random)
- **Location-based** (sequential, balanced binary tree)
- **Frequency-based**
- **Content-based** (Length, Alphabetical - based on Unicode)
- **Model-based** (Adaptive)
  - Easy-first: pushes model's posterior to where it already has high probability mass
  - Hard-first: opposite of easy-first

Paper 4: An Empirical Study of Generation Order for Machine Translation, Chan et al., [ARXIV 2019]

Experiments and Results

Order	En → De			En → Zh		
τ	0.5	1.0	2.0	0.5	1.0	2.0
Binary Tree	91%	86%	80%	88%	83%	78%
Random	86%	81%	72%	82%	77%	68%
Left-to-Right	95%	88%	77%	88%	82%	70%
Right-to-Left	95%	90%	78%	92%	83%	72%
Common First	92%	88%	80%	88%	84%	76%
Rare First	88%	81%	73%	83%	77%	67%
Shortest First	93%	88%	80%	91%	84%	76%
Longest First	92%	86%	77%	92%	84%	76%
Alphabetical (A → z)	93%	87%	77%	88%	82%	73%
Alphabetical (z → A)	90%	84%	74%	85%	78%	69%

Table 2: Percentage of insertions that follow the target order exactly, averaged over the development set.

Capable of learning the oracle policy order preference

Paper 4: An Empirical Study of Generation Order for Machine Translation, Chan et al., [ARXIV 2019]

Experiments and Results

Order	En → De		En → Zh	
	Serial	**Parallel**	**Serial**	**Parallel**
	Vaswani et al. (2017)		This Work	
Transformer	27.3		35.8	
	Stern et al. (2019)		This Work	
Uniform	27.12	26.72	32.9	33.1
Binary Tree	27.29	27.41	32.6	34.0
	This Work			
Random	26.15	26.10	32.6	32.4
Left-to-Right	26.37	25.56	31.7	31.2
Right-to-Left	26.60	24.49	32.4	30.8
Common First	26.88	26.86	33.5	32.9
Rare First	26.06	26.24	32.5	32.2
Shortest First	27.05	27.15	33.0	32.7
Longest First	26.45	26.41	32.8	33.2
Alphabetical (A → z)	26.86	26.58	32.7	32.5
Alphabetical (z → A)	27.22	26.37	33.1	33.0
Easy First	26.95	27.05	32.5	32.5
Hard First	25.85	26.30	32.4	32.9

Paper 4: An Empirical Study of Generation Order for Machine Translation, Chan et al., [ARXIV 2019]

Decoding Examples

Input: It would of course be a little simpler for the Germans if there were a coherent and standardised European policy, which is currently not the case.

Output: Es wäre für die Deutschen natürlich ein wenig einfacher, wenn es eine kohärente und einheitliche europäische Politik gäbe, was derzeit nicht der Fall ist.

Parallel decode (alphabetical):

Es_ wäre_ für_ die_ Deutschen_ natürlich_ ein_ wenig_ einfacher_ ._ wenn_ es_ eine_ kohärent e_ und_ einheitliche_ europäische_ Politik_ gäbe_ ,_ was_ derzeit_ nicht_ der_ Fall_ ist_ ._
Es_ wäre_ für_ die_ Deutschen_ natürlich_ ein_ wenig_ einfacher_ ,_ wenn_ es_ eine_ kohärent e_ und_ einheitliche_ europäische_ Politik_ gäbe_ ,_ was_ derzeit_ nicht_ der_ Fall_ ist_ ._
Es_ wäre_ für_ die_ Deutschen_ natürlich_ ein_ wenig_ einfacher_ ,_ wenn_ es_ eine_ kohärent e_ und_ einheitliche_ europäische_ Politik_ gäbe_ ,_ was_ derzeit_ nicht_ der_ Fall_ ist_ ._
Es_ wäre_ für_ die_ Deutschen_ natürlich_ ein_ wenig_ einfacher_ ,_ wenn_ es_ eine_ kohärent e_ und_ einheitliche_ europäische_ Politik_ gäbe_ ,_ was_ derzeit_ nicht_ der_ Fall_ ist_ ._
Es_ wäre_ für_ die_ Deutschen_ natürlich_ ein_ wenig_ einfacher_ ,_ wenn_ es_ eine_ kohärent e_ und_ einheitliche_ europäische_ Politik_ gäbe_ ,_ was_ derzeit_ nicht_ der_ Fall_ ist_ ._
Es_ wäre_ für_ die_ Deutschen_ natürlich_ ein_ wenig_ einfacher_ ,_ wenn_ es_ eine_ kohärent e_ und_ einheitliche_ europäische_ Politik_ gäbe_ ,_ was_ derzeit_ nicht_ der_ Fall_ ist_ ._
Es_ wäre_ für_ die_ Deutschen_ natürlich_ ein_ wenig_ einfacher_ ,_ wenn_ es_ eine_ kohärent e_ und_ einheitliche_ europäische_ Politik_ gäbe_ ,_ was_ derzeit_ nicht_ der_ Fall_ ist_ ._

Input: according to the data of National Bureau of Statistics , the fixed asset investment growth , total imports and other data in July have come down .

