ACL 2022
22ND – 27TH MAY | 60TH MEETING | DUBLIN

Non-autoregressive Sequence Generation

Jiatao Gu, Meta AI
Xu Tan, Microsoft Research Asia

Microsoft Meta AI

https://github.com/NAR-tutorial/acl2022

Speaker information

- Jiatao Gu (顾佳涛)
- Research Scientist @ FAIR Labs, Meta AI
- Research interests:
  - Generative Models, NLP, 3D and multimodal learning, Deep Learning

- Some links
  - Homepage: https://jiataogu.me/
  - Google Scholar: https://scholar.google.com/citations?user=cB1mFBsAAAAJ

Speaker information

- Xu Tan (谭旭)
- Senior Researcher @ Microsoft Research Asia
- Research interests: deep learning, NLP/Speech/Music, data generation
  - NMT, text generation, language pre-training
  - TTS, ASR, AI Music
- Some links
  - Homepage: https://www.microsoft.com/en-us/research/people/xuta/
  - Google Scholar: https://scholar.google.com/citations?user=tob-U1oAAAAJ

Outline

- Part I: Introduction (Jiatao Gu)
- Part II: Methods (Jiatao Gu)
- Part III: Applications (Xu Tan)
- Part IV: Open Problems (Xu Tan)

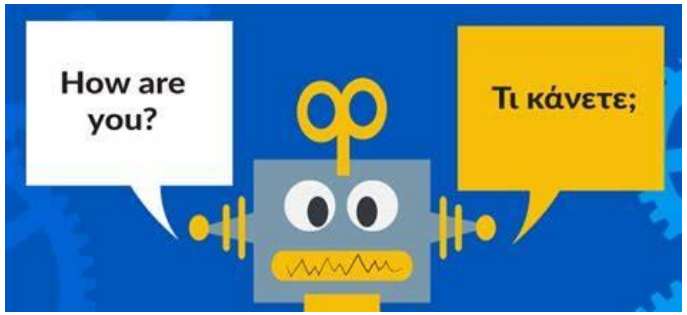
https://github.com/NAR-tutorial/acl2022

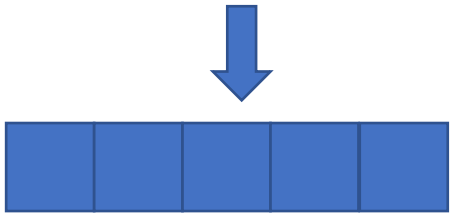
Non-Autoregressive Sequence Generation

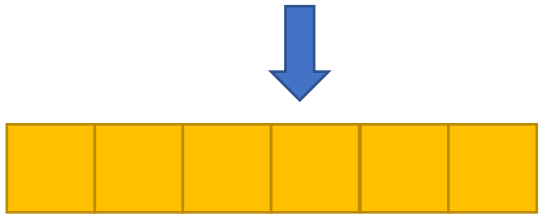
(Part I: Introduction)

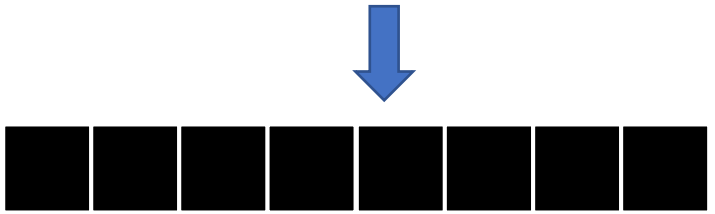
Neural Sequence Generation

- Many real-world applications can be seen as **sequence generation**!
  - We can transform any structured data into sequence
  - For instance, speech as wave sequence; images can be flattened into pixel sequences, etc.



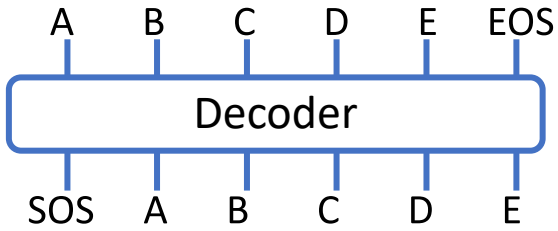






Autoregressive (AR) Sequence Generation

- Generate sequence token by token in an autoregressive way



- Factorize the joint probability in a chain rule

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i|x_1, x_2, \ldots, x_{i-1}) = \prod_{i=1}^{n} p(x_i|\mathbf{x}_{<i})$$

- Has been poplar in many sequence generation tasks (NLP, speech, CV)
  - Can be implemented with RNN/CNN/Transformer
  - Can be unconditional (a single decoder) or conditional (encoder-decoder)

Popular models in AR: Transformer & LLM

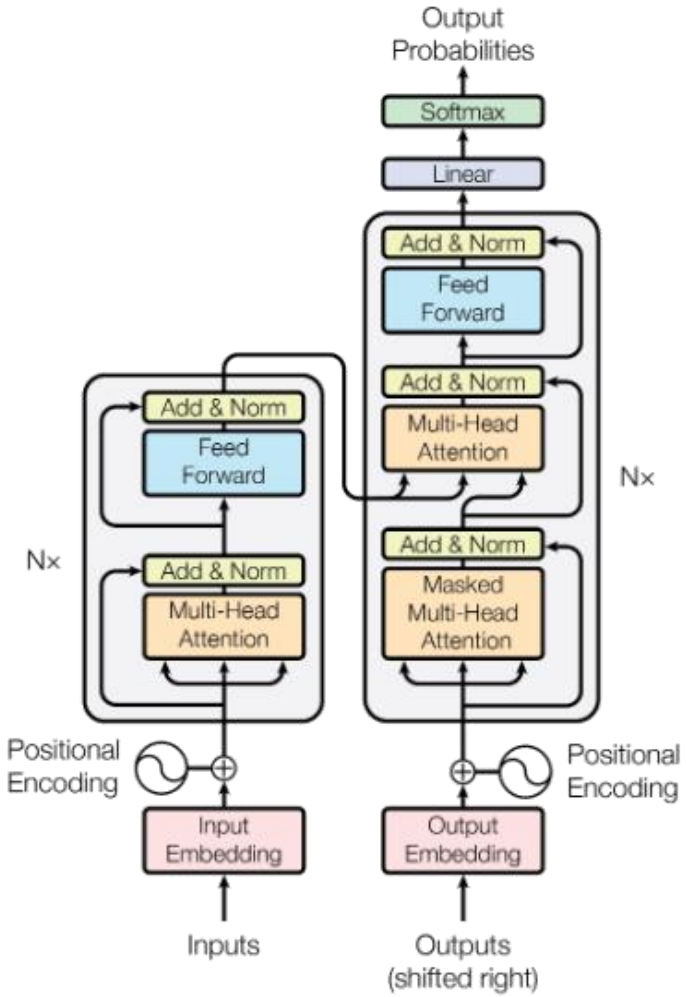
- Transformer (2017)
  - Unleash the modeling power by removing the inductive bias in RNN and CNN