Output: 根据国家统计局的数据，7月份的固定资产投资增长、进口总额和其他数据有所下降。

Parallel decode (alphabetical):

根据_国家 统 计 局_的_ 数据_ ，_ 7_月份_的_固 定 资产_投资_增长_ 、_ 进口_总额_和_其他_数据_有所_下降_ 。_
根据_国家 统 计 局_的_ 数据_ 、_ 7_月份_的_固 定 资产_投资_增长_ 、_ 进口_总额_和_其他_数据_有所_下降_ 。_
根据_国家 统 计 局_的_ 数据_ ，_ 7_月份_的_固 定 资产_投资_增长_ 、_ 进口_总额_和_其他_数据_有所_下降_ 。_
根据_国家 统 计 局_的_ 数据_ ，_ 7_月份_的_固 定 资产_投资_增长_ 、_ 进口_总额_和_其他_数据_有所_下降_ 。_
根据_国家 统 计 局_的_ 数据_ ，_ 7_月份_的_固 定 资产_投资_增长_ 、_ 进口_总额_和_其他_数据_有所_下降_ 。_
根据_国家 统 计 局_的_ 数据_ ，_ 7_月份_的_固 定 资产_投资_增长_ 、_ 进口_总额_和_其他_数据_有所_下降_ 。_
根据_国家 统 计 局_的_ 数据_ ，_ 7_月份_的_固 定 资产_投资_增长_ 、_ 进口_总额_和_其他_数据_有所_下降_ 。_
根据_国家 统 计 局_的_ 数据_ ，_ 7_月份_的_固 定 资产_投资_增长_ 、_ 进口_总额_和_其他_数据_有所_下降_ 。_

Figure 3: Example decodes for models trained to generate tokens in alphabetical (Unicode) order. Blue tokens correspond those being inserted at the current time step, and gray tokens correspond to those not yet generated. Note that the desired ordering applies on a per-slot basis rather than a global basis.

Decoding Examples

Input: It will be sung by all the artists at all the three concerts at the same time.

Output: Es wird von allen Künstlern bei allen drei Konzerten gleichzeitig gesungen.

Parallel decode (longest-first):

Es_ wird_ von_ allen_ Künstler n_ bei_ allen_ drei_ Konzert en_ gleichzeitig_ ges ungen_ ._
Es_ wird_ von_ allen_ Künstler n_ bei_ allen_ drei_ Konzert en_ gleichzeitig_ ges ungen_ ._
Es_ wird_ von_ allen_ Künstler n_ bei_ allen_ drei_ Konzert en_ gleichzeitig_ ges ungen_ ._
Es_ wird_ von_ allen_ Künstler n_ bei_ allen_ drei_ Konzert en_ gleichzeitig_ ges ungen_ ._
Es_ wird_ von_ allen_ Künstler n_ bei_ allen_ drei_ Konzert en_ gleichzeitig_ ges ungen_ ._
Es_ wird_ von_ allen_ Künstler n_ bei_ allen_ drei_ Konzert en_ gleichzeitig_ ges ungen_ ._

Figure 4: An example of longest-first generation.

Input: imagine eating enough peanuts to serve as your dinner .