    through QKV attention

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

  - Has become the model backbone for NLP, Speech and CV

- GPT-1/2/3
  - Large-scale pre-training models using autoregressive Transformer



Transformer & LLM

- Zero-shot / One-shot in-context learning with LLM

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  cheese =>                           ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← example
3  cheese =>                           ← prompt
```

- Multi-modal generation (DALLE-1)

TEXT PROMPT

an illustration of a baby daikon radish in a tutu walking a dog

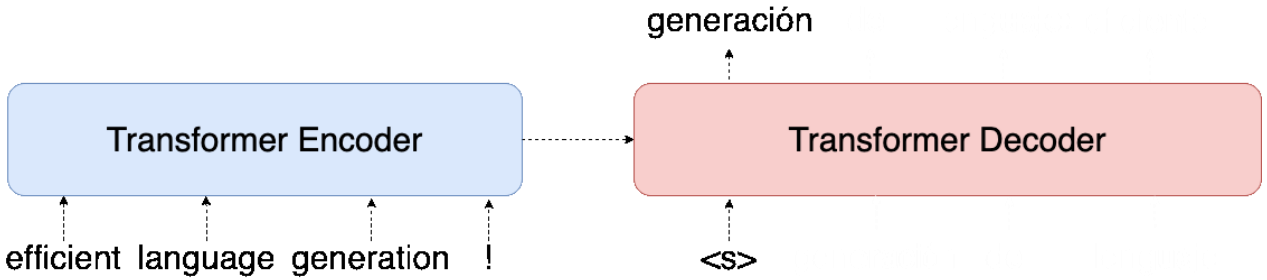
AI-GENERATED IMAGES

Meta Microsoft

Then, why do we want to explore other alternatives

Why do we care?

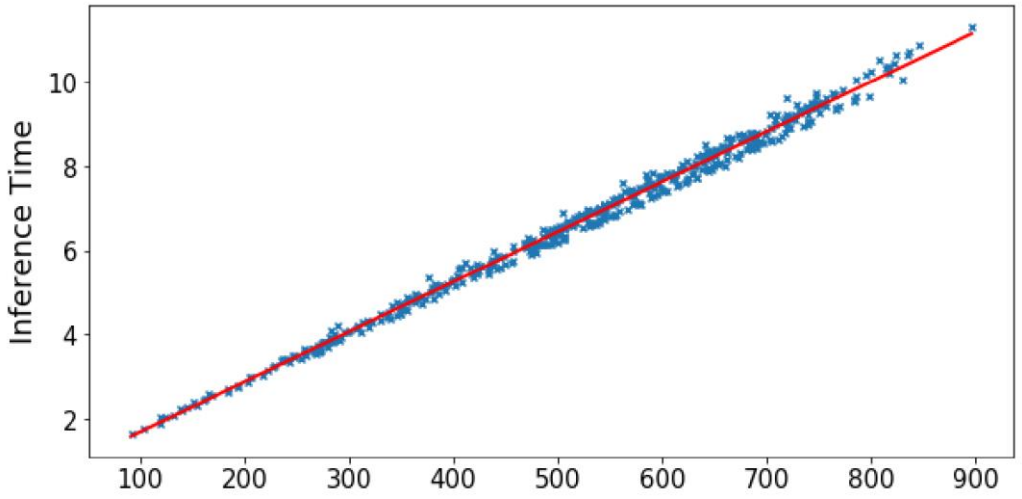
- Text generation models are mostly autoregressive
- Each step needs a forward pass of deep Transformer layers.



(a possible illustration)

AR model is slow

- Inference time linearly grows with sequence length



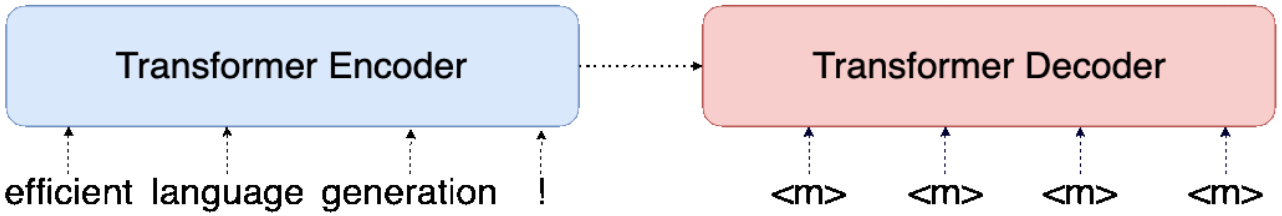
- Time consuming in real-time scenarios, especially for long text/speech/image sequence
  - Text: 10~100
  - Speech: 500 for spectrogram or 80K for waveform for a 1s speech
  - Image: 65536 (256*256) for pixel and 256 (16*16) for tokens

Harmful biases in AR generation

- Exposure bias
  - Exposure bias: ground-truth tokens are taken as input in training (teacher-forcing), but predicted tokens are taken as input in inference (free-run)
  - Error propagation: later tokens will be affected by the accumulated errors in previous tokens
- Label bias
  - The normalization constraint over vocabulary items at each decoding step in autoregressive models poses a harmful inductive bias which leads to learning miscalibrated distributions over tokens and sequences
- Order bias
  - Left-to-right generation may not be the best order for generation, or some data prefers no order

Non-autoregressive Text Generation

- Can we synthesize text in parallel like typical image synthesis?
- We propose the first non-autoregressive translation (NAT) model, which builds on top of Transformer.

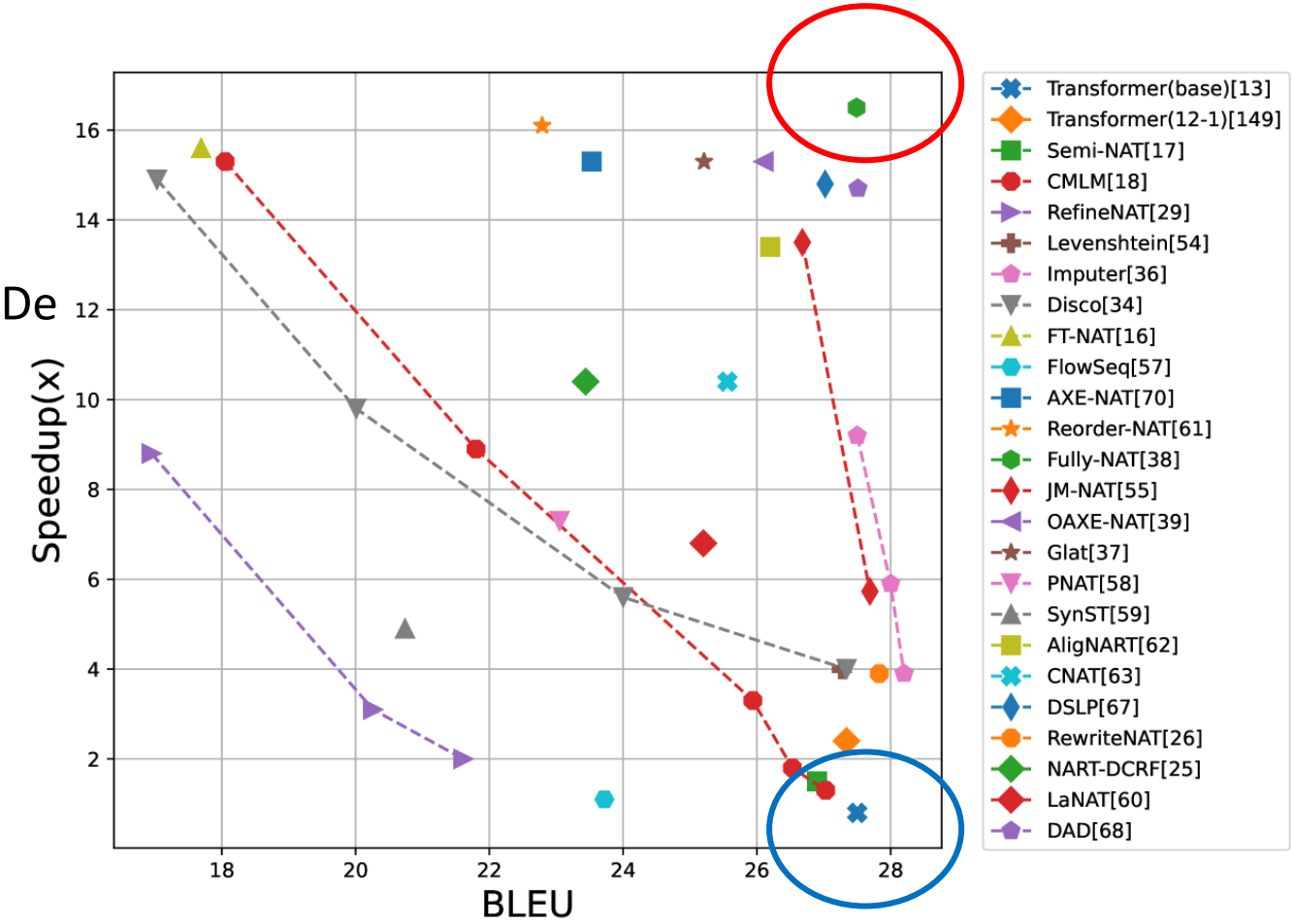


Gu, J., Bradbury, J., Xiong, C., Li, V.O. and Socher, R., 2017. Non-autoregressive neural machine translation. *ICLR 2018*

Evaluation protocol

How can we compare the performance between AR and NAR systems?

- Standard benchmark:
  - Machine Translation, WMT14 En → De
- Quality measure
  - BLEU score
- Efficiency measure
  - Latency (GPU batch=1, batch=full and CPU batch=1)
- Quality-efficiency trade-off figure

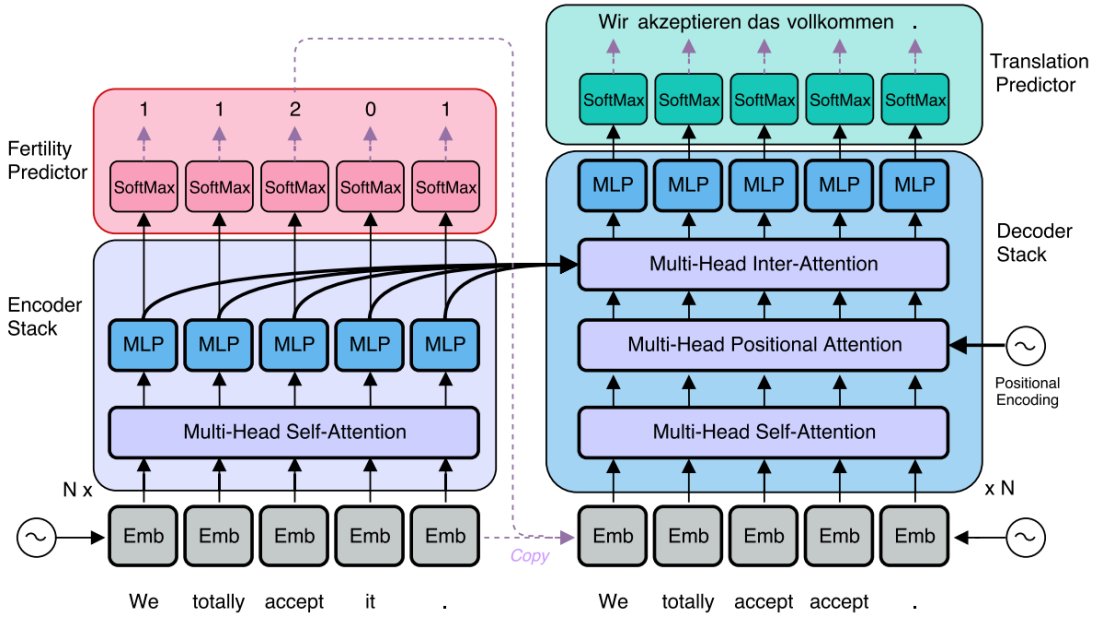


Meta Microsoft

Why is NAR difficult?

- The original NAR system is far from the AR baselines.

Models	BLEU
Autoregressive Baselines	**27.48**
NAT	17.69
NAT with reranking	19.17



Gu, J., Bradbury, J., Xiong, C., Li, V.O. and Socher, R., 2017. Non-autoregressive neural machine translation. *ICLR 2018*

Why is NAR difficult?

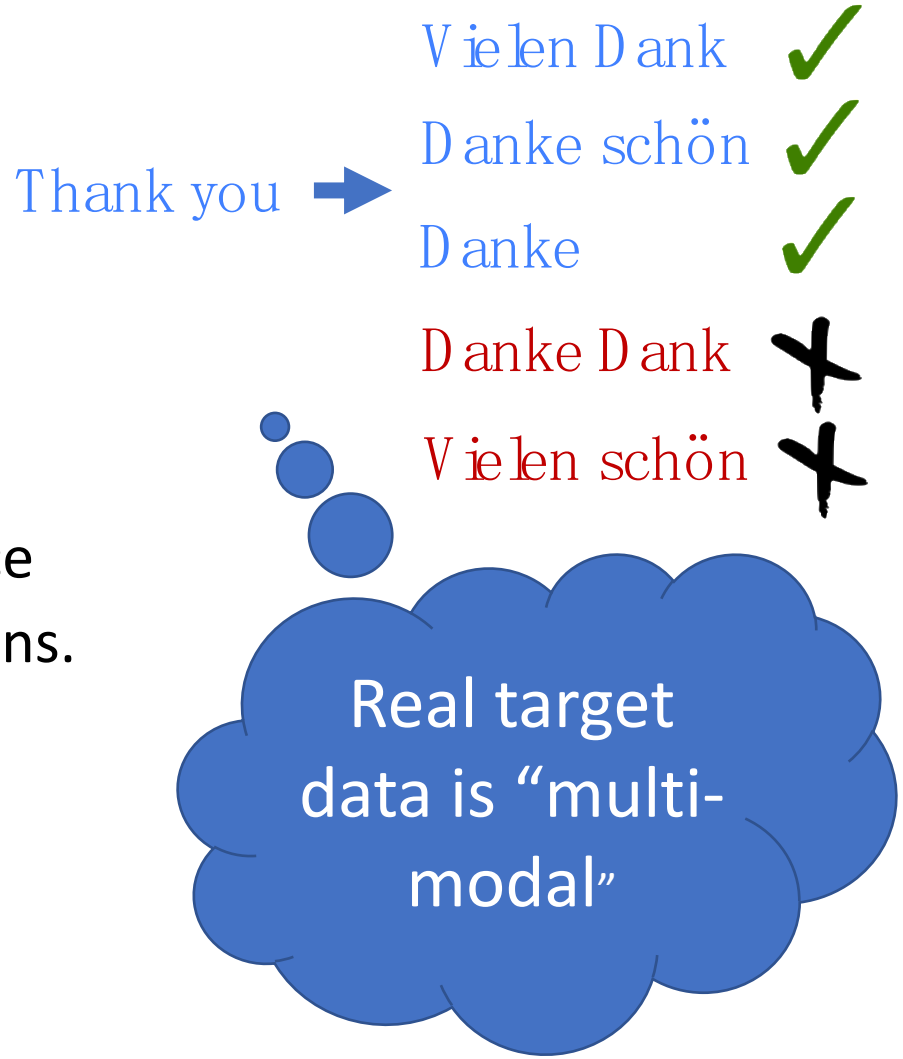
Typical errors made by a NAR system:

- Repetitive tokens (over generation)
- Shorter or broken sentences (under generation)
- Influent sentence (no/weak language model)

Fundamental issue:

- The independence assumption in the output space **ignores** the real dependency between target tokens.

- Maximum-likelihood training force to cover all possible modes



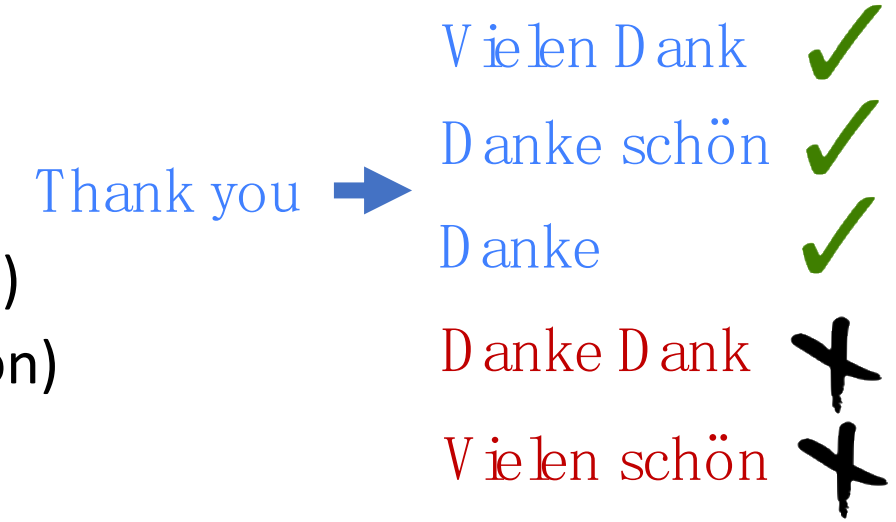
Why is NAR difficult?

How AR system solve this problem?

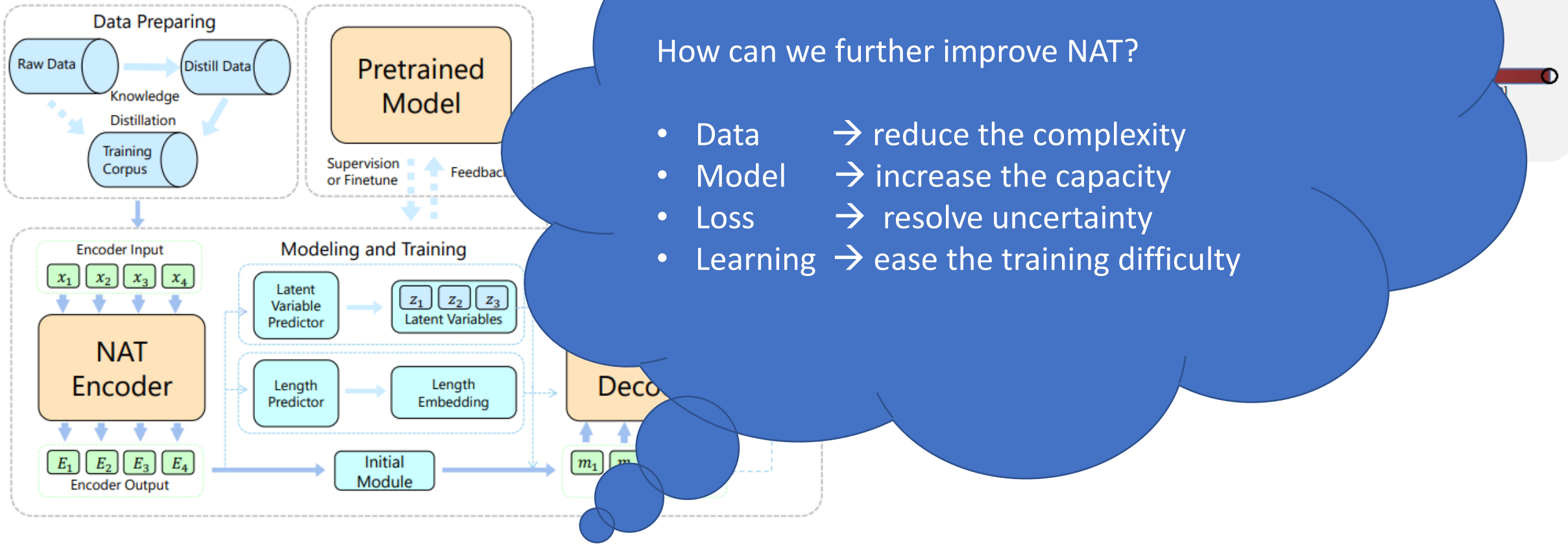
- P(Vielen) * P(Dank|Vielen) * P(<eos>|Vielen Dank)
- P(Danke) * P(schon|Danke) * P(<eos>|Danke schon)

AR model explicitly models the dependency in the target space. Generation is to make choice at each step and affect the next choice.

- P(Vielen) * P(Dank)
- P(Danke) * P(schon)



How can we improve NAR models to match AR systems?



Xiao, Y., Wu, L., Guo, J., Li, J., Zhang, M., Qin, T. and Liu, T.Y., 2022. A Survey on Non-Autoregressive Generation for Neural Machine Translation and Beyond. arXiv preprint arXiv:2204.09269.

Non-Autoregressive Sequence Generation

(Part II: Methods)

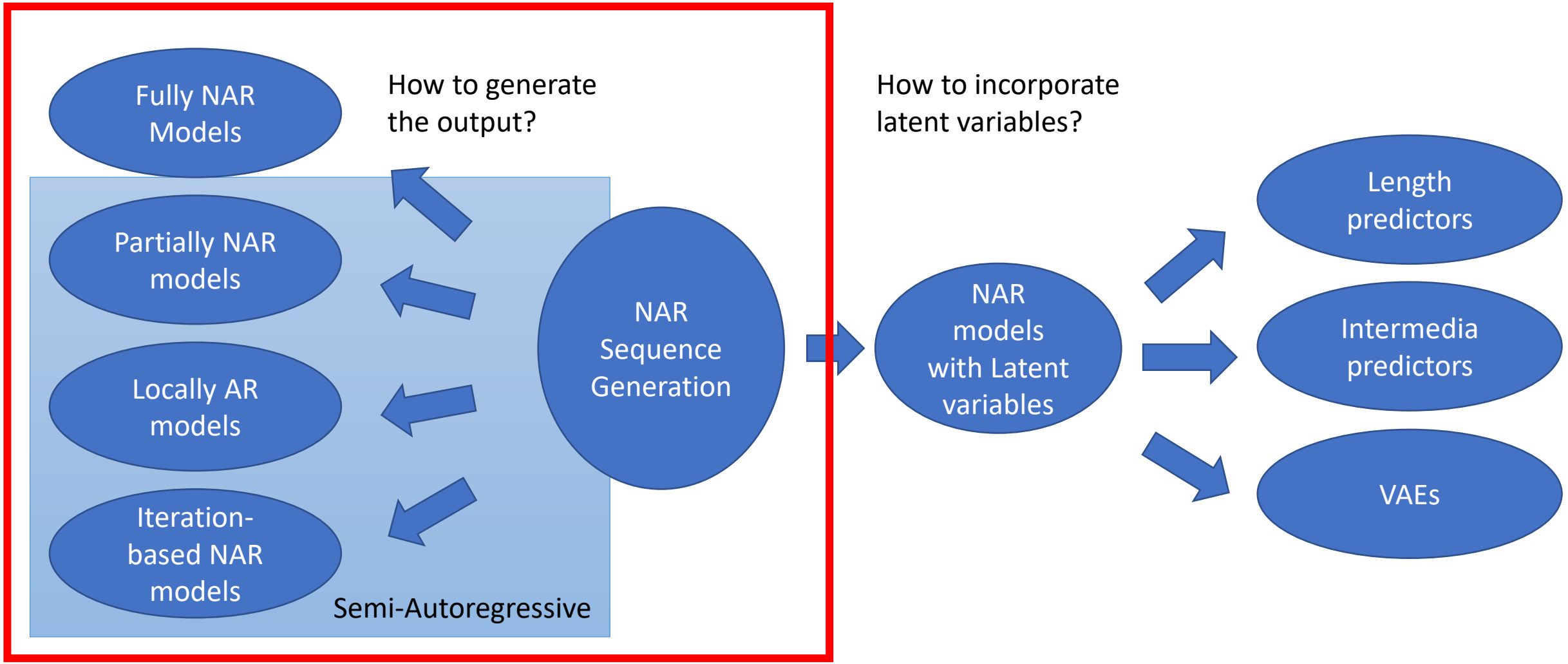
A principled goal

- The main challenge of NAR generation is "*failure of capturing the target side dependency*".

- Overall instructions:

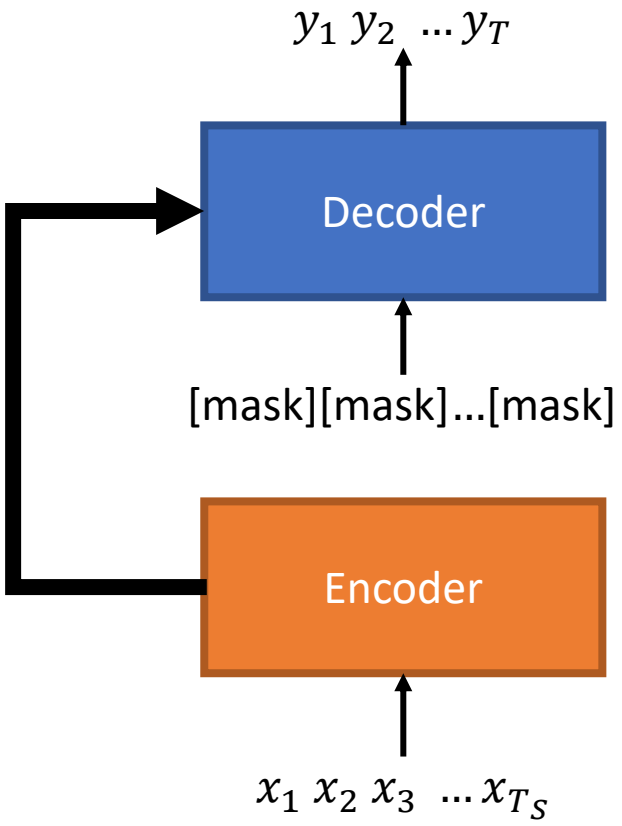
Methods	Overall instructions
Model architecture	Improve the model's capability of capturing target-side dependency
Objective function	Provide learning signal that resolves uncertainty
Training data	Reduce the target-side dependency in the training
Learning paradigm	Better paradigm to ease the difficulty of learning
Inference techniques	Additional techniques that improve the final performance

Model Architecture



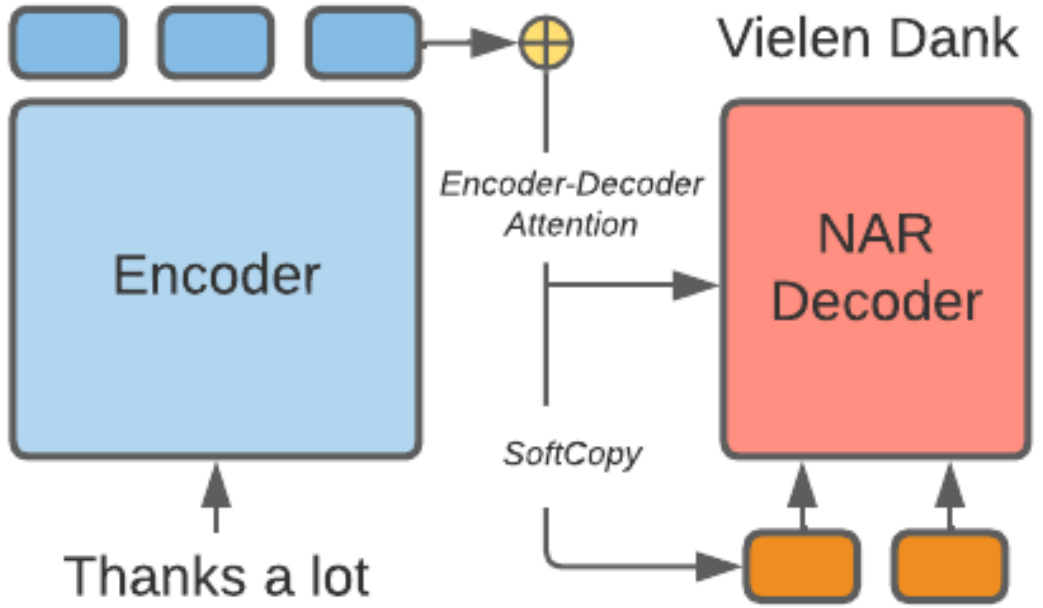
Fully NAR Models

- The simplest way of generating tokens in parallel.
  - Time complexity: O(1)
  - Pros:
    - Simple to implement
    - Fastest decoding
    - **Easy to incorporate other improvements.**
  - Difficulties:
    - Relatively worst performance
    - **CANNOT** handle target token dependencies without other improvements
    - Hard to work well on LONG sequences.



Example 1: vanilla NAT model

- Following the initial work, we can simply implement such vanilla NAR by removing causal masking in the Transformer decoder.
- Inputs to the decoder:
  - Special mask tokens
  - Soft-copy of encoder's hidden states or embeddings
- Naïve implementation is very bad:
  - ~10 BLEU on raw dataset.



Gu, J., Bradbury, J., Xiong, C., Li, V.O. and Socher, R., 2017. Non-autoregressive neural machine translation. *ICLR 2018*

Fully NAR model + tiny AR layer

- As a special case, fully NAR systems can be significantly improved by adding a **small AR layer** in the output. It will help to model the “missing” target side dependency without sacrificing too much on latency.

- It is technically autoregressive, while most of the computation is done in the NAR part. In this sense, it is very different from typical Transformer-based models.

- Tiny AR layer can be:
  - N-gram LM
  - CRF
  - Tiny RNN (e.g. ,1-layer GRU or SRU)

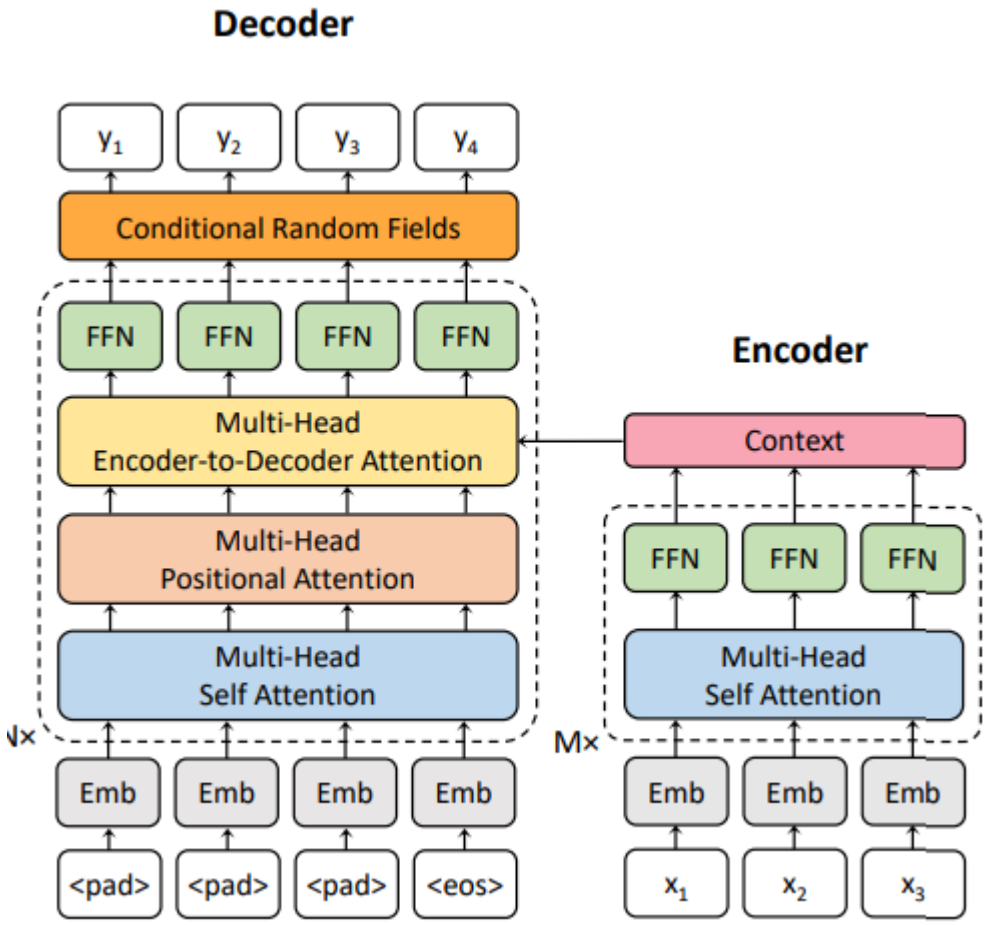
Example 2: NAR + CRF layer

- By adding the conditional random fields (CRF) on top of the original NAR model's output, it considers the dependency of the neighbors:

$$P(y|x) = \frac{1}{Z(x)} \exp\left(\sum_{i=1}^{n} s(y_i, x, i) + \sum_{i=2}^{n} t(y_{i-1}, y_i, x, i)\right)$$

- Apply low-rank/beam approximation to make the transition feasible for large vocabulary.

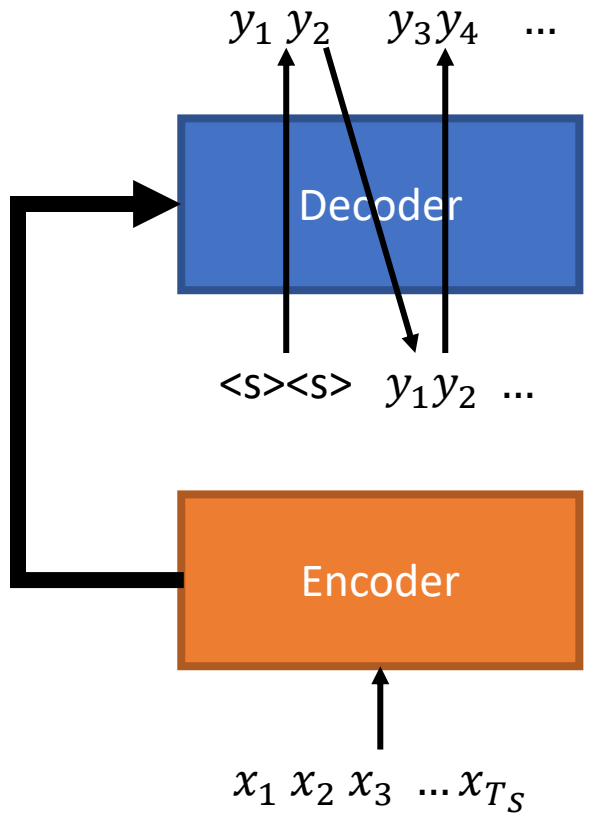
- Decoding is sequential but much faster than typical AR models.



Z. Sun, Z. Li, H. Wang, D. He, Z. Lin, and Z. Deng, "Fast structured decoding for sequence models," NeurIPS, vol. 32, pp. 3016–3026, 2019.

Partially NAR models

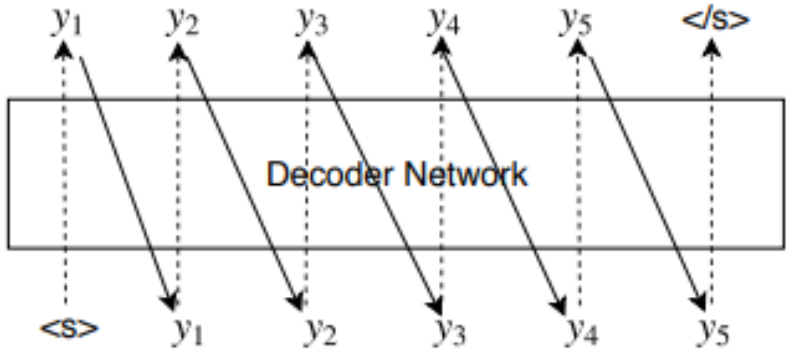
- AR at sequence level, output multiple tokens in parallel
  - In the middle between AR and fully NAR models
  - Time complexity: O(N/K) (K is the segment length)
  - Pros:
    - Produce better performance
    - If tuning properly, it can achieve the same performance as AR
  - Difficulties:
    - Globally still AR, it did not change the linear complexity
    - Speed-up is small



Example 3: n-gram decoding

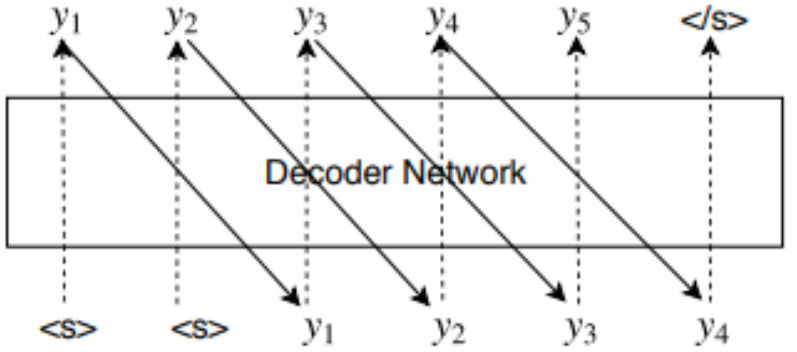
- Decode multiple tokens in parallel (usually n-gram), and then the overall sentence is still autoregressive.

$$p(y_1 \dots y_n|\mathbf{x}) = \prod_{t=1}^{[(n-1)/K]+1} p(G_t|G_1 \dots G_{t-1}, \mathbf{x})$$



- Relaxed causal mask: a block-wise attention which can make max use of the decoded contexts.

$$\begin{bmatrix} 1 & \mathbf{0} & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & \mathbf{0} & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & \mathbf{0} \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & \mathbf{1} & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & \mathbf{1} & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & \mathbf{1} \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$



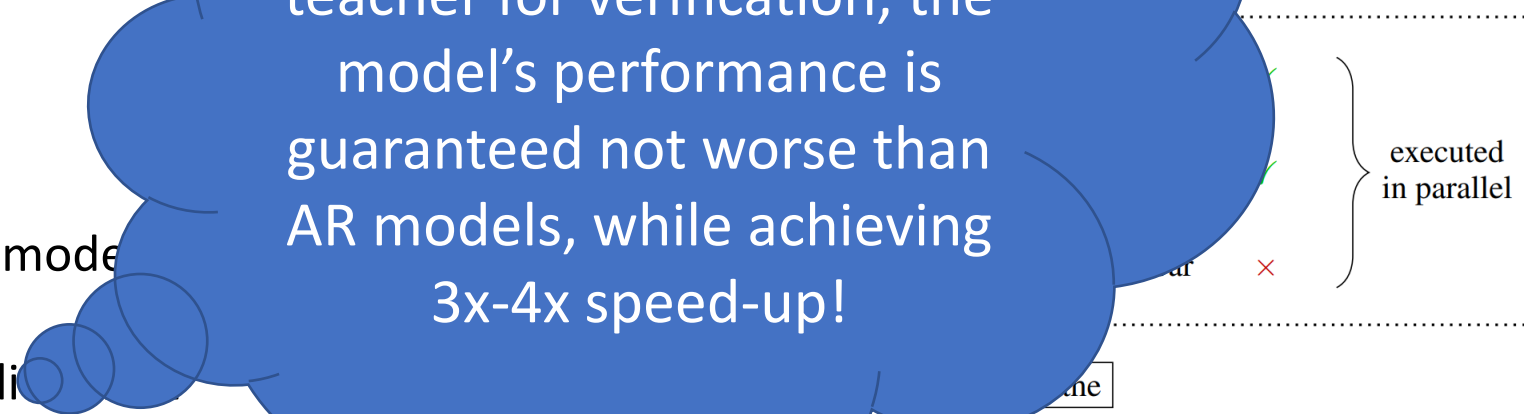
C. Wang, J. Zhang, and H. Chen, "Semi-autoregressive neural machine translation," in EMNLP, 2018, pp. 479–488

Example 4: Block-wise decoding

- A dynamic approach compared to “fixed” n-gram prediction…

- The model has three stages:
  - **Prediction**: each step the NAR mode predict an n-gram block.
  - **Verification**: compare the predi NAR model, and the AR teacher’s prediction in a *teacher forcing* matter.
  - **Accept** the sequence with longest match and move to predict next block.

Due to the use of AR teacher for verification, the model’s performance is guaranteed not worse than AR models, while achieving 3x-4x speed-up!



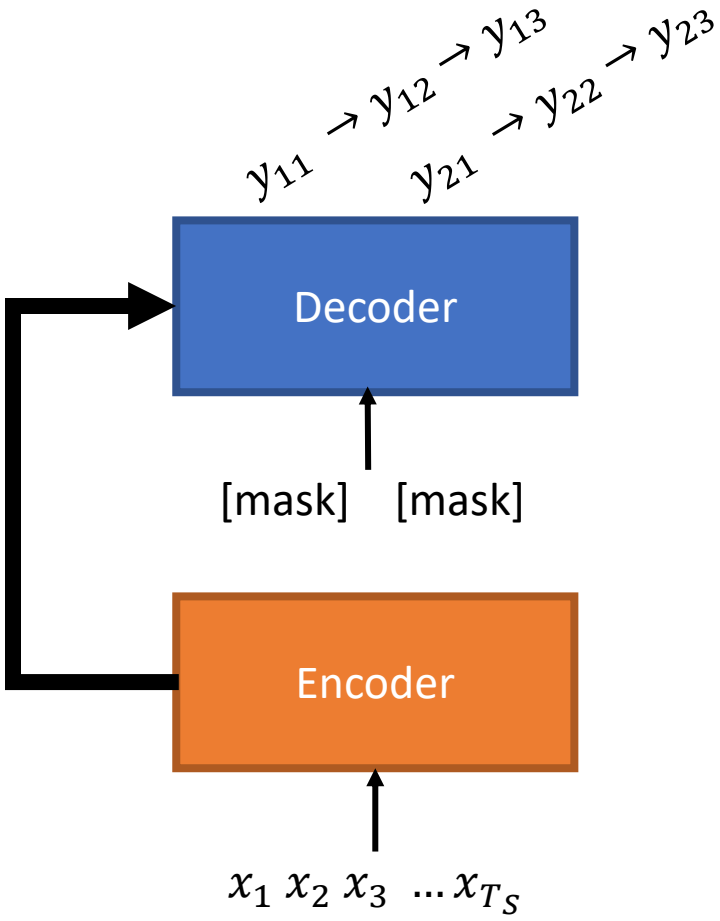
Stern, M., Shazeer, N. and Uszkoreit, J., 2018. Blockwise parallel decoding for deep autoregressive models. Advances in Neural Information Processing Systems, 31.

Locally AR models

- NAR at sequence level, output each segment in AR
  - In the middle between AR and fully NAR models
  - Time complexity: ~O(K) (K is the segment length)
  - Also have similarities to NAR + tiny AR layer.

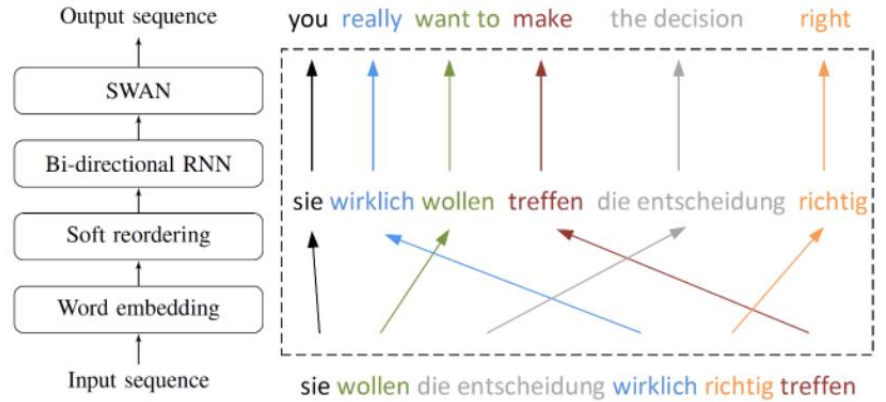
  - Pros:
    - Produce better performance
    - **Motivated by the fact the NAR models deal with local dependencies badly, e.g., repetitive words.**
    - Faster generation compared to previous type

  - Difficulties:
    - Need **complex algorithm** to merge the output or resolve conflicts between each position.

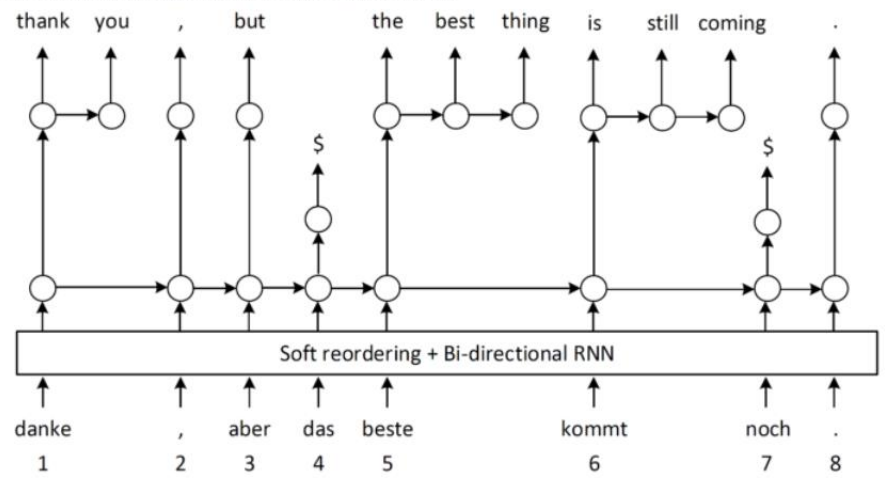


Example 5: Neural Phrase-based Translation

- Pre-transformer era work, still based on RNNs.
- Similar to traditional statical machine translation, model is learned to translate phrase independently with local RNNs.
- Merging phrases need to run dynamic programming (DP) which limits its application to high-resource domains.



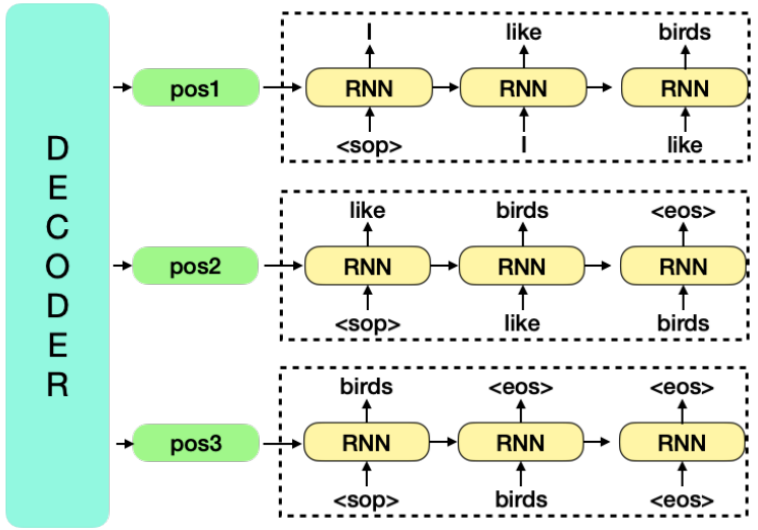
An illustration of using NPMT in German-English translation



Huang, P.S., Wang, C., Huang, S., Zhou, D. and Deng, L., 2018, February. Towards Neural Phrase-based Machine Translation. In International Conference on Learning Representations.

Example 6: Local autoregressive translation (LAT)

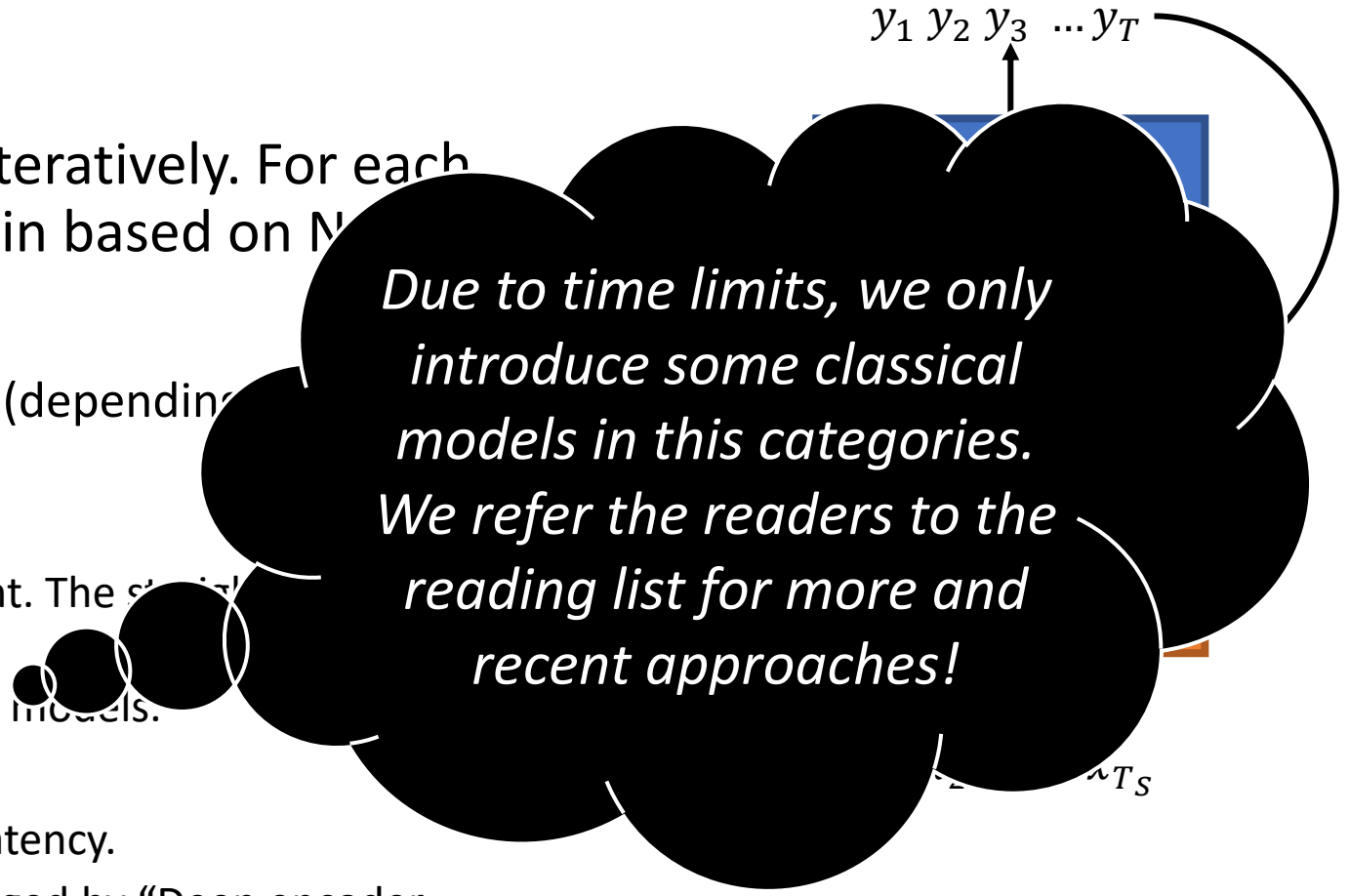
- Similar to NPMT, this paper also learned to predict local sequence with a small RNN locally.
- No DP needed, but a heuristic merging operation based on longest common string between two positions.
- Different from NAR + tiny RNN, it is still constant time. Because of the local AR, it can mostly remove artifacts like repetition.



X. Kong, Z. Zhang, and E. Hovy, "Incorporating a local translation mechanism into non-autoregressive translation," in EMNLP, 2020, pp. 1067–1073

Iteration-based Models

- Generate/refine the sequence iteratively. For each iteration, the sequence output in based on N
  - Time complexity: O(K) or O(logN) (depending methods to use)
  - Pros:
    - General and simple to implement. The st… extension of fully NAR models.
    - Best performance so far for NAR models.
  - Difficulties:
    - Trade-off between quality and latency.
    - The speed-up advantage challenged by “Deep encoder, shallow decoder”.

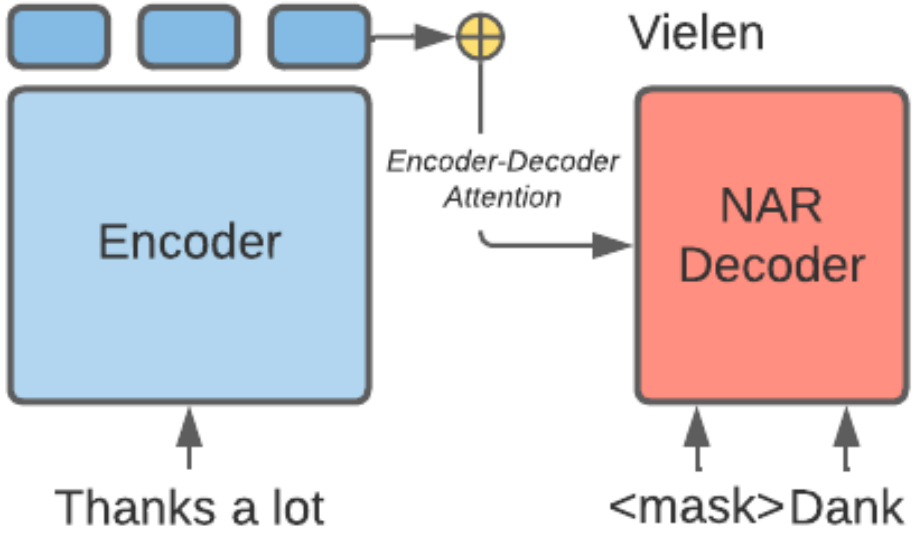


Kasai, J., Pappas, N., Peng, H., Cross, J. and Smith, N.A., 2020. Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. arXiv preprint arXiv:2006.10369.

Example 7: Mask-Predict

- Following similar training tasks of BERT (masked language model), we can directly use the mask prediction model for iterative generation.

- BERT training 15%, while CMLM is trained by randomly masking 0~100% tokens.

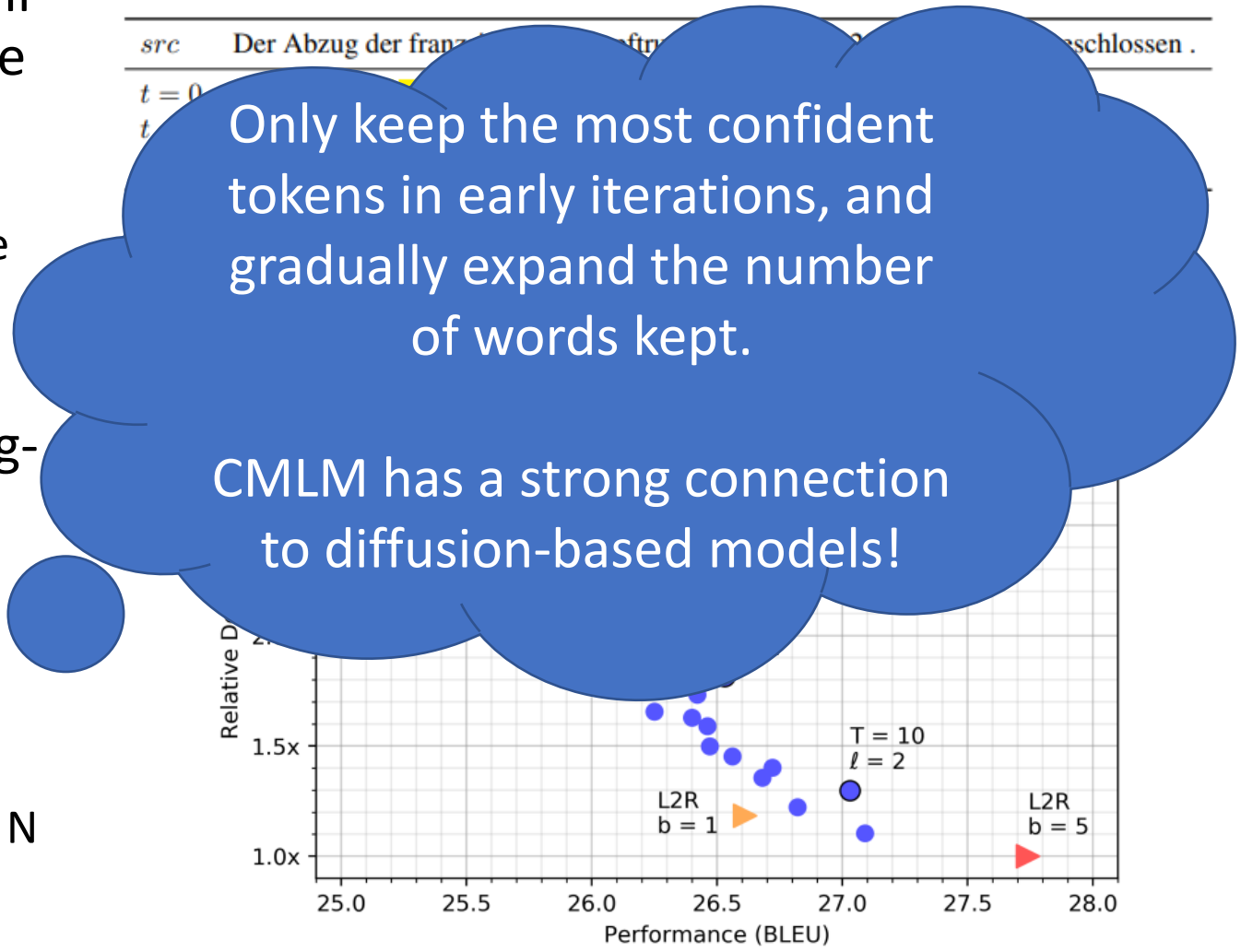


Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

M. Ghazvininejad, O. Levy, Y. Liu, and L. Zettlemoyer, “Maskpredict: Parallel decoding of conditional masked language models,” in EMNLP-IJCNLP, 2019, pp. 6112–6121.

- The BERT-style training did not tell us how should we inference in the testing time.
  - As it is just forcing the model to predict the masked tokens from the remaining.

- Skeptical decoding -- an annealing-based method was used for iterative translation
  - Set the target iteration T.
  - Start with all masks;
  - For each iteration t, make the prediction, and then mask (1-t/T) * N tokens with lowest scores.



M. Ghazvininejad, O. Levy, Y. Liu, and L. Zettlemoyer, “Maskpredict: Parallel decoding of conditional masked language models,” in EMNLP-IJCNLP, 2019, pp. 6112–6121.

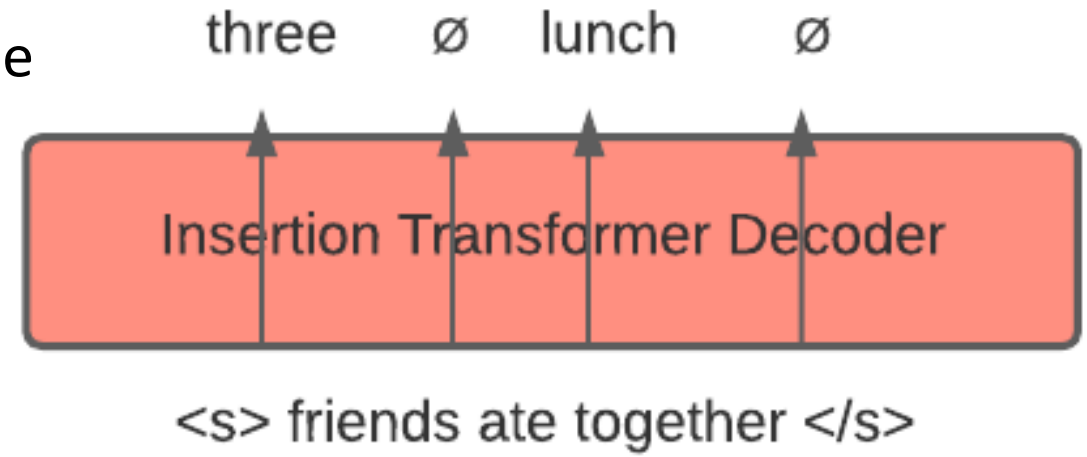
Example 8: Insertion Transformer

- There is not only one way to perform iterative generation. Sentence can also be composed by insertion!

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]	(⟨EOS⟩, 5)

- Mask-Predict is constrained to know (predict) the length before generation, while with insertion, it is possible to generate sequences in any lengths during iteration.



- Generation terminates if all positions are not insertable. The complexity is logarithm.

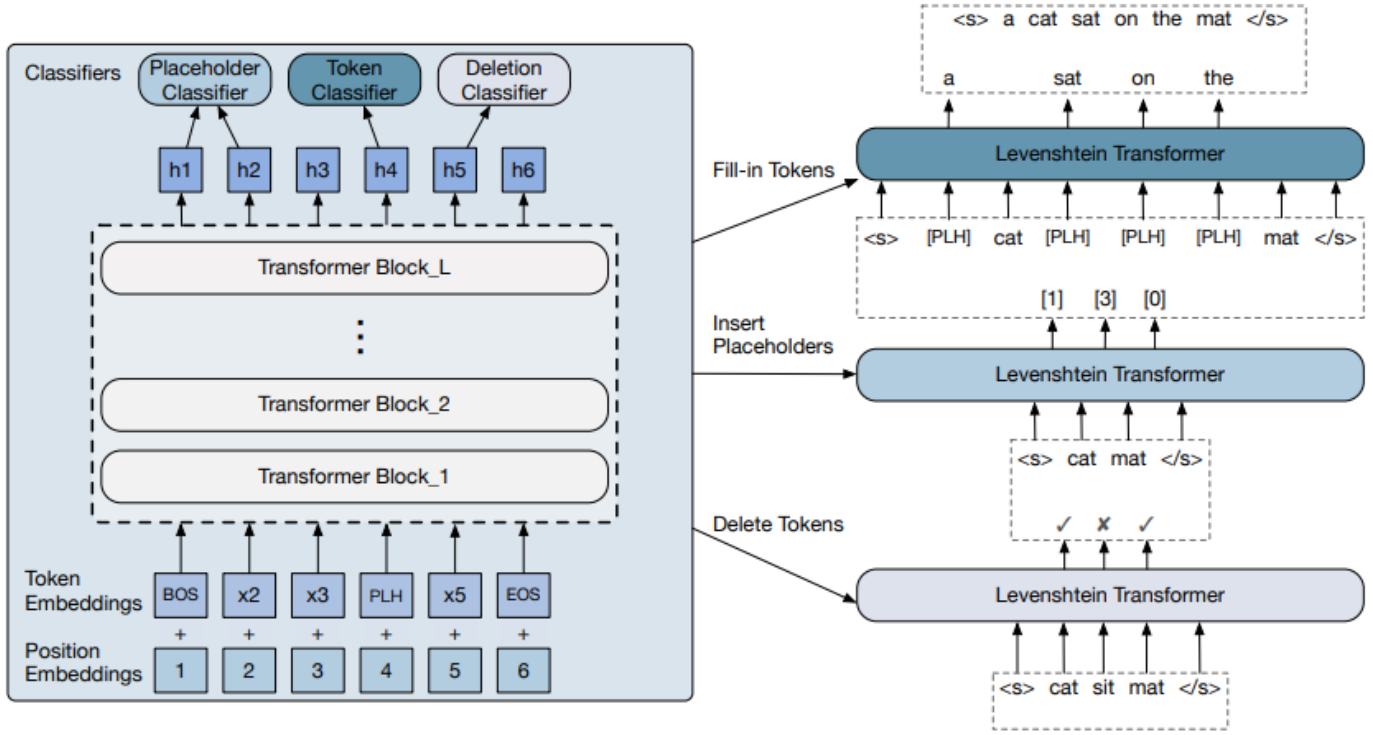
M. Stern, W. Chan, J. Kiros, and J. Uszkoreit, "Insertion transformer: Flexible sequence generation via insertion operations," in ICML. PMLR, 2019, pp. 5976–5985

Meta Microsoft

Example 9: Levenshtein Transformer (LevT)

- With one step further, LevT combines insertion & deletion, which means the model can freely edit over the generation, change the length and terminate the iteration in a dynamic way.

- For each iteration, it contains three forward passes:
  - Parallel deletion
  - Parallel insertion (predict # of masks, mask prediction)



J. Gu, C. Wang, and J. Zhao, "Levenshtein transformer," NeurIPS, vol. 32, pp. 11 181–11 191, 2019

Meta Microsoft

- LevT is trained through **imitation learning**, with a dual policy (simplifie version):
  - *Learning to insert tokens by predicting random deletion;*
  - *Learning to delete tokens by fixing errors made by insertion;*
- The expert action are automatically generated based on Levenshtein distance with ground-truth.

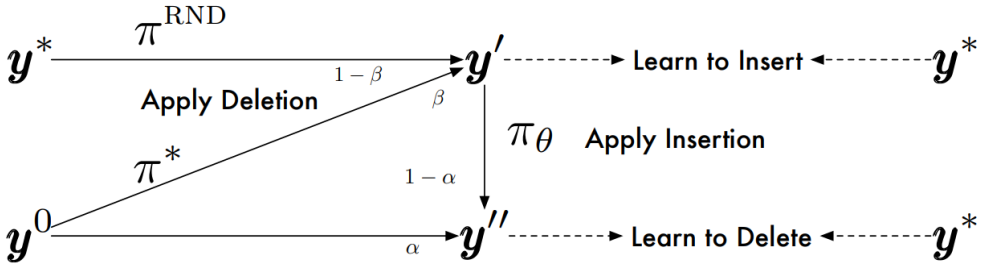
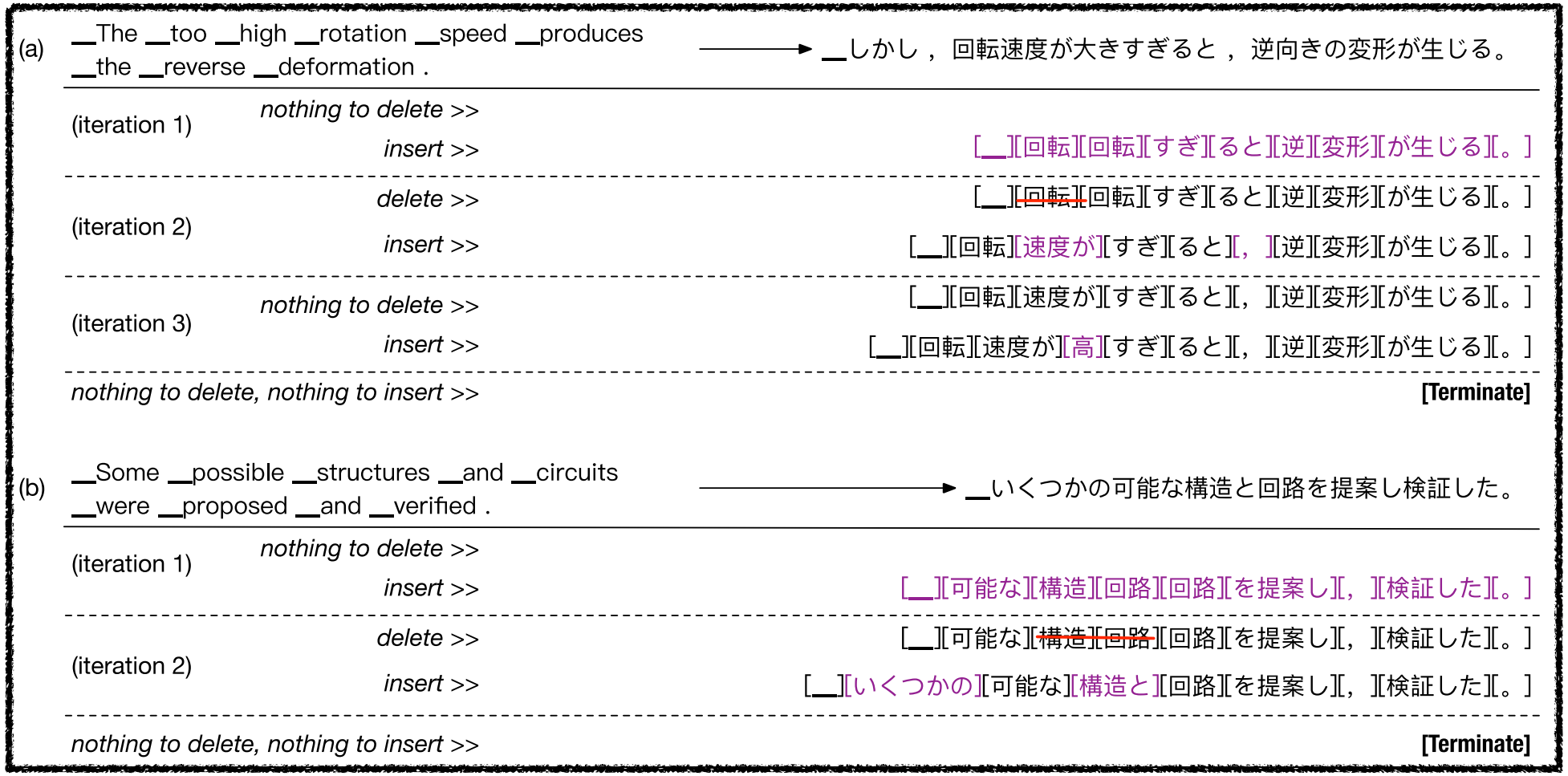


Figure 2: The data-flow of learning.

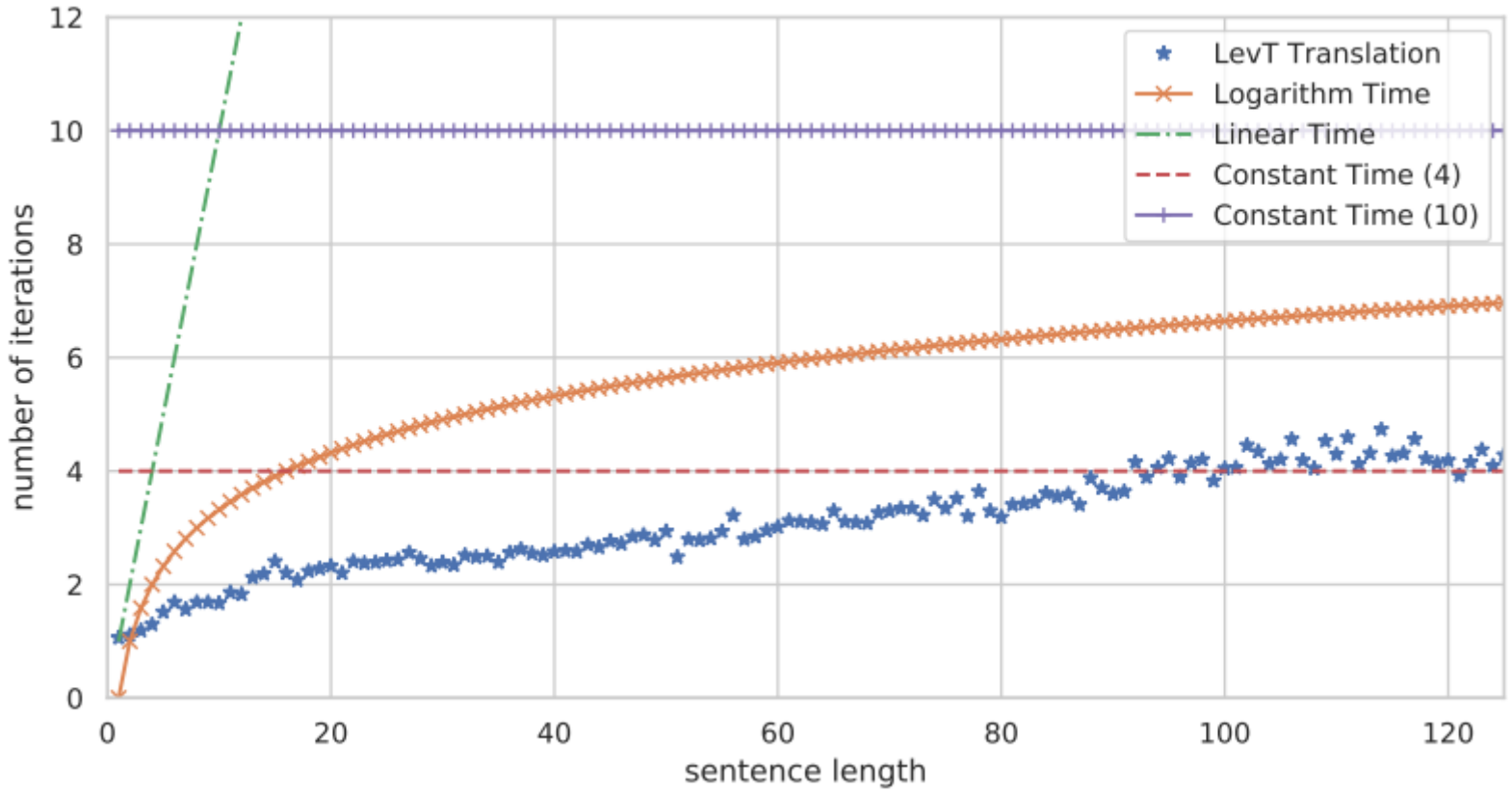
J. Gu, C. Wang, and J. Zhao, “Levenshtein transformer,” NeurIPS, vol. 32, pp. 11 181–11 191, 2019

- An example of iterative refinement using LevT:



J. Gu, C. Wang, and J. Zhao, “Levenshtein transformer,” NeurIPS, vol. 32, pp. 11 181–11 191, 2019

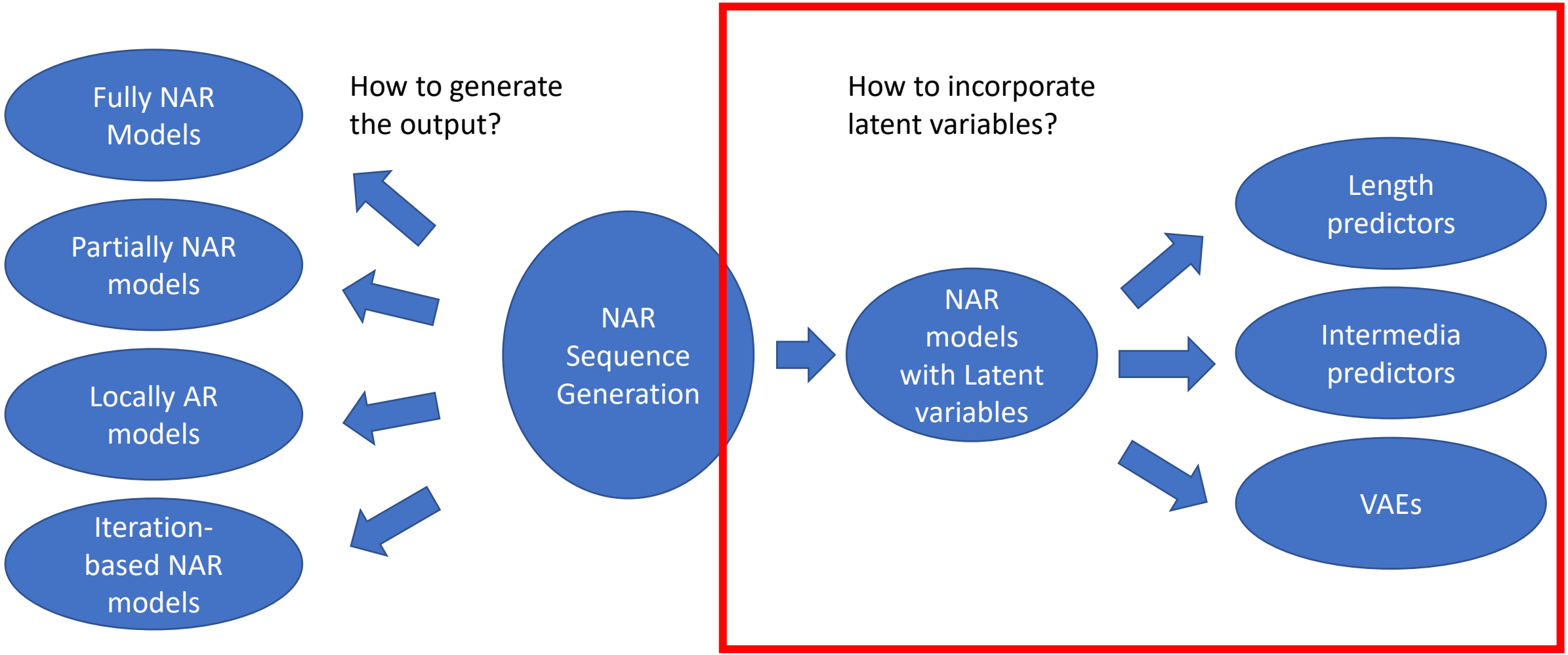
- Speed-up of LevT compared to CMLM (constant time) and Insertion Transformer (logarithm time)



J. Gu, C. Wang, and J. Zhao, “Levenshtein transformer,” NeurIPS, vol. 32, pp. 11 181–11 191, 2019

Meta Microsoft

Model Architecture



Latent Variable Models for NAR generation

The concept of “latent variables” are very important for NAR!

- The additional variables are used to capture uncer
- Iteration-based models can also be seen as “later
  - All the intermedia decoding results can be seen a
  - Y0 (all masks) → Y1 → Y2 → … → YT (target seq

Why latent variables will be useful in NAR genera

Suppose we only have three translation for this sentence, and we determine one mode first, then the output can be generated in NAR!

Vielen Dank

Danke schön

Thank you

Danke

Z=1

Z=2

Z=

Latent variables

Latent Variable Models for NAR generation

- Normally, we need to marginalize the latent variables, or maximize the evidence lower bound (ELBO):

$$L_{ELBO} = \mathbb{E}_{q(Z|X)}[\log P(X|Z) + \log P(Z)] + H(q)$$

- In practice, we can either choose to work on pre-defined "latent variables" with some off-the-shelf predictor or learn everything jointly (e.g., VAEs).

Length predictor

Most NAR systems at least have one “latent variable” – **length**

- AR model usually uses the special symbol <eos> to show the end of generation, and the length is determined when the generation ends.

- However, for most NAR models (except for CTC-based models) need to pre-determine the “length” before NAR generation can starts. In this sense, length itself can also be used to capture some dependency information!

Thank you

Vielen Dank	L=2
Danke schön	L=2
Danke	L=1

Meta Microsoft

Length predictor

Types of length predictor:

- Training a separate classifier to predict the correct length
  - Using the encoder's hidden states and pooling
  - Using specific [LENGTH] token in the encoder

- Statistics
  - In some work, it is also possible to directly use the dataset statistics

$$T_y = \alpha \ T_x + B$$

  - Mostly, B is the size of length beam, and it relies on the re-ranking.

Intermedia predictors

It is a vague categories, which refers to modeling **latent variables**:

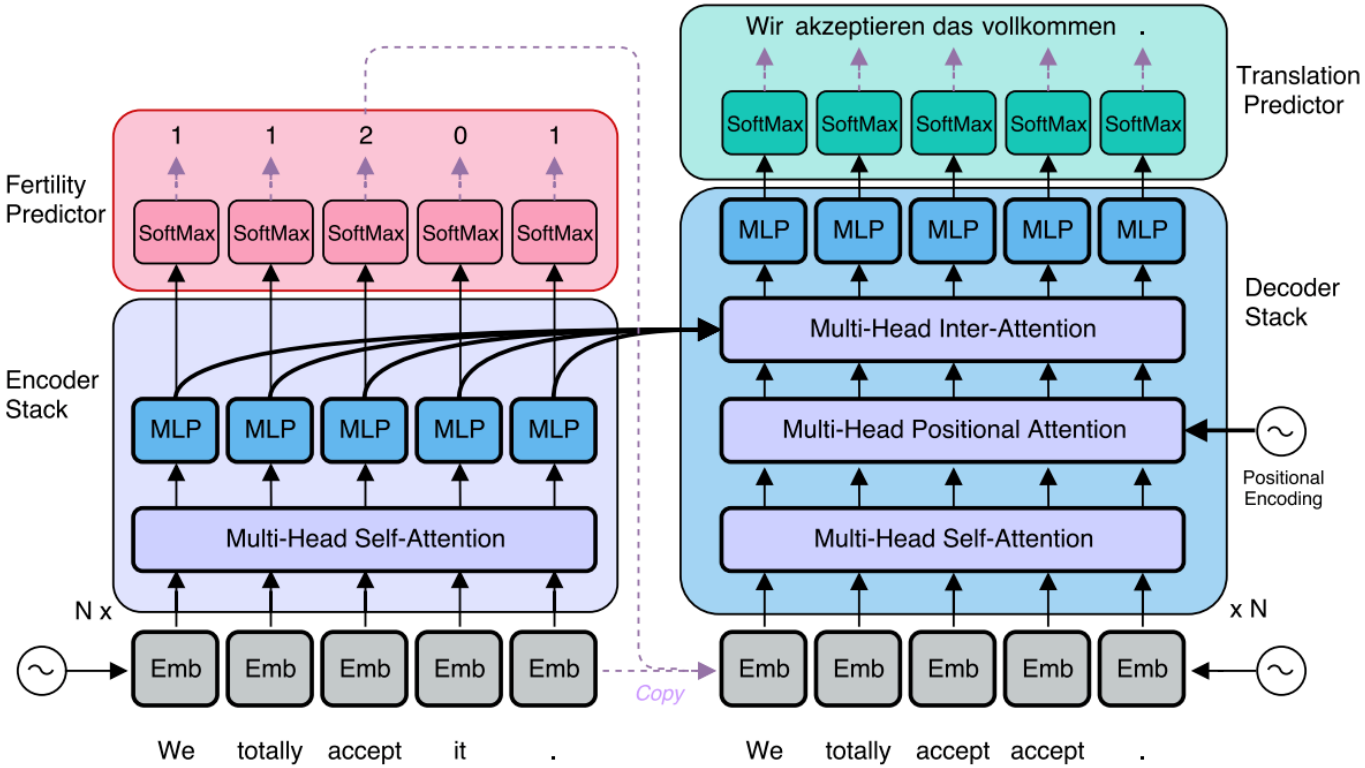
- With clear definitions / meanings
- Predictable given the target or source-target pairs
- Off-the-shelf predictors are available (and typically fixed) to predict these information