Output: 想象一下，吃足够的花生作为你的晚餐。

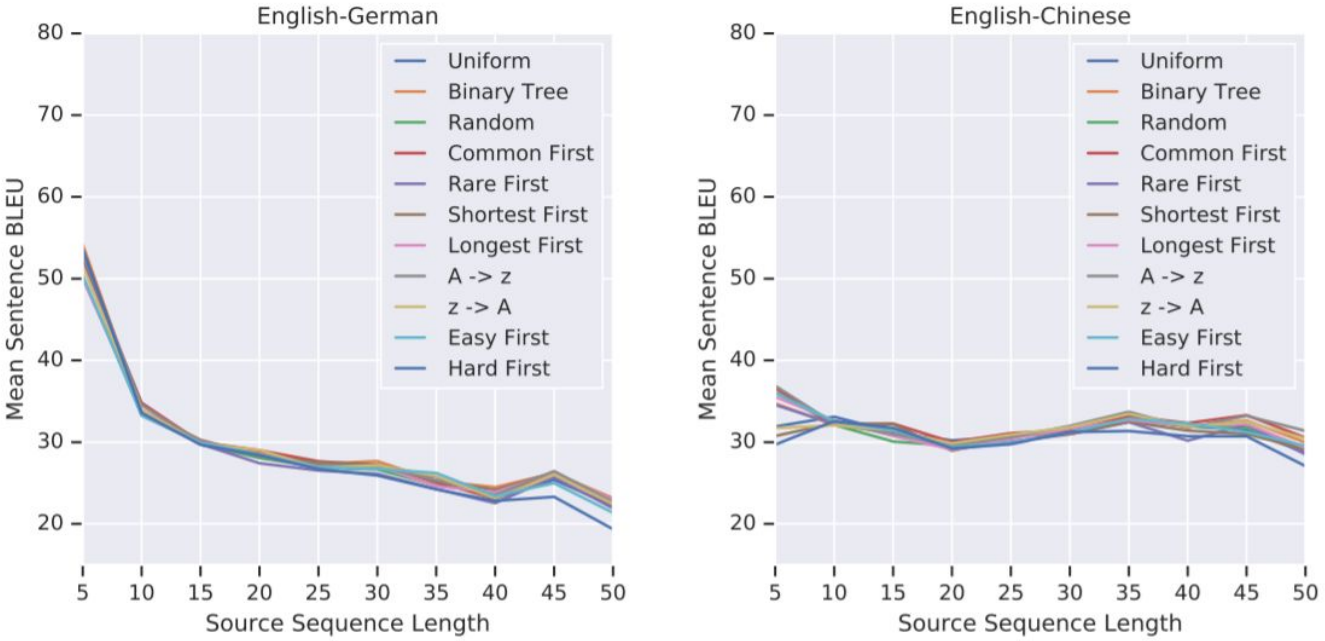
Parallel decode (common-first):

想象_ 一下_ ，_ 吃_ 足够_ 的_ 花 生_ 作为_ 你_ 的_ 晚餐_ 。_
想象_ 一下_ ，_ 吃_ 足够_ 的_ 花 生_ 作为_ 你_ 的_ 晚餐_ 。_
想象_ 一下_ ，_ 吃_ 足够_ 的_ 花 生_ 作为_ 你_ 的_ 晚餐_ 。_
想象_ 一下_ ，_ 吃_ 足够_ 的_ 花 生_ 作为_ 你_ 的_ 晚餐_ 。_
想象_ 一下_ ，_ 吃_ 足够_ 的_ 花 生_ 作为_ 你_ 的_ 晚餐_ 。_
想象_ 一下_ ，_ 吃_ 足够_ 的_ 花 生_ 作为_ 你_ 的_ 晚餐_ 。_

Figure 5: An example of common-first generation.

Paper 4: An Empirical Study of Generation Order for Machine Translation, Chan et al., [ARXIV 2019]

Performance vs. Sentence Length vs. Generation Order



(a) English → German

(b) English → Chinese

Figure 6: Sentence-level BLEU scores as a function of sentence length for several of our model variants. Source sentences in each development set are binned into groups of size 5, up to length 50.