- **Not optimal, and the performance affected by the predictor.**

Length can be seen as a special "latent variable" as we don't need to separate inference network to infer that.

Example 10: Fertility predictor

- Fertility: how many words each source token will be translated to, which is estimated by alignment tools.
- Fertility can also be predicted in a NAR way.
- *Bonus point*: we don't need additional length predictor as we can add fertilities together.

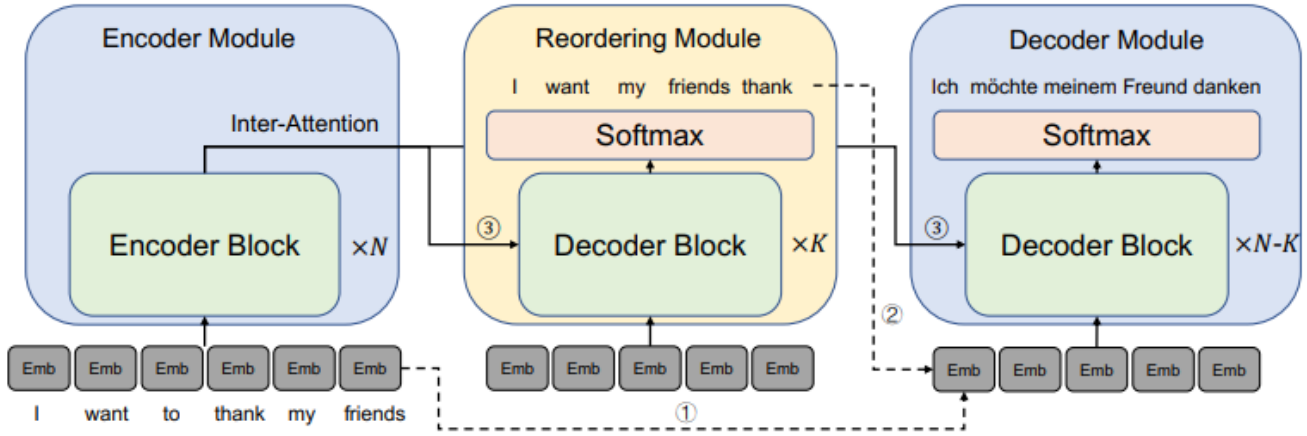


Gu, J., Bradbury, J., Xiong, C., Li, V.O. and Socher, R., 2017. Non-autoregressive neural machine translation. *ICLR 2018*

Meta Microsoft

Example 11: ReorderNAT

- Fertility can only partially help the translation, while it is not possible to handle reordering in MT.
- ReorderNAT used an additional module to predict reordering (which can be either NAR or light-weight AR model).
- Ordering can also be obtained from alignment tools.

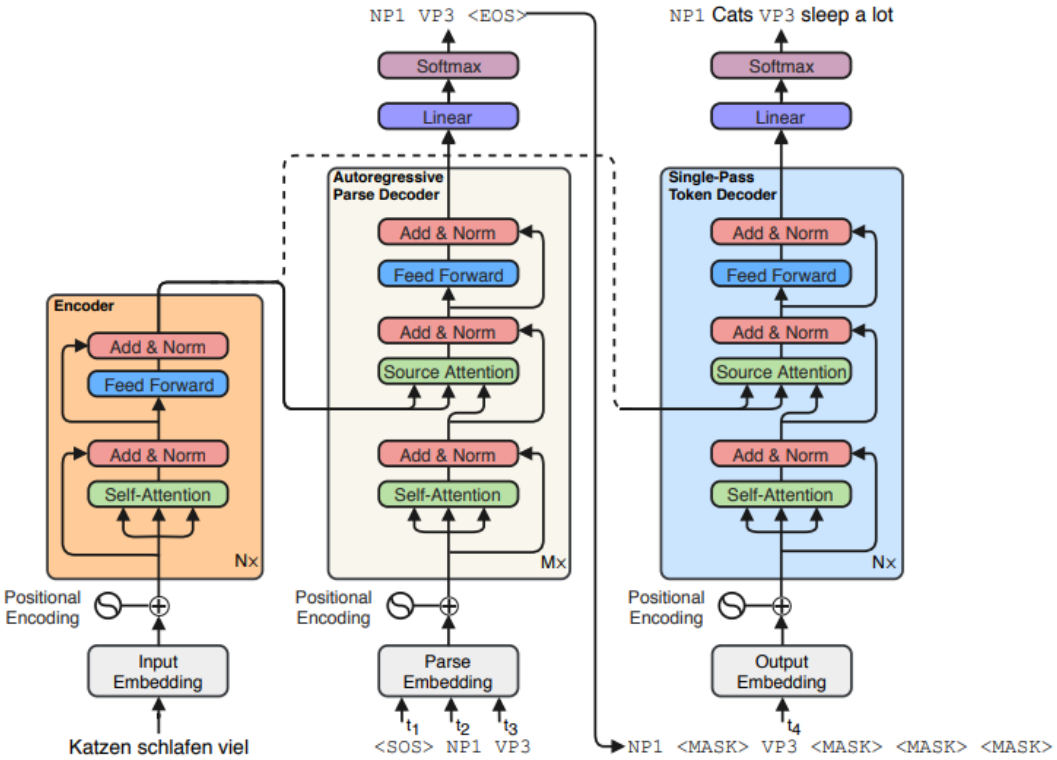


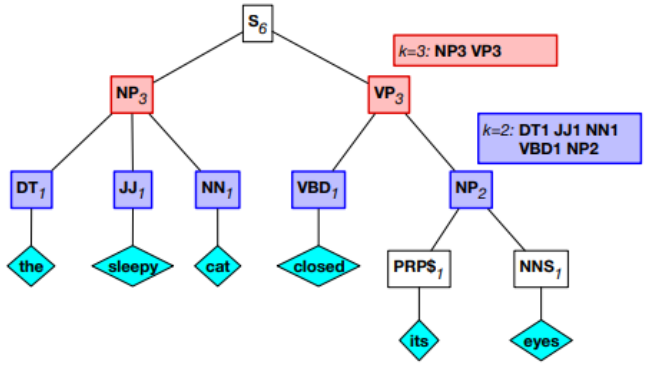
Q. Ran, Y. Lin, P. Li, and J. Zhou, "Guiding non-autoregressive neural machine translation decoding with reordering information," in AAAI, vol. 35, no. 15, 2021, pp. 13 727–13 735

Meta Microsoft

Example 12: SynST

- Similar to ReorderNAT, we can also plug-in other type of predictors in the middle of encoder and decoder.
- SynST learns to autoregressively predict "high-level" chunks using a light-weight AR model, and then generate texts based on these chunks.
- Chunk information can be obtained from a syntactic parser.





N. Akoury, K. Krishna, and M. Iyyer, "Syntactically supervised transformers for faster neural machine translation," in *ACL*, 2019, pp. 1269–1281.

Meta Microsoft

Variational Autoencoders

Different from the previous case, such models are more general

- Latent variables have NO clear definitions / meanings
- Need to optimize both the generator and the encoder (and potentially also the prior).

- Basically, we can either model it with Continuous VAE or Discrete VAE. Both have pros and cons.

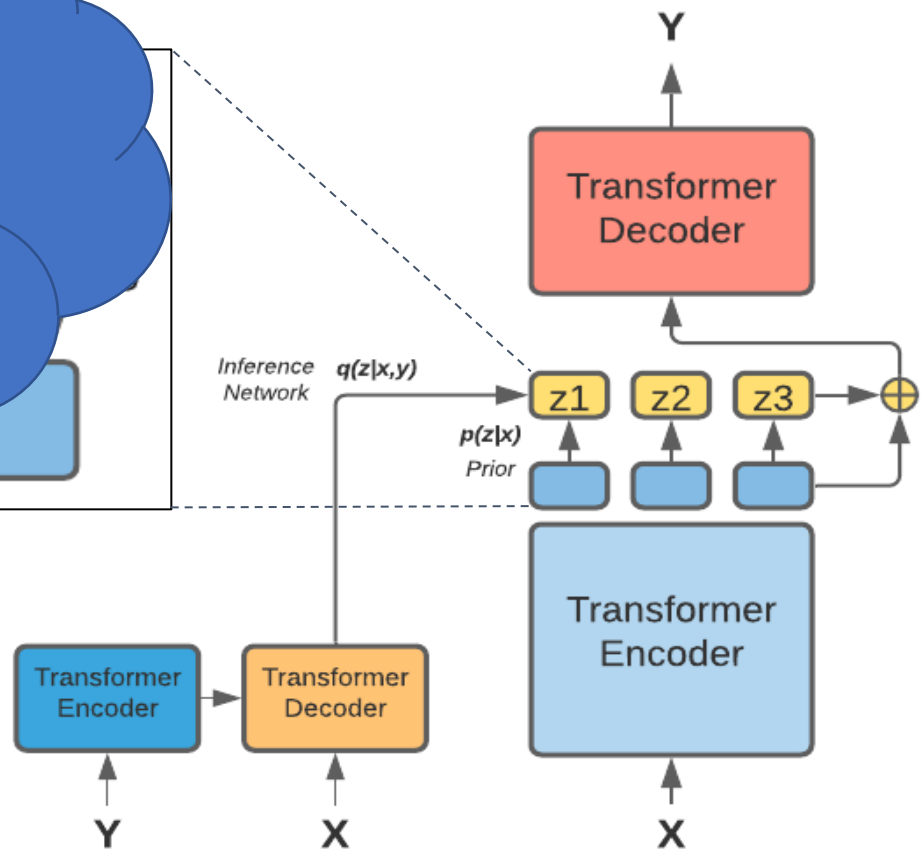
Example 13: Continuous VAEs (LaNMT)

- One way is to model the latent as standard continuous VAE w Gaussian prior.

$$\log p_\theta(y \mid x) = \log \int_z p_\theta(z \mid$$

However, the mean vectors are usually not the best choice.
The prior is TOO simple to capture complex dependences!

- Inference time:
  - with mean / samples from the prior

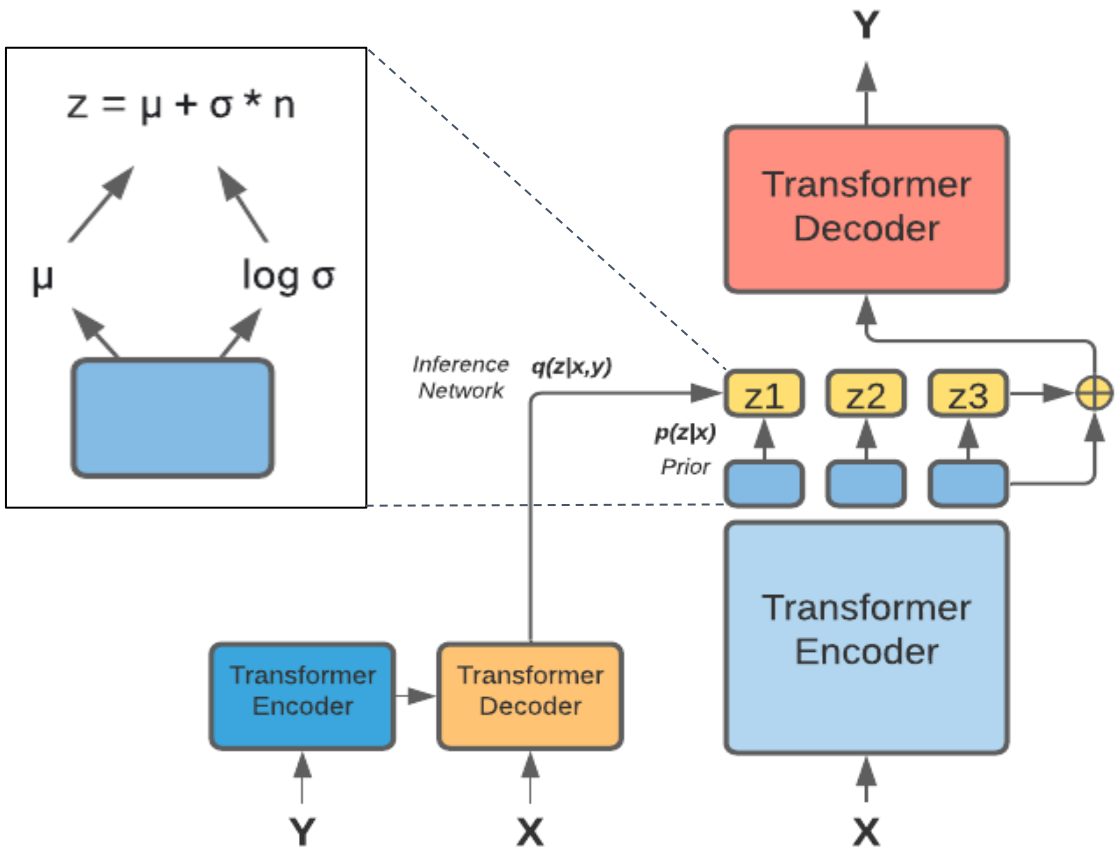


R. Shu, J. Lee, H. Nakayama, and K. Cho, "Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior," in AAAI, vol. 34, no. 05, 2020, pp. 8846–8853.

Problem of simple VAEs

- Spherical Gaussian prior is typically too weak to capture complex dependencies.
  - The mean vector of the prior is usually far from the posterior.
- Iterative inference with delta posterior

$$\mu_0 = \mathbb{E}_{p_\omega(z|x)}[z]$$
$$y_0 = \text{argmax}_y \log p_\theta(y|x, z = \mu_0)$$
for $t \leftarrow 1$ to T **do**
  $\mu_t = \mathbb{E}_{q_\phi(z|x,y_{t-1})}[z]$
  $y_t = \text{argmax}_y \log p_\theta(y|x, z = \mu_t)$
  if $y_t = y_{t-1}$ **then**
    break
output y_t



R. Shu, J. Lee, H. Nakayama, and K. Cho, "Latent-variable non-autoregressive neural machine translation with deterministic inference using a delta posterior," in AAAI, vol. 34, no. 05, 2020, pp. 8846–8853.

Example 14: Continuous VAEs + EBM

- As a follow-up, it is also possible to train an energy function to estimate gradients in the continuous space.
- The goal is to find z closer to the posterior more efficiently!

Algorithm 1: Inference for Latent Variable Models using Learned Gradients

Input : $\mathbf{x}$, α, θ, ψ
Output : $\hat{\mathbf{y}}$
let $\mathbf{z} = \mathbb{E}_{\mathbf{z} \sim p_\theta(\mathbf{z}|\mathbf{x})}[\mathbf{z}]$
while *termination condition not met,* **do**
  $\mathbf{z} = \mathbf{z} - \alpha \cdot (\nabla_{\mathbf{z}} E_\psi(\mathbf{z}; \mathbf{x}))$
end
$\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y}} \log p_\theta(\mathbf{y}|\mathbf{z}, \mathbf{x})$

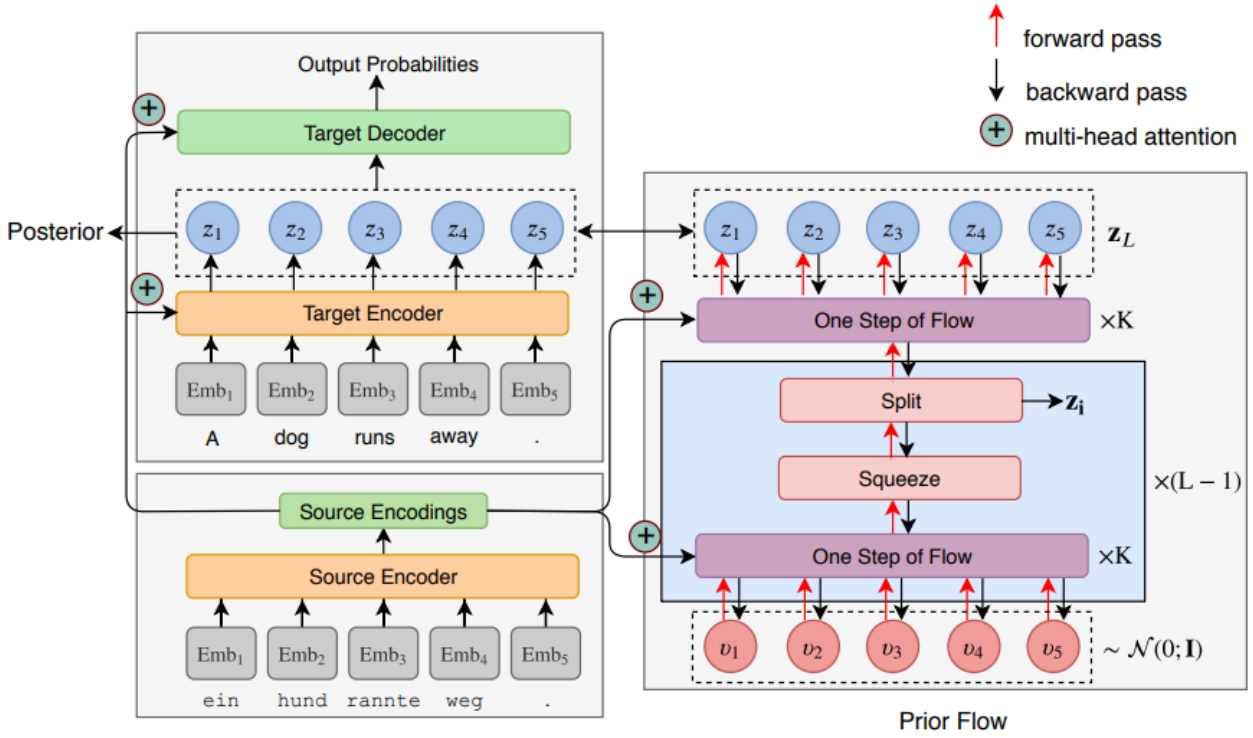
Meta Microsoft

Lee, J., Shu, R. and Cho, K., 2020. Iterative refinement in the continuous space for non-autoregressive neural machine translation. *EMNLP 2020* (pp. 1006-1015).

Example 15: FlowSeq

- Another direction is to actively learn a more complex prior

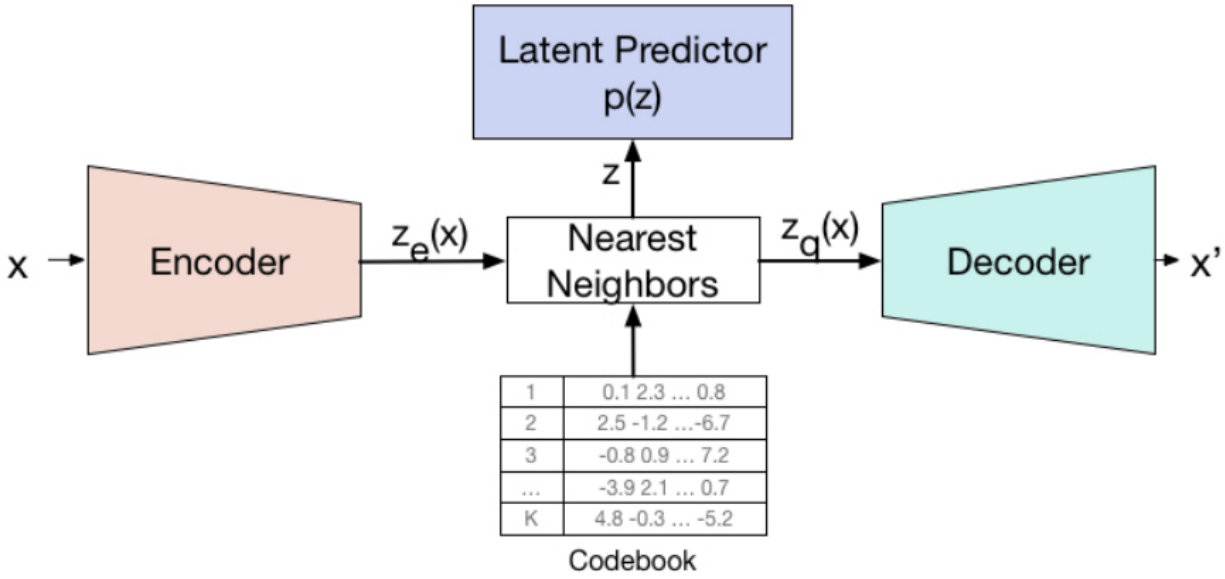
- Flowseq is a model which uses flow to map Gaussian noise to a complex prior.



X. Ma, C. Zhou, X. Li, G. Neubig, and E. Hovy, "Flowseq: Non-autoregressive conditional sequence generation with generative flow," in EMNLP-IJCNLP, 2019, pp. 4282–4292.

Example 16: Discrete VAEs (VQ-VAE)

- Training of VQ-VAE for NAR generation are usually two steps:
  - Learning the encoder and decoder
  - Learning the AR prior over the discrete symbols.
- Although the prior part is based on AR, it is typically shorter than the original length.



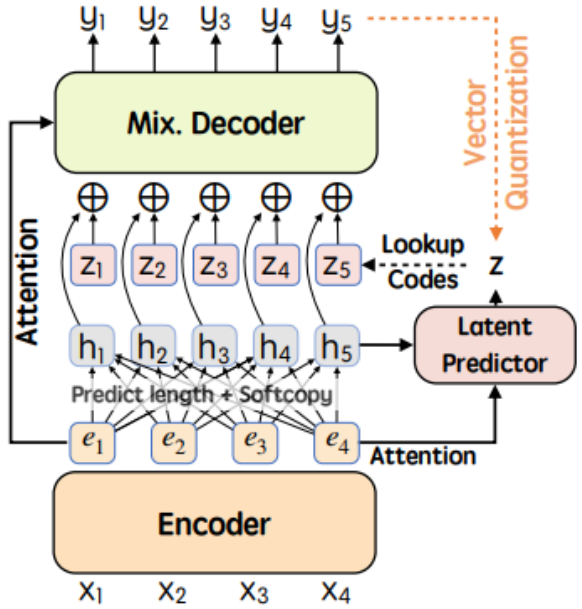
L. Kaiser, S. Bengio, A. Roy, A. Vaswani, N. Parmar, J. Uszkoreit, and N. Shazeer, "Fast decoding in sequence models using discrete latent variables," in ICML. PMLR, 2018, pp. 2390–2399.

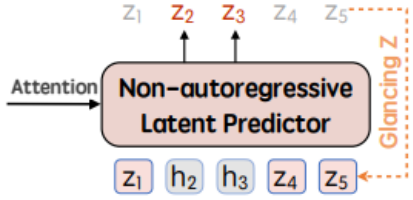
Roy, A., Vaswani, A., Neelakantan, A. and Parmar, N., 2018. Theory and experiments on vector quantized autoencoders. arXiv preprint arXiv:1805.11063.

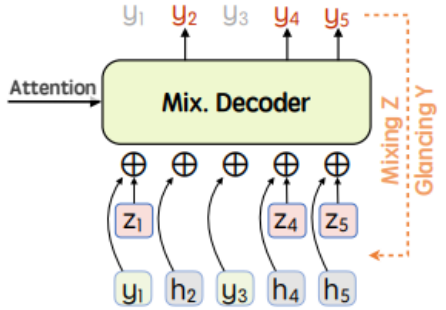
Example 17: Discrete VAEs (latent-GLAT)

- Recently, there are also papers applying NAR generation in the discrete codes as well.

- Compared to the typical framework that relies on AR model to generate the prior, it seems more efficient to generate them iteratively.







Meta Microsoft

Y. Bao, H. Zhou, S. Huang, D. Wang, L. Qian, X. Dai, J. Chen, and L. Li, “latent-glat: Glancing at latent variables for parallel text generation,” ACL, 2022.

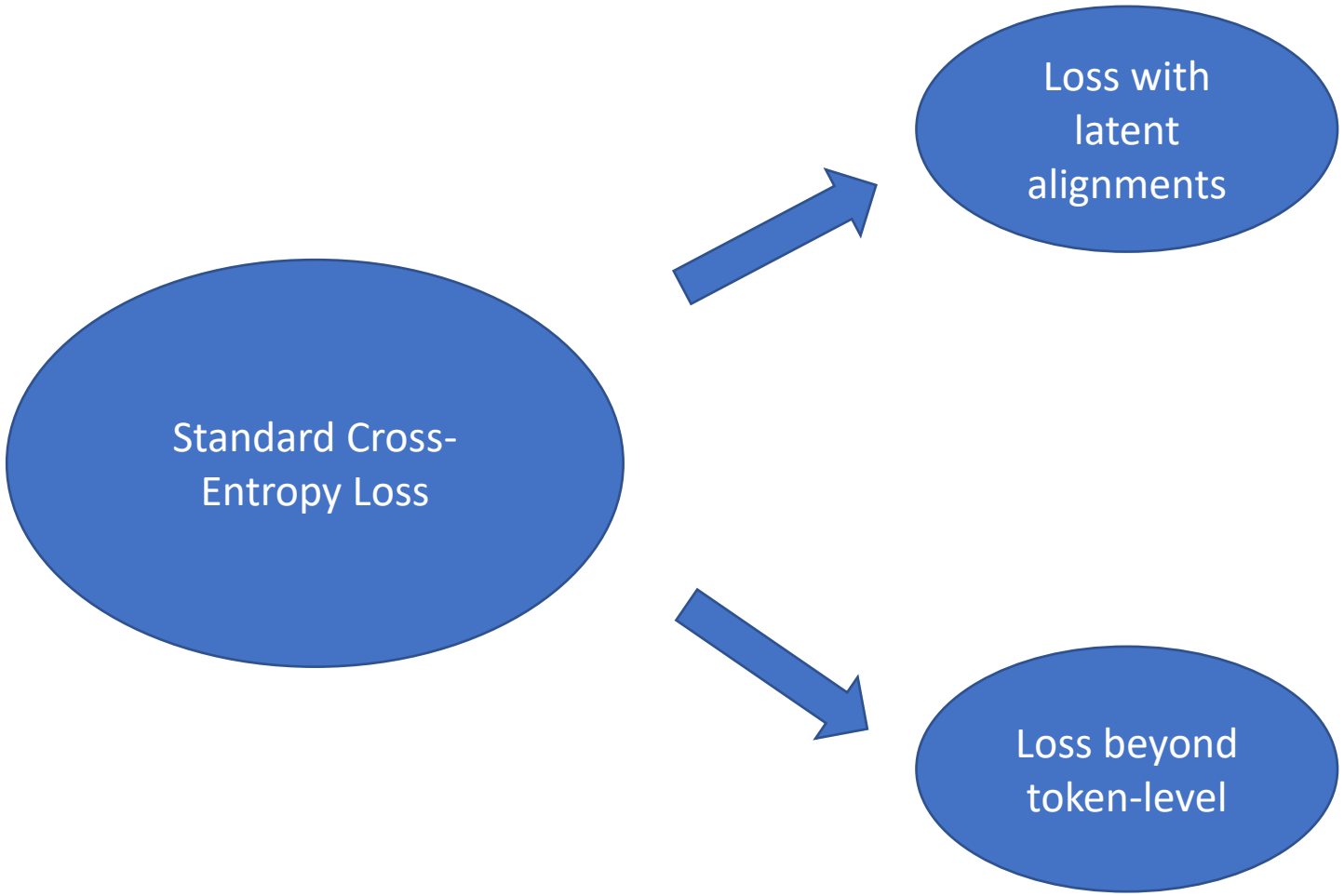
A principled goal

- The main challenge of NAR generation is "*failure of capturing the target side dependency*".

- Overall instructions:

Methods	Overall instructions
Model architecture	Improve the model's capability of capturing target-side dependency
Objective function	Provide learning signal that resolves uncertainty
Training data	Reduce the target-side dependency in the training
Learning paradigm	Better paradigm to ease the difficulty of learning
Inference techniques	Techniques that improve the final performance

Objective Functions



Cross Entropy

- Most NAR models apply the standard loss function (cross-entropy) for training

$$\mathcal{L}_{\text{CE}} = -\sum_{t=1}^{T} \log P(y_t|X;\theta)$$

- Standard cross entropy will compare tokens one by one between model prediction and ground-truth, which requires the length has to be correct!

Problems of Cross Entropy

- Over penalty for mis-alignment! (For AR model it is not an issue)

- Token-level CE does not consider the global correctness, which aggravates the weakness in capturing target side dependency.
- Maximum likelihood training will tend to cover **all possible mode**s, however, as we discussed, NAR model lack the ability of capture target side dependencies.

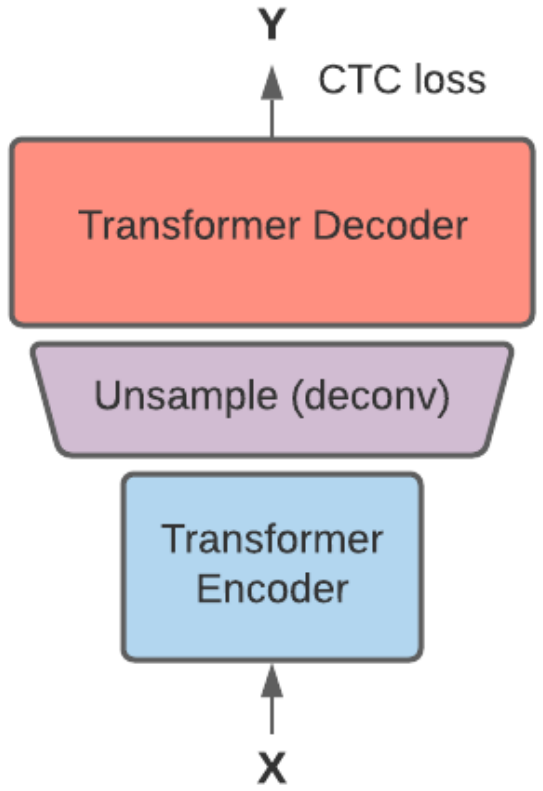
Example 18: CTC & AXE

- Inspired from ASR literature, we replace can replace objective with CTC that marginalizes all possible alignments
  - It assumes output always longer than the target, which in text generation we need to up-sample the decoder.
  - The final output is decoded by collapsing repetitive tokens.

$$\mathcal{L}_{\text{CTC}} = -\sum_{a \in \beta(y)} \prod_{i} p(a_i | x, \theta)$$

- Similarly, AXE is another loss using the monotonic alignment to improve cross entropy
  - Difference from CTC: (1) no need to up-sample; (2) using DP to find the best alignment instead of marginalizing the alignments.

$$\mathcal{L}_{\text{AXE}} = -\sum_{t=1}^{T} \log P_\alpha(y_t | X; \theta) - \sum_{k \notin \theta} P_k(\epsilon)$$

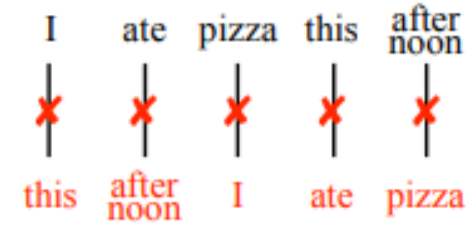


J. Libovický and J. Helcl, "End-to-end non-autoregressive neural machine translation with connectionist temporal classification," in *EMNLP*, 2018, pp. 3016–3021.

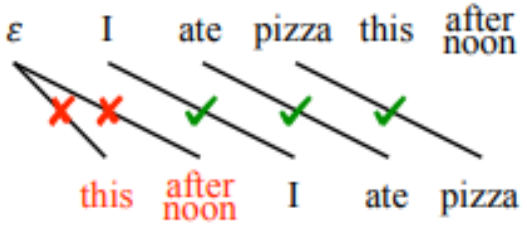
G. Marjan, V. Karpukhin, L. Zettlemoyer, and O. Levy, "Aligned cross entropy for non-autoregressive machine translation," in ICML. PMLR, 2020, pp. 3515–3523

Example 19: OAXE

- Both CTC and AXE can only resolve the mismatch when for “monotonic” alignment, while in practice re-ordering exists in real data, especially for tasks such as MT.

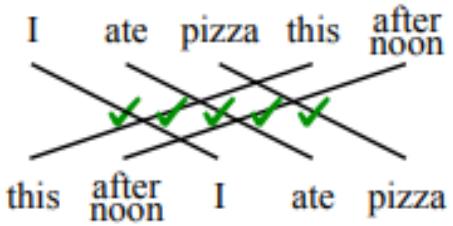


(a) Standard XE



(b) Aligned XE

- Order-agnostic cross-entropy (OAXE) loss applies the Hungarian algorithm to find the best possible alignment, which allows non-monotonic alignments in NAR generation.



(c) *Order-Agnostic* XE

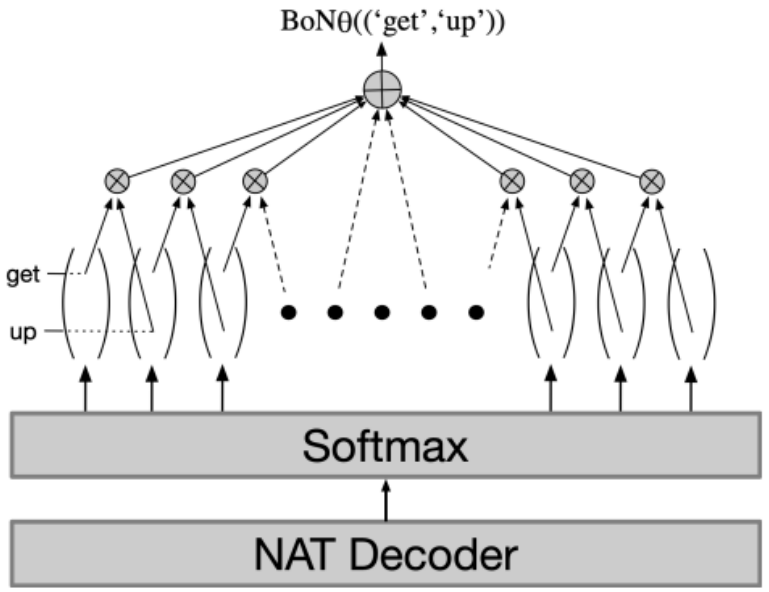
$$\mathcal{L}_{\text{OAXE}} = \underset{O^i \in \mathbf{O}}{\operatorname{argmin}} \left(-\log P(O^i|X) \right)$$

C. Du, Z. Tu, and J. Jiang, “Order-agnostic cross entropy for non-autoregressive machine translation,” in ICML. PMLR, 2021, pp. 2849–2859.

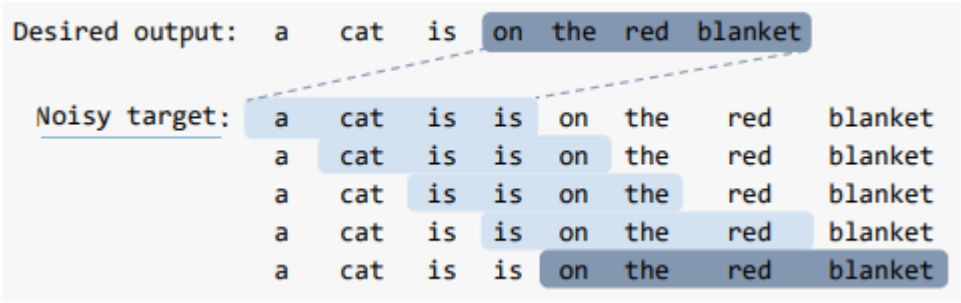
Meta Microsoft

Example 20: N-gram loss

- [a] N-gram level loss minimizes the Bag-of-Ngrams (BoN) difference between the model output and the reference sentence.



- [b] proposed “edit invariant sequence loss (EISL)” to replace CE, which also focuses on n-gram matching as convolution.



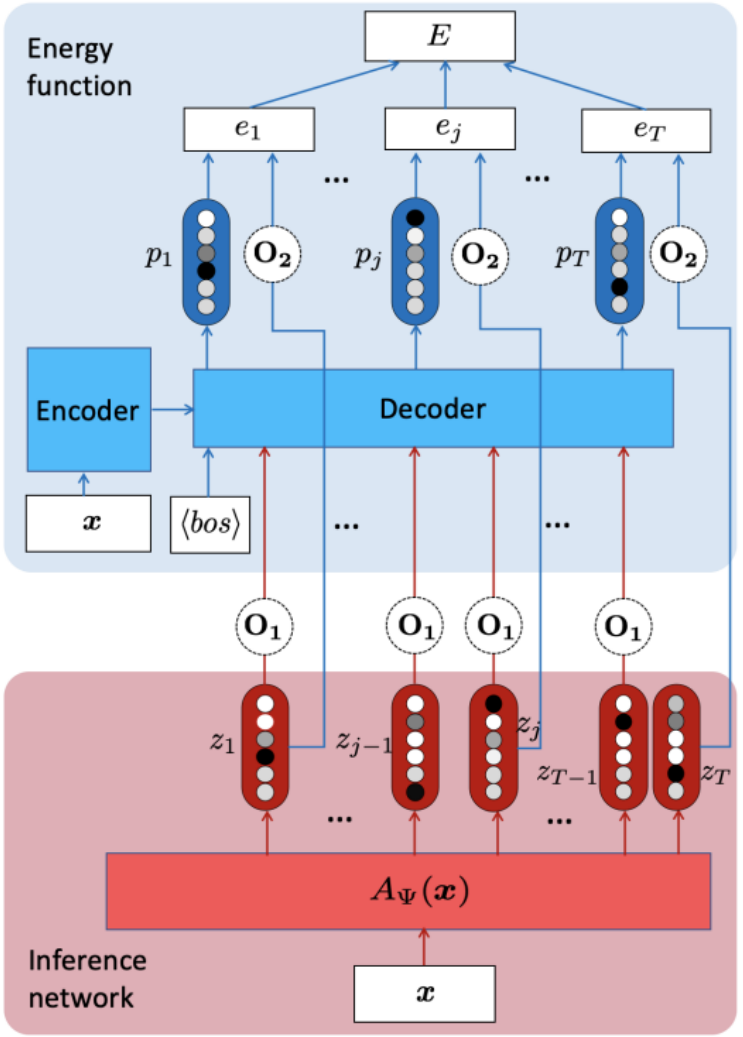
[a] C. Shao, J. Zhang, Y. Feng, F. Meng, and J. Zhou, “Minimizing the bag-of-ngrams difference for non-autoregressive neural machine translation,” in AAAI, vol. 34, no. 01, 2020, pp. 198–205.

[b] G. Liu, Z. Yang, T. Tao, X. Liang, Z. Li, B. Zhou, S. Cui, and Z. Hu, “Don’t take it literally: An edit-invariant sequence loss for text generation,” *arXiv preprint arXiv:2106.15078*, 2021.

Meta Microsoft

Example 21: ENGINE

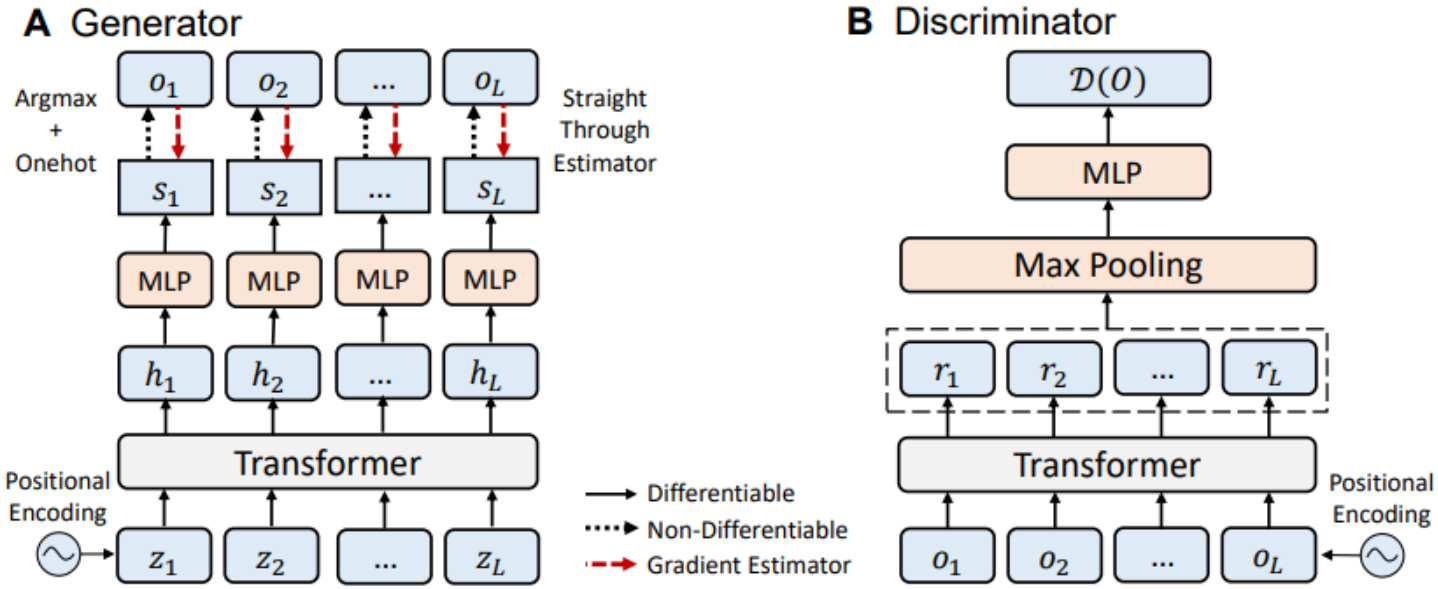
- We can further go from n-gram level to entire sequence level with a learned energy function.
- In ENGINE, a pretrained AR model is used as an energy score for evaluating the output globally.
- Gradient is passed via straight-through / softmax during training.



L. Tu, R. Y. Pang, S. Wiseman, and K. Gimpel, "Engine: Energybased inference networks for non-autoregressive machine translation," in ACL, 2020, pp. 2819–2826

Example 22: GANs

- Extending similar ideas from learning with a pretrained energy function, it is also possible to learn jointly!
- For example, it might be useful to train text-GAN in NAR settings.



Huang, F., Guan, J., Ke, P., Guo, Q., Zhu, X. and Huang, M., 2020. A text GAN for language generation with non-autoregressive generator.

A principled goal

- The main challenge of NAR generation is "*failure of capturing the target side dependency*".

- Overall instructions:

Methods	Overall instructions
Model architecture	Improve the model's capability of capturing target-side dependency
Objective function	Provide learning signal that resolves uncertainty
Training data	Reduce the target-side dependency in the training
Learning paradigm	Better paradigm to ease the difficulty of learning
Inference techniques	Techniques that improve the final performance

Sequence-level Knowledge Distillation (KD)

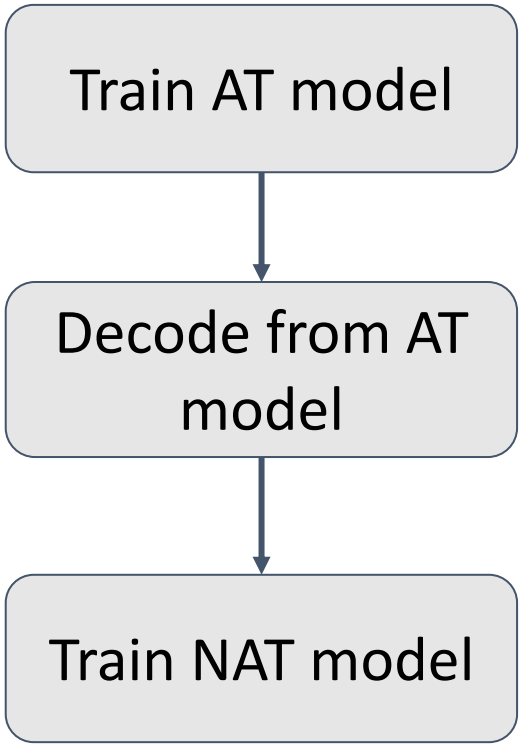
- Knowledge distillation
  - Use the soft logit probability prediction from a teacher model to teach a student model

- Knowledge distillation in sequence tasks
  - Token-level knowledge distillation
    - The logit probability of each token is used to teach student
  - Sequence-level knowledge distillation
    - The sequence generated by the teacher model is used to teach student

Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network[J]. arXiv preprint arXiv:1503.02531, 2015, 2(7).
Kim Y, Rush A M. Sequence-level knowledge distillation[J]. arXiv preprint arXiv:1606.07947, 2016.

Sequence-level Knowledge Distillation (KD)

- Distillation at sequence level can significantly improve NAT systems:
  - 1. train teacher autoregressive model
  - 2. replace targets with teacher's prediction
  - 3. train NAT models on the synthetic pairs



- Almost all NAT systems benefit from KD:

	w/o distillation	w/ distillation
Vanilla NAT (Gu et al, 2017)	11.4	19.5 (+8.1)
FlowSeq (Ma et al, 2019)	18.6	21.7 (+3.1)
LevT (Gu et al, 2019)	25.2	26.9 (+1.7)

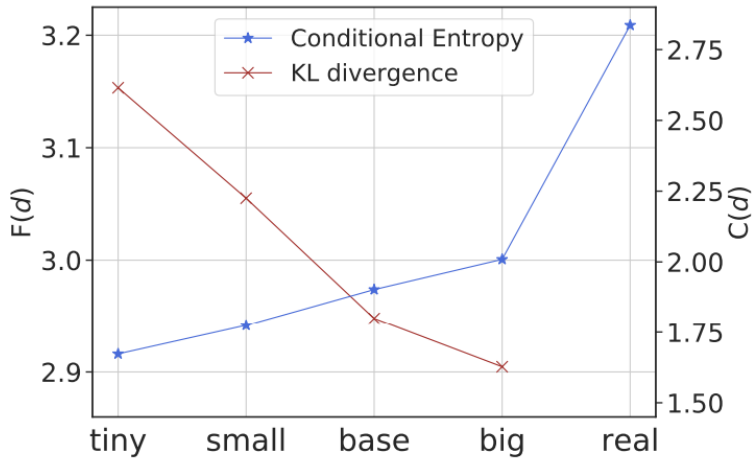
Why KD works?

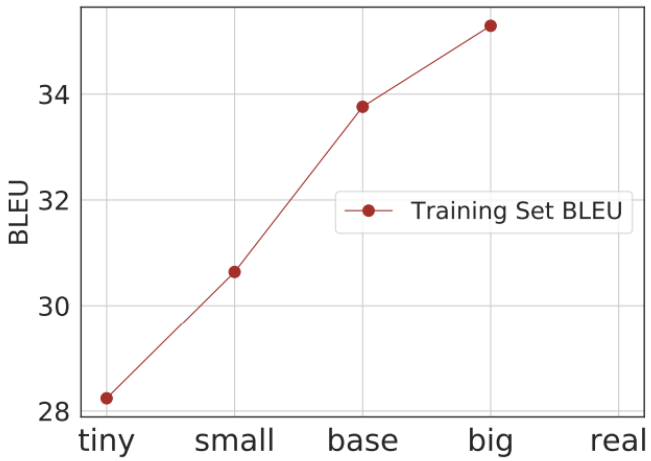
- Simplify the data distribution of target data, and thus reduce target data dependency
  - Data is complicated, the teacher (AR) model only learn the most dominated distribution from the data
  - During beam search/sampling, the most probability data pattern is generated instead of the whole data distribution
  - E.g., “Thank You” → “Vielen Dank” or “Danke” , after distillation may only have “Danke” in German, reduce the multi-modality of target data

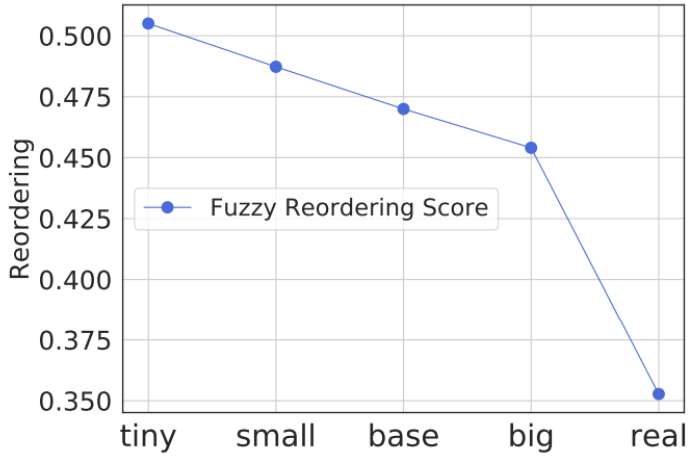
C. Zhou, J. Gu, and G. Neubig, “Understanding knowledge distillation in non-autoregressive machine translation,” in ICLR, 2019

How KD works?

- Quantitive results: A cross-entropy based measure is used for dataset complexity and prepare distilled data from different teacher models.



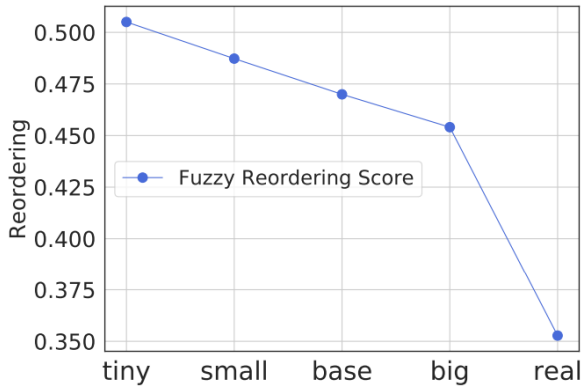


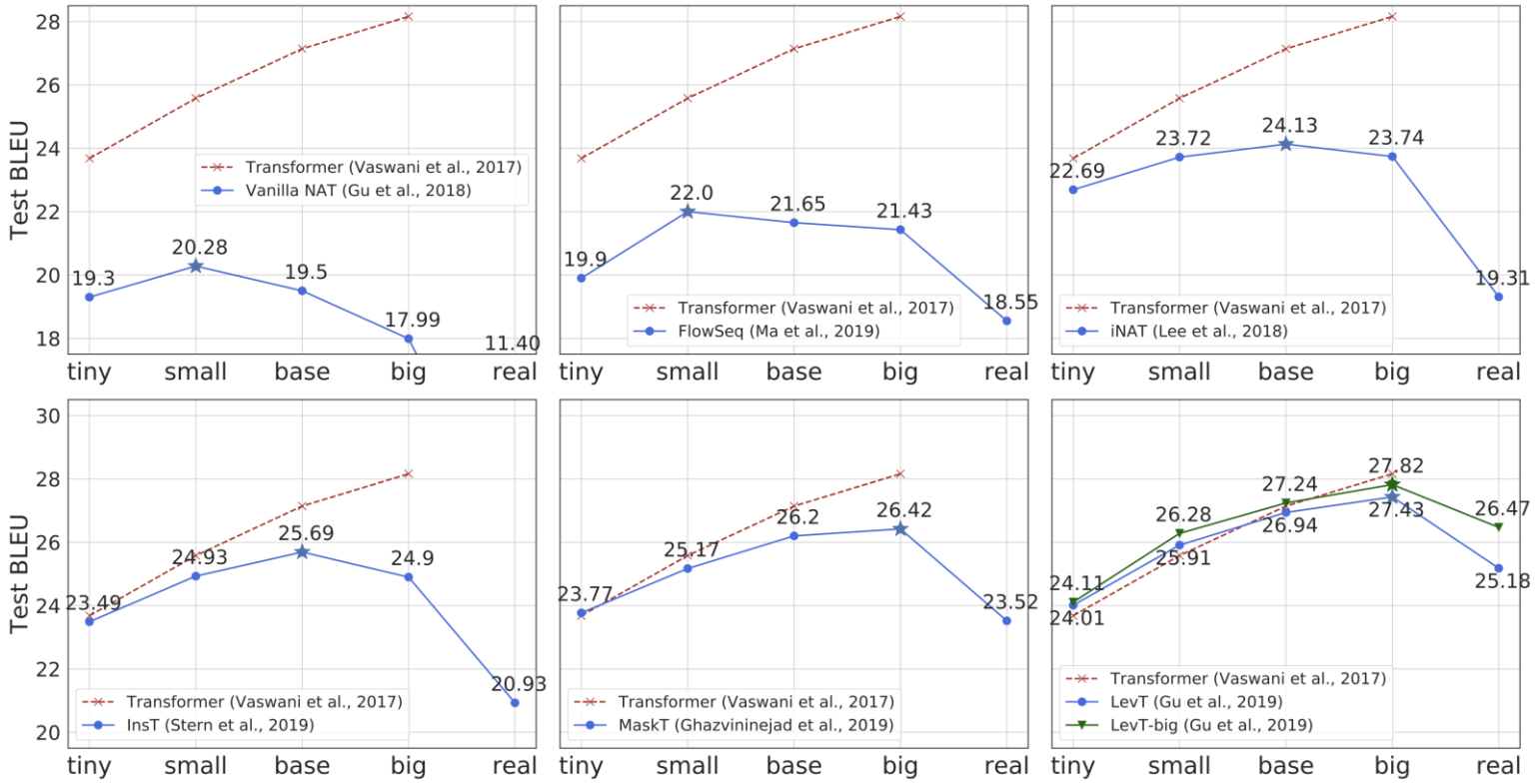


C. Zhou, J. Gu, and G. Neubig, “Understanding knowledge distillation in non-autoregressive machine translation,” in ICLR, 2019

How KD works?

- NAR models perform the best when the data complexity matches the model's capacity





C. Zhou, J. Gu, and G. Neubig, "Understanding knowledge distillation in non-autoregressive machine translation," in ICLR, 2019

Problems of KD

- KD makes the training pipeline too long
  - You always need to first train an AR model as the teacher
- The capacity of NAT models should be correlated with the complexity of the distilled dataset
- KD hurts the lexical choice especially on low-frequency words.

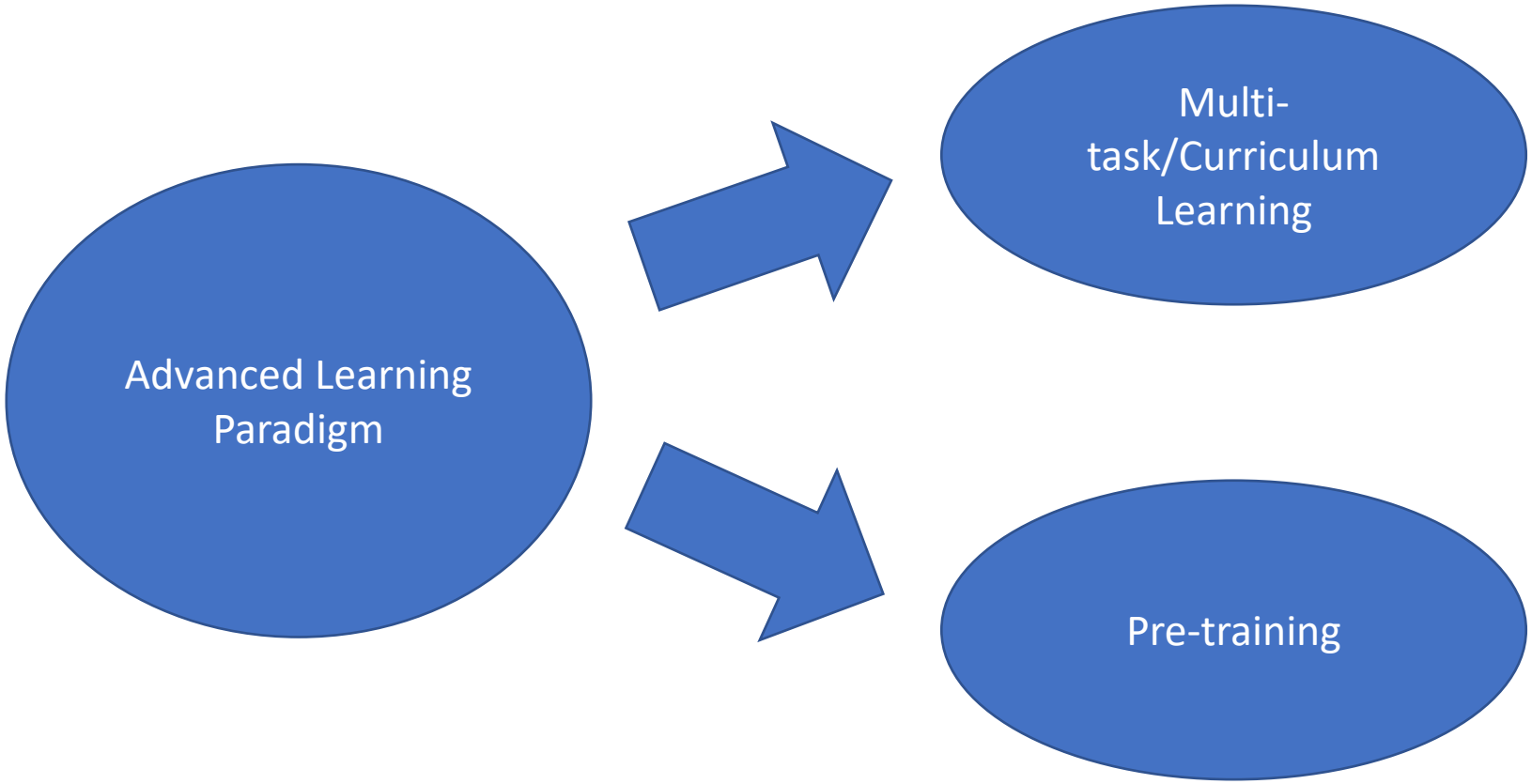
A principled goal

- The main challenge of NAR generation is "*failure of capturing the target side dependency*".

- Overall instructions:

Methods	Overall instructions
Model architecture	Improve the model's capability of capturing target-side dependency
Objective function	Provide learning signal that resolves uncertainty
Training data	Reduce the target-side dependency in the training
Learning paradigm	Better paradigm to ease the difficulty of learning
Inference techniques	Techniques that improve the final performance

Learning paradigm



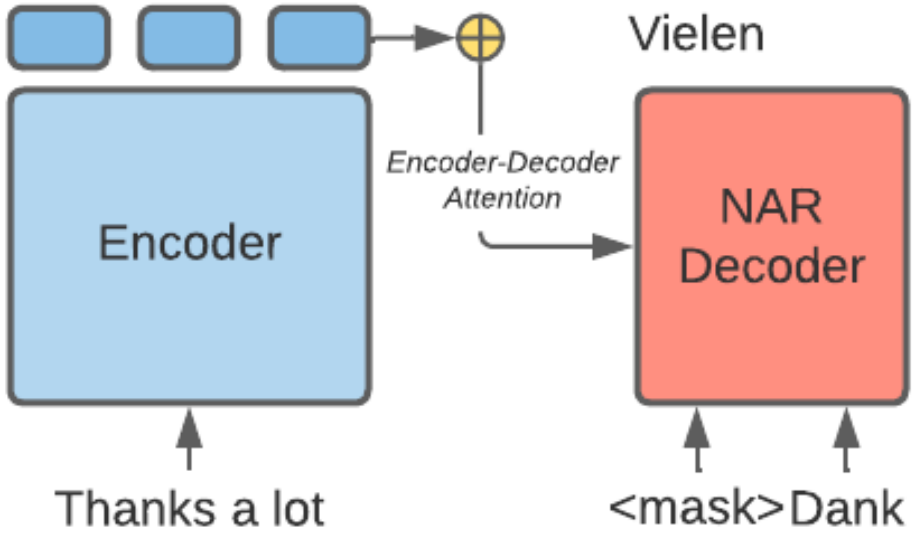
Multi-task and Curriculum Learning

- Directly learning NAR models over a given dataset is hard, and easily gets stuck into bad optimum.
- NAR models can effectively benefit from learning multiple (easier) tasks jointly, with a better curriculum
- Easier tasks:
  - Learning with partially masked input
  - Learning with partially autoregressive generation
  - Learning on smaller granularity

Example 23: GLAT

- In the original Mask-Predict, it is found that when training on sequences with different mask ratio, the performance of single iteration model is better than those trained directly on “full-mask” settings.

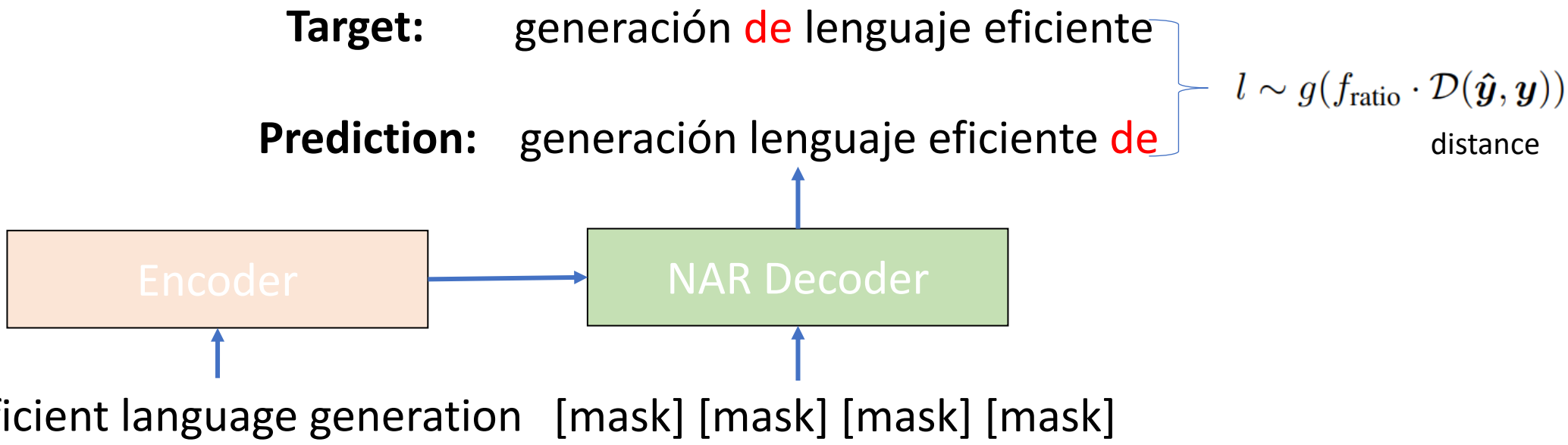
- Such results in fact indicate that, when training NAR models, it is beneficial to ease the difficulty by jointly training on glanced targets.



M. Ghazvininejad, O. Levy, Y. Liu, and L. Zettlemoyer, “Maskpredict: Parallel decoding of conditional masked language models,” in EMNLP-IJCNLP, 2019, pp. 6112–6121.

Example 23: GLAT

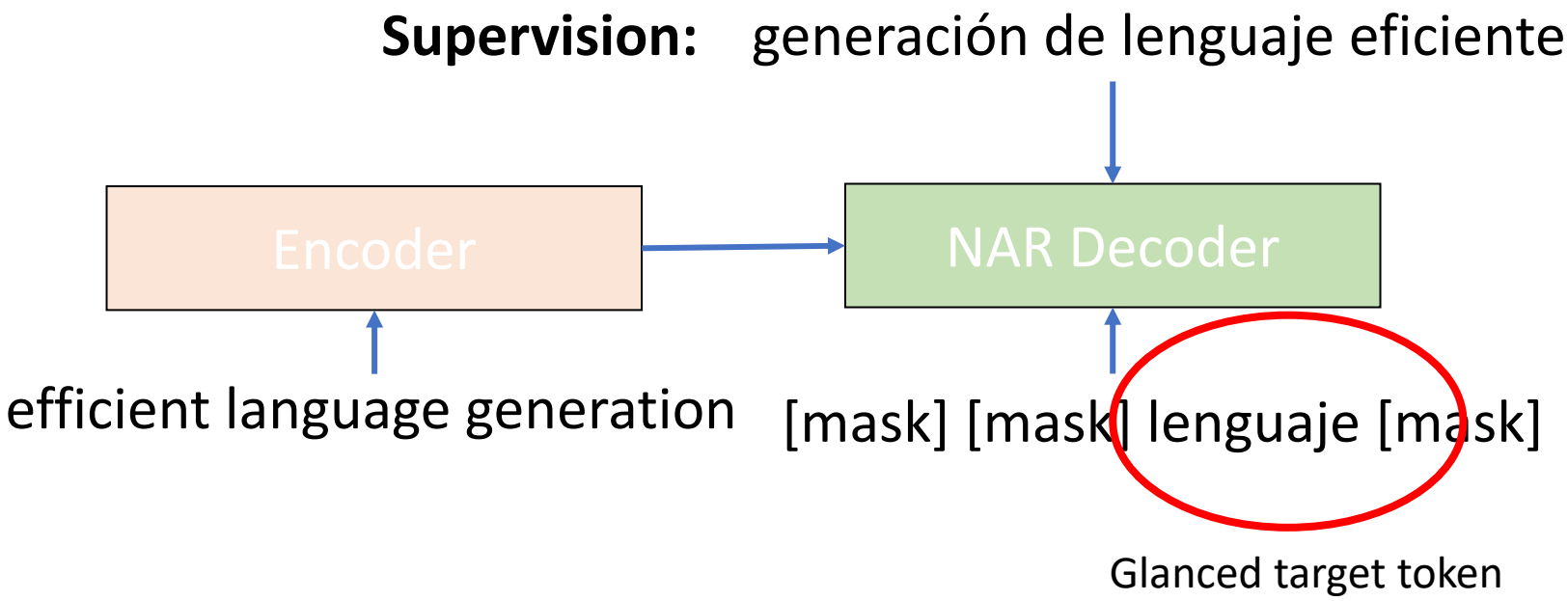
- Compared to Mask-Predict where the glancing is completely random, it is possible to design a curriculum: *We can control the number of tokens being glanced based on the training progress*.



L. Qian, H. Zhou, Y. Bao, M. Wang, L. Qiu, W. Zhang, Y. Yu, and L. Li, "Glancing transformer for non-autoregressive neural machine translation," in ACL-IJCNLP, 2021, pp. 1993–2003.

Example 23: GLAT

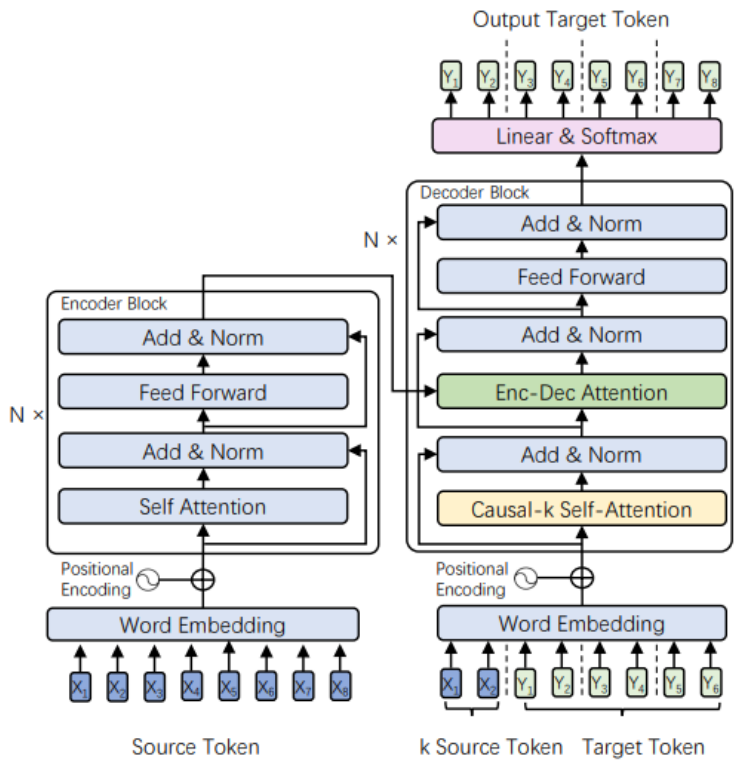
- Compared to Mask-Predict where the glancing is completely random, it is possible to design a curriculum: *We can control the number of tokens being glanced based on the training progress*.

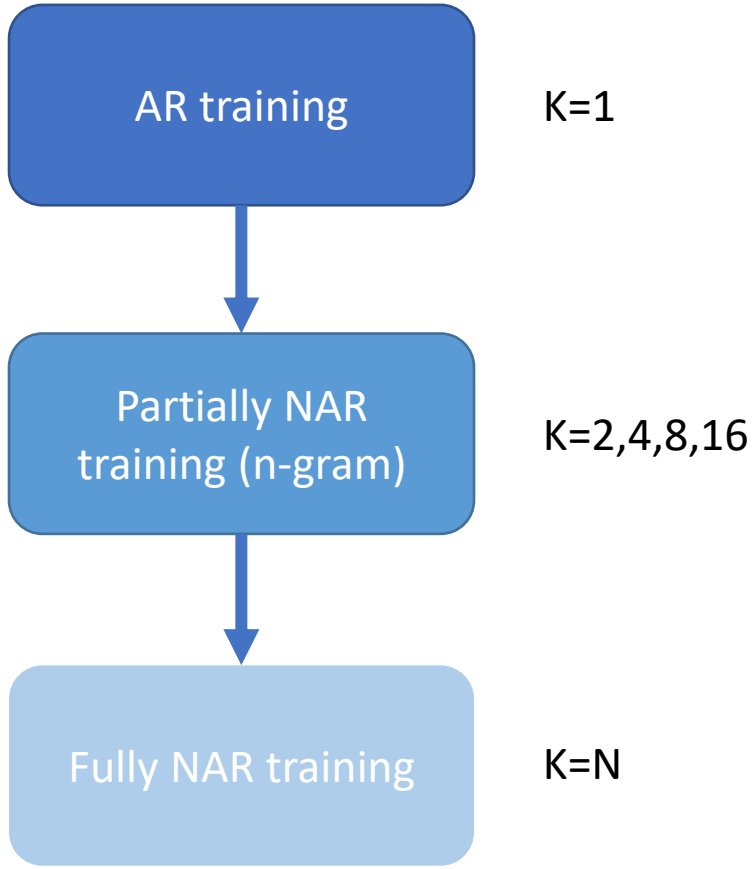


L. Qian, H. Zhou, Y. Bao, M. Wang, L. Qiu, W. Zhang, Y. Yu, and L. Li, "Glancing transformer for non-autoregressive neural machine translation," in ACL-IJCNLP, 2021, pp. 1993–2003.

Example 24: Task-level curriculum learning

- Task-level curriculum learning to shift the training strategy from AR to SAR gradually, finally to NAR generation.





$$P(y|x) = \prod_{t=0}^{\lfloor N/k \rfloor} \prod_{j=1}^{k} P(y_{tk+j} | y_{<tk+1}, x; \theta)$$

J. Liu, Y. Ren, X. Tan, C. Zhang, T. Qin, Z. Zhao, and T.-Y. Liu, "Task-level curriculum learning for non-autoregressive neural machine translation," in *IJCAI*, 2021, pp. 3861–3867.