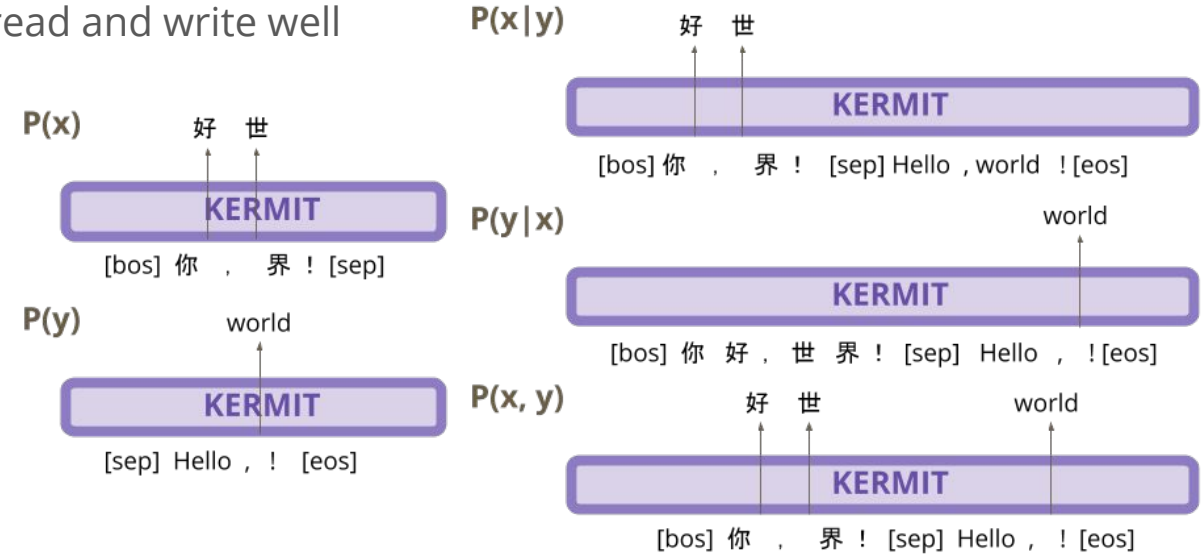
Analysis

- No generation ordering is notably better or worse than others (even random order is not far behind)
  - Binary-tree order achieves highest BLEU across all orders with parallel decoding
  - Possible to use non-AR order to achieve similar BLEU
- AR Transformer and Insertion Transformer have a large BLEU gap for English-Chinese translation task (AR Transformer better)
  - Larger discrepancy between word orders comparing to English-German
  - What language pairs to include matters when studying generation order
- Small model performance variance across all sentence lengths
  - No ordering is notably better or worse when dealing with long/difficult sentences
  - One exception: hard-first ordering is worse than others when generating long sentences

KERMIT: Kontextuell Encoder Representations Made by Insertion Transformations

$$p(x, y) \qquad p(x), p(y) \qquad p(y|x), p(x|y)$$

- Models the **joint distribution**, **marginals** and **conditionals** using a single neural network, and does not rely on a prespecified factorization of the data distribution
- Can also sample from the joint distribution or the marginals
- Supports serial AR decoding and parallel decoding
- Can read and write well



Only contains insertion-transformer decoder stack without causal masking of the self-attention sub-layer

Source of image

KERMIT: Kontextuell Encoder Representations Made by Insertion Transformations

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

Table 3: GLUE benchmark scores (as computed by the GLUE evaluation server). Of these models, only GPT and KERMIT admit a straightforward generation process.

NLU as good as BERT

KERMIT: Kontextuell Encoder Representations Made by Insertion Transformations

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	$\approx \log_2 n \ll 10$

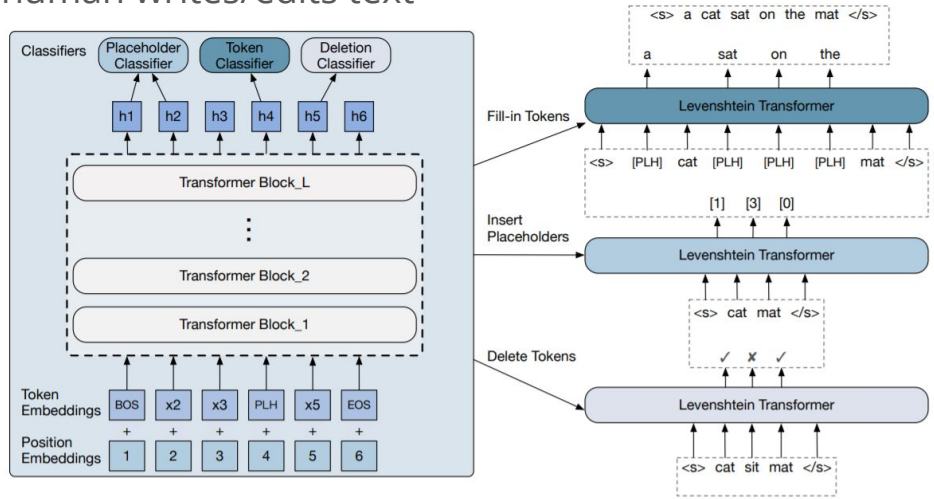
Table 2: WMT English ↔ German newstest2014 BLEU. Models capable of translating in both directions are marked with ↔.