Meta Microsoft

Example 25: Multi-granularity Curriculum Learning

- Curriculum learning can also be performed on data with different translation granularities.
- Training data is divided into words, phrases, and sentences. A progressive multi-granularity training strategy is used to train the model from easy to hard

	Source	Targets
Word	bank	银行 岸 储库
Phrase	hollow structural	中空 结构 空心 的 结构 镂空 结构
Sentence	He is very good at English.	他 英文 很 好。 他 非常 擅长 英语 。 他的 英语 水平 很 高 。

L. Ding, L. Wang, X. Liu, D. F. Wong, D. Tao, and Z. Tu, "Progressive multi-granularity training for non-autoregressive translation," in Findings of ACL-IJCNLP, 2021, pp. 2797–2803.

Improve NAR generation with Pre-training

- Like typical AR sequence generation, NAR models can also be benefitted by fine-tuning from a pretrained models, especially from "Encoder-only" pretraining such as BERT.

Y. Su, D. Cai, Y. Wang, D. Vandyke, S. Baker, P. Li, and N. Collier, "Non-autoregressive text generation with pre-trained language models," in EACL, 2021, pp. 234–243.

P. Li, L. Li, M. Zhang, M. Wu, and Q. Liu, "Universal conditional masked language pre-training for neural machine translation," ACL, 2022.

A principled goal

- The main challenge of NAR generation is "*failure of capturing the target side dependency*".

- Overall instructions:

Methods	Overall instructions
Model architecture	Improve the model's capability of capturing target-side dependency
Objective function	Provide learning signal that resolves uncertainty
Training data	Reduce the target-side dependency in the training
Learning paradigm	Better paradigm to ease the difficulty of learning
Inference techniques	Techniques that improve the final performance

Length beam / Noisy Parallel Decoding

- First predict a target length L, and then construct a length beam [L-B, L+B] with beam size 2B+1

- Use NAR model (encoder one time, decode 2B+1 times) to generate sentences with these lengths

- For latent-variable based models, we can further sample more by sampling multiple latent codes, and then choose the best one with highest model scores.

AR model re-ranking

- After obtaining multiple candidates, we can use AR model to give a probability score to help select a better candidates

- Usually weighted combine the probability score from both AR and NAR for final reranking

Combine with n-gram LM

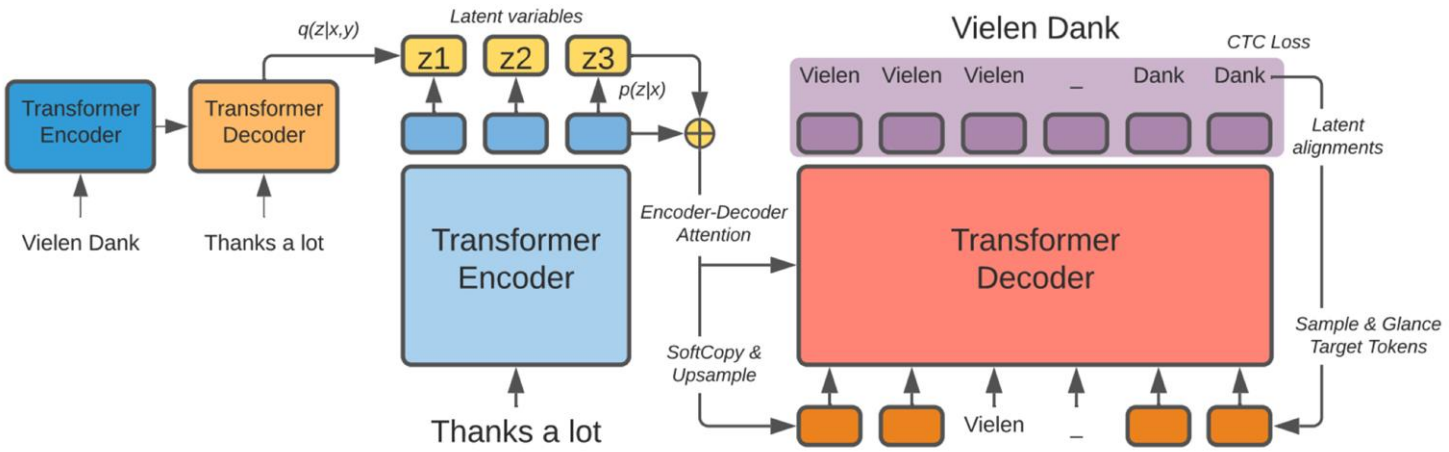
- Similar to the common practice in ASR, it is also useful to combine n-gram LM into NAR generation, while maintaining overall speed-up.

	Configuration	BLEU (Δ)	BP	$\mathcal{L}_1^{\text{GPU}}$ (Speed-up)		$\mathcal{L}_1^{\text{CPU}}$ (Speed-up)	
AT	*big* (teacher)	21.07	0.920	345 ms	1.0 ×	923 ms	1.0 ×
	base	18.91	0.908	342 ms	1.0 ×	653 ms	1.4 ×
	base (12-1)	15.47	0.806	152 ms	2.3 ×	226 ms	4.0 ×
	base (12-1) + KD	18.76	0.887	145 ms	2.4 ×	254 ms	3.6 ×
NAT	KD + CTC	16.93 (+0.00)	0.828	17.3 ms	19.9 ×	84 ms	11.0 ×
	KD + CTC + VAE	18.73 (+1.80)	0.862	16.4 ms	21.0 ×	83 ms	11.1 ×
	w. *BeamSearch20*	19.80 (+2.87)	0.958	28.5 ms	12.1 ×	99 ms	9.3 ×
	w. *BeamSearch20 + 4-gram LM*	**21.41 (+4.48)**	0.954	31.5 ms	11.0 ×	106 ms	8.7 ×
	w. *NPD5*	18.88 (+1.95)	0.866	34.9 ms	9.9 ×	313 ms	2.9 ×
	w. *NPD5 + BeamSearch20 + 4-gram LM*	**21.84 (+4.91)**	0.962	57.6 ms	6.0 ×	284 ms	3.2 ×

Gu, J. and Kong, X., 2021. Fully Non-autoregressive Neural Machine Translation: Tricks of the Trade. *ACL 2021.*

A practical system of fully NAR model for machine translation

Methods	Distillation	Latent Variables	Latent Alignments	Glancing Targets
What it can do?	simplifying the training data	model any types of dependency in theory	handling token shifts in the output space	ease the difficulty of learning hard examples
What it cannot?	uncertainty exists in the teacher model	constrained by the modeling power of the used latent variables	unable to model non-monotonic dependency, e.g. reordering	training / testing phase mismatch
Potential issues	sub-optimal due to the teacher's capacity	difficult to train; posterior collapse	decoder inputs must be longer than targets	difficult to find the optimal masking ratio

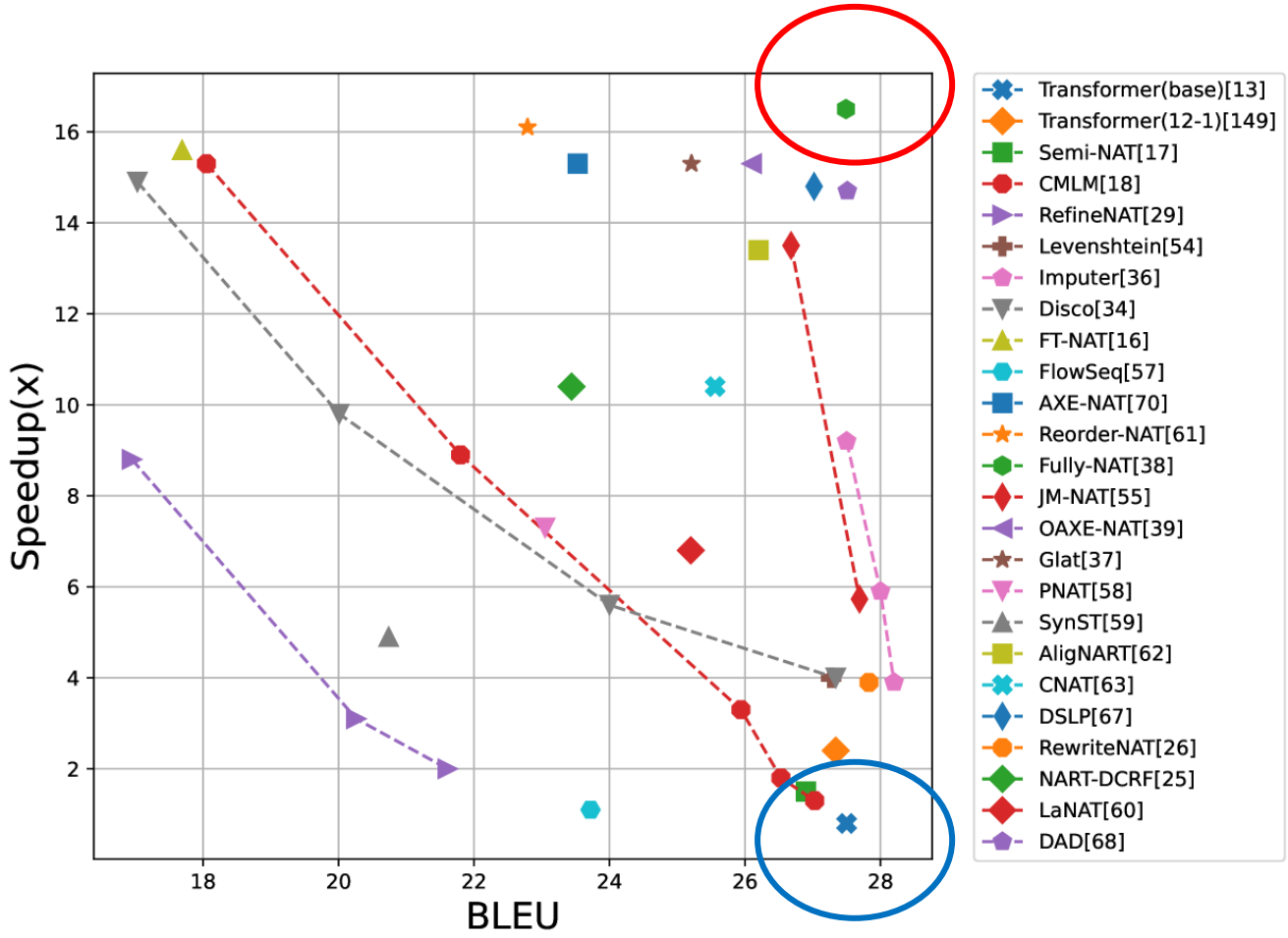


Gu, J. and Kong, X., 2021. Fully Non-autoregressive Neural Machine Translation: Tricks of the Trade. *ACL 2021.*

Final System

- Combine the technique of KD with proposed model, we can finally close the performance gap between autoregressive models.

- In the meantime, the fully NAT model maintains over x17 speed-up.

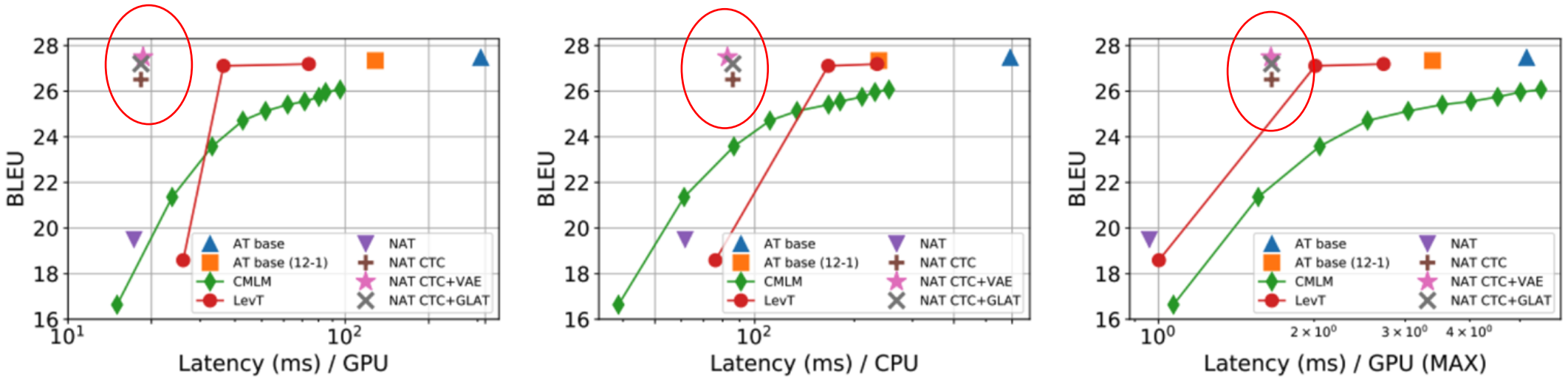


Gu, J. and Kong, X., 2021. Fully Non-autoregressive Neural Machine Translation: Tricks of the Trade. *ACL 2021.*

Final System

Speed vs Quality Trade-off

- Iterative NAT models (LevT and CMLM) require multiple iterations to achieve reliable performance with the sacrifice of latency
- Speed advantage of fully NAT models shrinks when parallelism is constrained



Gu, J. and Kong, X., 2021. Fully Non-autoregressive Neural Machine Translation: Tricks of the Trade. *ACL 2021.*

Final System

How important all these techniques?

- The combination without KD has a clear performance drop compared to the one with KD
- CTC-based model obtains better accuracy through marginalizing all valid alignments
- The model with GLAT is more superior to the one with the RND training method, however it performs similarly with VAEs

KD	AXE	CTC	VAE	RND	GLAT	BLEU
						11.40
✓						19.50
	✓					16.59
✓	✓					21.66
		✓				18.18
✓		✓				**26.51**
		✓	✓			23.58
✓	✓		✓			22.19
✓		✓	✓			**27.49**
✓	✓			✓		22.74
✓	✓				✓	24.67
✓		✓		✓		26.16
		✓			✓	21.81
✓		✓			✓	**27.20**
✓		✓	✓		✓	**27.21**

Gu, J. and Kong, X., 2021. Fully Non-autoregressive Neural Machine Translation: Tricks of the Trade. *ACL 2021.*

Meta Microsoft

Summary of Part II

- The main challenge of NAR generation is "*failure of capturing the target side dependency*".

- Overall instructions:

Methods	Overall instructions
Model architecture	Improve the model's capability of capturing target-side dependency
Objective function	Provide learning signal that resolves uncertainty
Training data	Reduce the target-side dependency in the training
Learning paradigm	Better paradigm to ease the difficulty of learning
Inference techniques	Additional techniques that improve the final performance

Outline

- Part I: Introduction (Jiatao Gu)
- Part II: Methods (Jiatao Gu)
- Part III: Applications (Xu Tan)
- Part IV: Open Problems (Xu Tan)

https://github.com/NAR-tutorial/acl2022

Non-Autoregressive Sequence Generation

(Part III: Applications)

Xu Tan
Microsoft Research Asia

Meta Microsoft

Outline

- Overview of NAR generation tasks in NLP/Speech/CV
  - Target-target vs target-source dependency
- Key tasks
  - Neural machine translation
  - Text error correction
  - Speech to text recognition
  - Text to speech/Singing voice synthesis
  - Image generation
- Summary of NAR applications
  - Benefits of NAR for different tasks
  - Addressing target-target dependency (model multimodal distributions)
  - Addressing target-source dependency (learn source-target alignment)
  - Data difficulty vs model capacity
  - Streaming vs NAR, AR vs Iterative NAR

Target-target vs target-source dependency

- Tradeoff in dependency
  - Target-target dependency: dependency among target tokens
  - Target-source dependency: dependency on source tokens
  - If target-target is stronger than target-source dependency → more information is needed from target tokens instead of source tokens → NAR is more difficult
- Connection to multi-modality
  - Multi-modality: P(x|y) is not single-modal, not one-one mapping
    - e.g., “Thank You” → “Vielen Dank” or “Danke”
  - If target-source dependency dominates, then P(x|y) is more like single-modal, a source token will have one definite translation
  - If target-target dependency dominates, then P(x|y) will be like multi-modal, a source token will have multiple token translations

Modality	Task	Source	Target	Target-Source vs Target-Target	Difficulty of NAR
Text Generation	Neural Machine Translation	Source language	Target language	$\approx$?	****
	Text Summarization	Long text	Short Summarization	$>$	***
	Text Error Correction	Error Text	Correct Text	$>$	***
	Text Style Transfer	Source Text	Target text	$>$	***
	Dialogue Generation	Dialogue	Response	$<$	*****
	Speech Recognition	Speech	Text	$\geq$?	***
Speech Generation	Text to Speech	Text	Speech	$>$	***
	Singing Voice Synthesis	Score	Singing Voice	$>>$	***
	Voice Conversion	Source Voice	Target Voice	$\gg$	**
	Speech Enhancement	Noisy Speech	Clean Speech	$\gg$	*
Image Generation	Pixel Generation	Class ID	Image Pixel	-	*
	Discrete Token Generation		Image Token	-	**

The values in the last two columns are just for reference

Target-source dependency

Dependency Type	Task	Alignment
Target has correspondence with source	Speech Enhancement	Alignment inherently
	Voice Conversion	
	Text to Speech	Need alignment
	Singing Voice Synthesis	Music score alignment
	Speech Recognition	CTC Alignment
Target is a minor change of source	Text Error Correction	Locate the minor changes
	Text Style Transfer	Content unchange and style change
Target is a translation of source	Machine Translation	Alignment through attention
Target is implicitly correlated to source	Dialogue Generation	Absorb the high-level abstraction
	Image Generation	Category information

Target-target dependency

Dependency Type	Task	Description
Text	Machine Translation	Discrete tokens in languages are contextualized, explained mutually. Language tokens have strong mutual dependency
	Text Summarization	
	Text Error Correction	
	Text Style Transfer	
	Dialogue Generation	
	Speech Recognition	
Speech and Image	Text to Speech	For continuous signal like speech/sound/image, they depends on the concept, like speech frames depend on a word, image pixel depend on a class. Maybe weaker mutual dependency
	Singing Voice Synthesis	
	Image Generation	

Outline

- Overview of NAR generation tasks in NLP/Speech/CV
  - Target-target vs target-source dependency
- Key tasks
  - Neural machine translation
  - Text error correction
  - Speech to text recognition
  - Text to speech/Singing voice synthesis
  - Image generation
- Summary of NAR applications
  - Benefits of NAR for different tasks
  - Addressing target-target dependency (model multimodal distributions)
  - Addressing target-source dependency (learn source-target alignment)
  - Data difficulty vs model capacity
  - Streaming vs NAR, AR vs Iterative NAR

Modality	Task	Source	Target	Target-Source vs Target-Target	Difficulty of NAR
Text Generation	**Neural Machine Translation**	Source language	Target language	$\approx$?	****
	Text Summarization	Long text	Short Summarization	$>$	***
	Text Error Correction	Error Text	Correct Text	$>$	***
	Text Style Transfer	Source Text	Target text	$>$	***
	Dialogue Generation	Dialogue	Response	$<$	*****
	Speech Recognition	Speech	Text	$\geq$?	***
Speech Generation	Text to Speech	Text	Speech	$>$	***
	Singing Voice Synthesis	Score	Singing Voice	$>>$	***
	Voice Conversion	Source Voice	Target Voice	$\gg$	**
	Speech Enhancement	Noisy Speech	Clean Speech	$\gg$	*
Image Generation	Pixel Generation	Class ID	Image Pixel	-	*
	Discrete Token Generation		Image Token	-	**

The values in the last two columns are just for reference

Text error correction

- Text errors: writing assistant, search engine, speech recognition, optical character recognition, etc.
- Input: text sequence with errors; Output: corrected text sequence
- **Naïve NAR solution usually fails**
- Challenges
  - Error detection and error correction, to avoid under/over-correction
  - Few modifications in text error correction (e.g., 10% WER)

Design with inductive bias, instead of black-box end-to-end learning in NAR!

Text error correction

- How to detect errors and correct errors?

- Implicit way
  - Target-source attention
  - CTC (connectionist temporal classification): duplicate source tokens multiple times, and use CTC loss $P(y|x) = \sum_{z \in \varphi(y)} P(z|x)$
- Explicit way
  - Detect the exact error patterns of insertion/deletion/substitution
  - Or use duration as an approximation: 0 for deletion, 1 for substitution or nochange, 2 or more for insertion
  - Expand the source tokens to the length of target tokens according to duration, and generate correct tokens

Text error correction

- Implicit error detection and correction
  - Target-source attention (NAR)
    - Determine the whole length of target sequence, but no need the exact alignment
    - Use QKV attention:
      - Query: blank token/ position embedding according to the predicted length
      - Key/Value: source hidden
  - CTC (connectionist temporal classification) $P(y|x) = \sum_{z \in \varphi(y)} P(z|x)$
    - e.g., I have a hat → I have a cat
    - Input of the CTC model: I I I have have have a a a hat hat hat (duplicate 3 times)
    - Output of the CTC model: I I Ø Ø have Ø a a Ø cat Ø Ø
    - CTC path merge: I have a cat

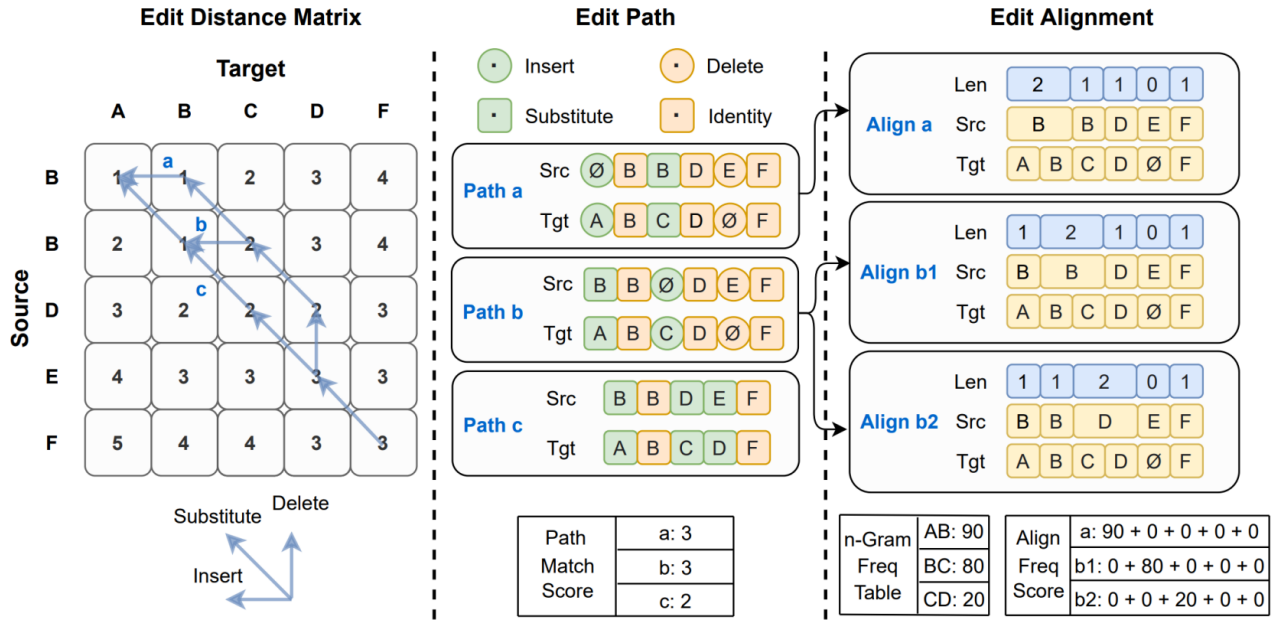
Text error correction

- Explicit error detection and correction
  - Exact insertion/deletion/substitution
    - Detect each source token as insertion/deletion/substitution
    - Still need to determine the length of insertion for parallel generation
  - Or directly use duration as an approximation: 0 for deletion, 1 for substitution or nochange, 2 or more for insertion

  - However, how to get the label for insertion/deletion/substitution or duration?

Text error correction

- How to get the label? Through target-source alignment!
  - Naïve hard match, not optimal
  - Alignment with dynamic programming, based on edit distance



Source: B B D E F

Target: A B C D F

Source: |B|,|B|,|B|,|D|,|E|,|F|

Target: |A|,|B|,|C|,|D|,| |,|F|

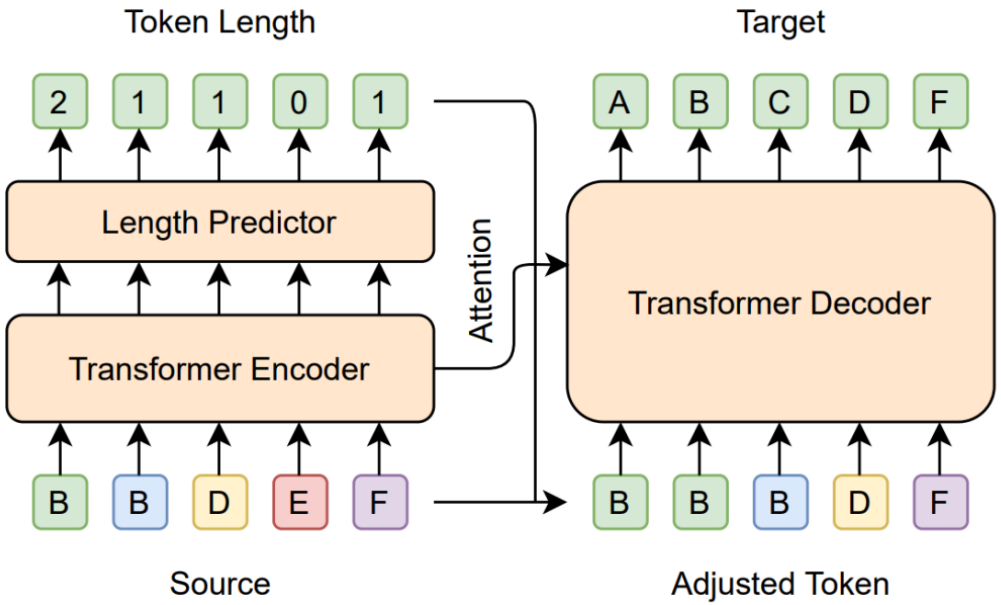
2 1 1 0 1

Leng Y, Tan X, Zhu L, et al. FastCorrect: Fast Error Correction with Edit Alignment for Automatic Speech Recognition[J]. NeurIPS 2021

Meta Microsoft

Text error correction

- How to use the alignment label?
  - Provide duration label or provide error detection label



Text error correction

- In training correction model, usually data augmentation/pre-training is used
  - Why? Few modifications in text error correction (e.g., 10% WER), few training signal can be leveraged
- How to augment data?
  - Manually augment: insertion/deletion/substitution
    - The probability distribution of deletion, insertion and substitution is set to the error distribution
  - Model based augment
    - BERT model with masked language modeling

Text error correction

- Text error correction on speech recognition (FastCorrect, NeurIPS 2021)
  - ASR model is a Conformer model from ESPnet, on Chinese AISHELL-1 dataset

AISHELL-1	Test Set		Dev Set		Latency (ms/sent) on Test Set		
	WER	WERR	WER	WERR	GPU	CPU*4	CPU
No correction	4.83	-	4.46	-	-	-	-
AR model	4.08	15.53	3.80	14.80	149.5 (1×)	248.9 (1×)	531.3 (1×)
LevT (MIter=1) [9]	4.73	2.07	4.37	2.02	54.0 (2.8×)	82.7 (3.0×)	158.1 (3.4×)
LevT (MIter=3) [9]	4.74	1.86	4.38	1.79	60.5 (2.5×)	83.9 (3.0×)	161.6 (3.3×)
FELIX [25]	4.63	4.14	4.26	4.48	23.8 (6.3×)	41.7 (6.0×)	85.7 (6.2×)
FastCorrect	**4.16**	**13.87**	**3.89**	**13.3**	**21.2** (7.1×)	**40.8** (6.1×)	**82.3** (6.5×)

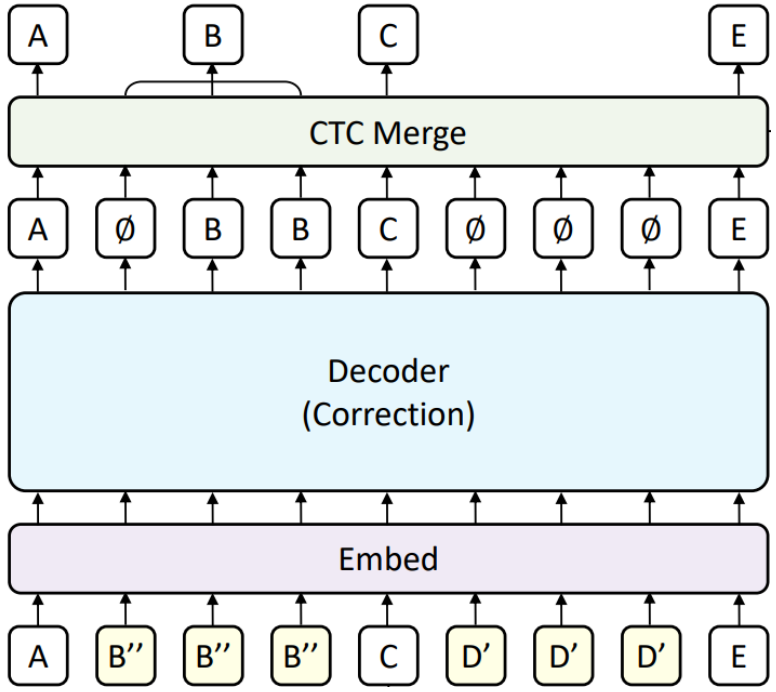
Text error correction

- Text error correction on speech recognition (FastCorrect, NeurIPS 2021)
  - Compare with deep encoder and shallow decoder (no inductive bias)

Model	AISHELL-1		
	WER	Latency (ms/sent)	
	%	GPU	CPU
No Correction	4.83	-	-
AR 6-6	4.08	149.5 (1×)	531.3 (1×)
AR 8-4	4.14	120.5 (1.2×)	427.6 (1.2×)
AR 10-2	4.23	84.0 (1.8×)	317.6 (1.5×)
AR 11-1	4.30	66.5 (2.2×)	281.0 (1.7×)
FastCorrect	4.16	**21.2** (7.1×)	**82.3** (6.5×)

Text error correction

- **Is implicit detection or explicit detection good enough?**
  - Implicit: does not provide clear signal about which tokens are incorrect
  - Explicit: suffers from detection accuracy (insertion/deletion/substitution)
  - A better way: a soft detection mechanism: neither too implicit nor too explicit



Text error correction

- Extensions of error correction on ASR/OCR or other text generation models
  - N-best output
    - Voting effect: tokens from multiple sentences can verify the correctness with each other
    - e.g., “I have a cat”, “I have a hat”, “I have a bat”
  - Source speech/image information
    - Two encoders, one for speech/image, the other for error text
    - Use cross-attention to serve as additional input
- Action based correction
  - First predict the correction action: keep, delete, generate
  - Then generate the corresponding correction

Modality	Task	Source	Target	Target-Source vs Target-Target	Difficulty of NAR
Text Generation	**Neural Machine Translation**	Source language	Target language	$\approx$?	****
	Text Summarization	Long text	Short Summarization	$>$	***
	Text Error Correction	Error Text	Correct Text	$>$	***