KERMIT can do zero-shot cloze QA as well. There's also Multilingual-KERMIT

Write (translate) faster and competitively comparing to AR Transformer

Insert+Delete

- Levenshtein Transformer
  - Modeling the sequence generation and refinement problem with a Markov Decision Process (MDP), reward function measures the Levenshtein distance between generated sequence and target sequence.
  - Train with imitation Learning (check out this NeurIPS 2019 tutorial and this paper)
  - Similar to how human writes/edits text



Other approaches

- Connectionist Temporal Classification (CTC)
  - Non-autoregressive model for speech recognition
- Imputer (CTC+Mask-Predict)
- Flow-based
- Blockwise-decoding
- Look-around-decoding
- Energy-based inference networks
- DisCo
- General decoding with target bidirectional context:
  - Synchronous, Asynchronous, past and future, dynamic past and future

Discussion Questions

1. How can we bridge the gap between training and inference in both iterative-based and insertion-based frameworks? (How do we reduce the exposure bias in NAT?)
    - During inference time, the model needs to predict a series of (intermediate, coordinate) pairs or (word, slot) pairs to generate the full sentences, while the model is trained with samples that are intermediate sequence (randomly masked) or partial hypothesis sampled from the parallel sentences.
    - Such gap/bias exists in AR decoding scheme as well: Teaching forcing while training.
    - What other sampling procedure for training might close this gap?
2. Besides fertility, what other concepts/structures can make for reasonable/interesting latent variables that can be used to guide Non-AR MT or generation in general?
    - Gu et al.'s criteria: simple to infer, should account for correlations across time (positions) of the outputs
    - Should help resolving multi-modal problem, and preferably provides a natural way of determining the output length

Discussion Questions

3. NAT with fertility, iterative or insertion-based methods are all trying to inject some conditional dependencies across the output sequence. Can you think of a new way to inject such component in the model? What can we learn from structured prediction?
4. What generation order/strategy do you think is best to describe human language production behavior? Do you think this strategy is language-dependent?
5. Any thoughts on how we can expand this non-autoregressive approaches to general conditional text generation with, say images or videos?

References

- Papers
  - Non-Autoregressive Neural Machine Translation
  - Mask-Predict: Parallel Decoding of Conditional Masked Language Models
  - A Generalized Framework of Sequence Generation with Application to Undirected Sequence Models
  - An Empirical Study of Generation Order for Machine Translation
  - Lee et al.
  - Stern et al.
  - KERMIT
  - Levenshtein Transformer
  - Libovicky and Helcl.
  - Saharia et al.
  - Ma et al.
  - Blockwise-decoding
  - Look-around-decoding
  - Energy-based inference networks
  - DisCo
  - Synchronous
  - Asynchronous
  - past and future
  - dynamic past and future
  - GPT-n, ELMo, BART, UniLM, BERT, XLM, XLNet, MASS, ERNIE
  - 2020 NMT review
  - Deliberation Networks
  - exposure bias

References

- Images
  - https://en.wikipedia.org/wiki/Tower_of_Babel#/media/File:Pieter_Bruegel_the_Elder_-_The_Tower_of_Babel_(Vienna)_-_Google_Art_Project.jpg
  - https://www.aspenideas.org/speakers/elmo
  - https://mlexplained.com/2019/01/07/paper-dissected-bert-pre-training-of-deep-bidirectional-transformers-for-language-understanding-explained/
  - https://en.wikipedia.org/wiki/Kermit_the_Frog#/media/File:Kermit_the_Frog.jpg
  - https://simpsons.fandom.com/wiki/Bart_Simpson
  - https://transformer.huggingface.co
- Miscellaneous
  - https://www.youtube.com/embed/jvyKmU4OM3c (Lecture)
  - https://docs.google.com/presentation/d/1lnXSxPL3hZQ_OHyV_Lksz3UcOmPyiDqC1eg29Q5lAr0/edit#slide=id.g6db193174d_1_0 (Slides)
  - https://homes.cs.washington.edu/~jkasai/2020-01-28/nat/ (Blog)
  - https://jlibovicky.github.io/2019/07/04/MT-Weekly-Generalized-Framework-for-Decoding.html (Blog)
  - https://andre-martins.github.io/docs/dsl2018/lecture_09.pdf (Slides)
  - http://www.cs.columbia.edu/~mcollins/courses/nlp2011/notes/ibm12.pdf (Lecture Notes)
  - https://transformer.huggingface.co/doc/gpt2-large (GPT-2 Writer)
  - https://github.com/brucepang/CPSC677-ANLP/blob/master/README.md (Past slides in 2018)
  - https://yale-lily.github.io/aan/ (Search)
  - https://chrome.google.com/webstore/detail/google-drawings/mkaakpdehdafacodkgkpghoibnmamcme?hl=en-US (For creating diagrams)
  - https://huggingface.co/transformers/index.html