	Text Style Transfer	Source Text	Target text	$>$	***
	Dialogue Generation	Dialogue	Response	$<$	*****
	Speech Recognition	Speech	Text	$\geq$?	***
Speech Generation	Text to Speech	Text	Speech	$>$	***
	Singing Voice Synthesis	Score	Singing Voice	$>>$	***
	Voice Conversion	Source Voice	Target Voice	$\gg$	**
	Speech Enhancement	Noisy Speech	Clean Speech	$\gg$	*
Image Generation	Pixel Generation	Class ID	Image Pixel	-	*
	Discrete Token Generation		Image Token	-	**

The values in the last two columns are just for reference

Automatic Speech Recognition

- Input: speech (waveform/spectrogram) sequence
- Output: text (word/BPE/character/phoneme) sequence
- Target dependency comparison with NMT
  - NMT: For a source word, the translation can be A-B or C-D. If the first word is A, then the next should be B, otherwise if the first is C, the next should be D.
  - ASR: For a source segment, the recognition should be A-B. If the first word is A, then the next should be B. If the first is C, the next should be still B.

Target dependency in ASR is weaker than that in NMT

Automatic Speech Recognition

- CTC (connectionist temporal classification)

$$P(y|x) = \sum_{z \in \varphi(y)} P(z|x)$$

  $\varphi(y)$ is called CTC path
  - e.g., z: HHEØLØLOO, or ØHHEELØLO → y: HELLO
- CTC assumes no dependency among the target tokens, but can still works well
- But modeling target dependency can still bring improvement

Automatic Speech Recognition

- Refine the CTC output with bidirectional dependency
  - Token-level (after CTC merge):
    - Mask-CTC: CMLM decoder refine the CTC output
    - Improved Mask-CTC: with length adjustment
    - Insertion Transformer:
    - KERMIT
  - Frame-level (before CTC merge)
    - Imputer:
    - Align-Refine:
    - Align-Denoise
    - Intermediate CTC
    - Self-conditioned CTC: DSLSP

Higuchi Y, Chen N, Fujita Y, et al. A Comparative Study on Non-Autoregressive Modelings for Speech-to-Text Generation[J]. arXiv 2021.

Automatic Speech Recognition

Model	#iter	Processing unit	CTC
A-CMLM [17]	3	token	
Imputer [18]	8	frame	✓
LASO [20]	1	token	
Spike-Triggered [22]	1	token	✓
Mask-CTC [19]	10	token	✓
Improved Mask-CTC [31]	5	token	✓
Align-Refine [23]	5	frame	✓
Align-Denoise [33]	1	frame	✓
Insertion Transformer [21]	$\simeq \log_2(L)$	token	
KERMIT [21]	$\simeq \log_2(L)$	token	✓
Intermediate CTC [35]	1	frame	✓
Self-conditioned CTC [36]	1	frame	✓
CIF-NA [39]	1	token	✓

Higuchi Y, Chen N, Fujita Y, et al. A Comparative Study on Non-Autoregressive Modelings for Speech-to-Text Generation[J]. arXiv 2021.

Automatic Speech Recognition

Model		#iter	Inference speed		LS-100 (WER)				TED2 (WER)		CSJ-APS (CER)		
			RTF	Speedup	dev clean	dev other	test clean	test other	dev	test	eval1	eval2	eval3
AR	CTC/attention	L	0.341	1.00×	6.8	18.8	7.4	19.0	11.6	8.7	5.4	4.0	9.8
	+ beam-search	$> L$	3.419	0.10×	**6.3**	**18.2**	**6.8**	**18.5**	10.4	8.4	**5.1**	**3.8**	**9.0**
	Transducer	L	0.069	4.94×	7.3	19.9	7.6	19.9	9.6	9.2	6.3	4.5	10.6
	+ beam-search	$> L$	0.234	1.46×	6.4	18.8	**6.8**	18.9	**8.6**	**8.2**	5.2	4.1	10.0
NAR	CTC	1	0.059	5.78×	7.4	20.5	7.8	20.8	8.9	8.6	5.4	4.0	9.6
	Mask-CTC	10	0.063	5.41×	7.2	20.3	7.5	20.6	8.9	8.5	5.6	4.0	9.6
	Improved Mask-CTC	5	0.072	4.74×	7.0	19.8	7.3	20.2	8.8	8.3	5.5	4.0	9.5
	Align-Denoise	1	0.073	4.67×	8.0	22.3	8.4	22.5	9.0	8.7	5.4	**3.7**	**9.1**
	Intermediate CTC	1	0.059	5.78×	6.9	19.7	7.1	20.2	**8.5**	8.3	5.6	4.1	9.8
	Self-conditioned CTC	1	0.059	5.78×	**6.6**	**19.4**	**6.9**	**19.7**	8.7	**8.0**	**5.3**	**3.7**	**9.1**
	KERMIT	$\simeq \log_2(L)$	0.361	1.06×	7.1	19.7	7.4	20.2	9.1	8.2	5.4	**3.7**	9.5
	Insertion Transformer	$\simeq \log_2(L)$	0.083	4.11×	16.0	27.3	16.2	27.4	–	–	–	–	–
	CIF-NA†	1	0.073	4.67×	15.4	34.0	15.7	34.6	–	–	–	–	–

Higuchi Y, Chen N, Fujita Y, et al. A Comparative Study on Non-Autoregressive Modelings for Speech-to-Text Generation[J]. arXiv 2021.

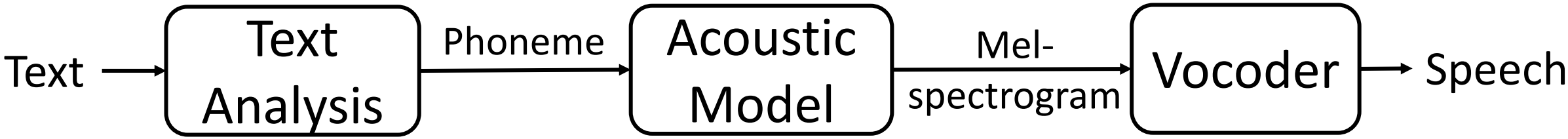
Meta Microsoft

Modality	Task	Source	Target	Target-Source vs Target-Target	Difficulty of NAR
Text Generation	**Neural Machine Translation**	Source language	Target language	$\approx$?	****
	Text Summarization	Long text	Short Summarization	$>$	***
	Text Error Correction	Error Text	Correct Text	$>$	***
	Text Style Transfer	Source Text	Target text	$>$	***
	Dialogue Generation	Dialogue	Response	$<$	*****
	Speech Recognition	Speech	Text	$\geq$?	***
Speech Generation	**Text to Speech**	Text	Speech	$>$	***
	Singing Voice Synthesis	Score	Singing Voice	$>>$	***
	Voice Conversion	Source Voice	Target Voice	$\gg$	**
	Speech Enhancement	Noisy Speech	Clean Speech	$\gg$	*
Image Generation	Pixel Generation	Class ID	Image Pixel	-	*
	Discrete Token Generation		Image Token	-	**

The values in the last two columns are just for reference

Text to Speech

- Input: text (character/phoneme) sequence
- Output: speech (waveform/spectrogram) sequence



- We will mainly focus on acoustic model: phoneme to mel-spectrogram mapping, and vocoder: mel-spectrogram to waveform mapping

Text to Speech

- Target dependency comparison with NMT and ASR
  - NMT: For a source word, the translation can be A-B or C-D. If the first word is A, then the next should be B, otherwise if the first is C, the next should be D.
  - ASR: For a source segment, the recognition should be A-B. If the first word is A, then the next should be B. If the first is C, the next should be still B.
  - Besides, NMT and ASR rely on target dependency for language modeling

  - TTS
    - Speech frames largely depend on the source word, waveform samples largely depend on the condition spectrogram
    - Dependency among speech frames and waveform samples? Yes, indeed, but…

Text to Speech

- Target dependency comparison with NMT and ASR
  - Discrete tokens in languages is contextualized, explained mutually. Language tokens have strong mutual dependency
  - But for continuous signal like speech/sound/image, they depends on the concept, like speech frames depend on a word, image pixel depend on a class
  - Maybe weak mutual dependency among signal itself, that is why parallel generation model is so popular in image or speech generation.

  - Another point
    - Waveform samples and image pixels are so long in sequence, inference speed is extremely slow for autoregressive generation. Strong demand for NAR generation!
  - Some specific reasons in speech/spectrogram generation:
    - Usually, speech frame is obtained via STFT with window/hop size (50ms/12.5ms), two adjacent frames have 3/4 overlapping. Autoregressive generation will lead to copy, unstable to model.

Text to Speech

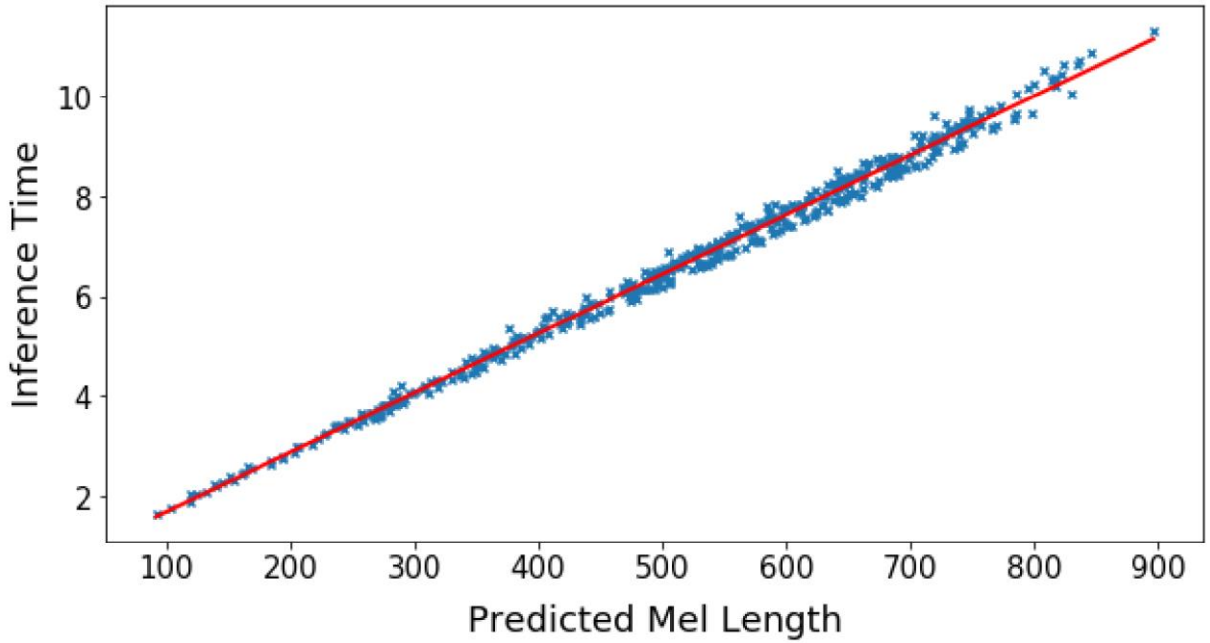
- Target dependency comparison with NMT and ASR
  - NMT: For a source word, the translation can be A-B or C-D. If the first word is A, then the next should be B, otherwise if the first is C, the next should be D.
  - ASR: For a source segment, the recognition should be A-B. If the first word is A, then the next should be B. If the first is C, the next should be still B.
  - Besides, NMT and ASR rely on target dependency for language modeling

  - TTS
    - Speech frames largely depend on the source word, waveform samples largely depend on the condition spectrogram
    - Dependency among speech frames and waveform samples? Yes, indeed, but…

Based on above analysis, TTS has much weaker target dependency than NMT, and slightly weaker than ASR

Text to Speech: NAR for speedup

- Compared with autoregressive mel-spectrogram/waveform generation
  - Sequence is very long, e.g., 1s speech, 500 mel, 24000 waveform points
  - Slow inference speed



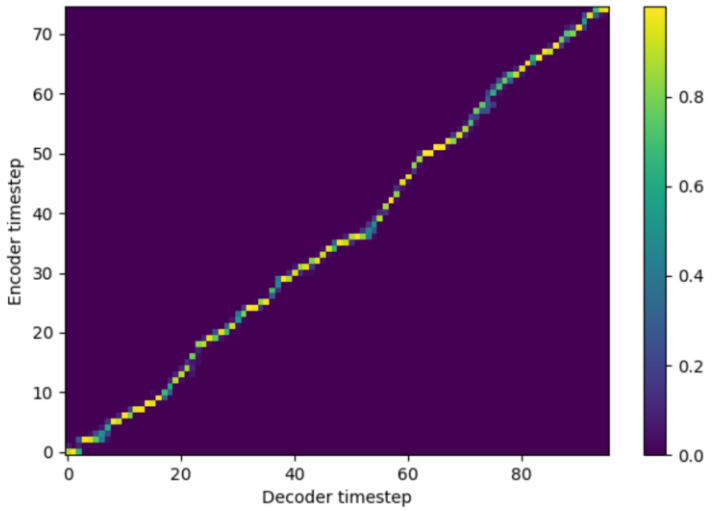
Meta Microsoft

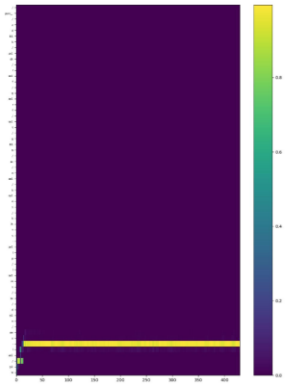
Text to Speech: NAR for robustness

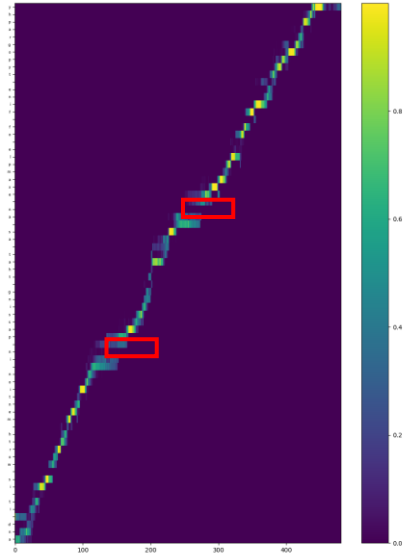
- AR model (Tacotron 2, DeepVoice 3, Transformer TTS) not robust: words skipping and repeating;

  You can call me directly at 4257037344 or my cell 4254447474 or send me a meeting request with all the appropriate information.

- Encoder-decoder attention: Attention between mel-spectrogram and phoneme: monotonic and diagonal







And it is worth mention in passing that,
as an example of fine typography

Text to Speech: NAR for controllability

- AR model automatically determines the length of speech
  - Lack of controllability: hard to control the voice speed/prosody in AR generation

  *was executed on a gibbet in front of his victim's house.*

  *after dinner | he went into hiding for a day or two*

Text to Speech: NAR overview

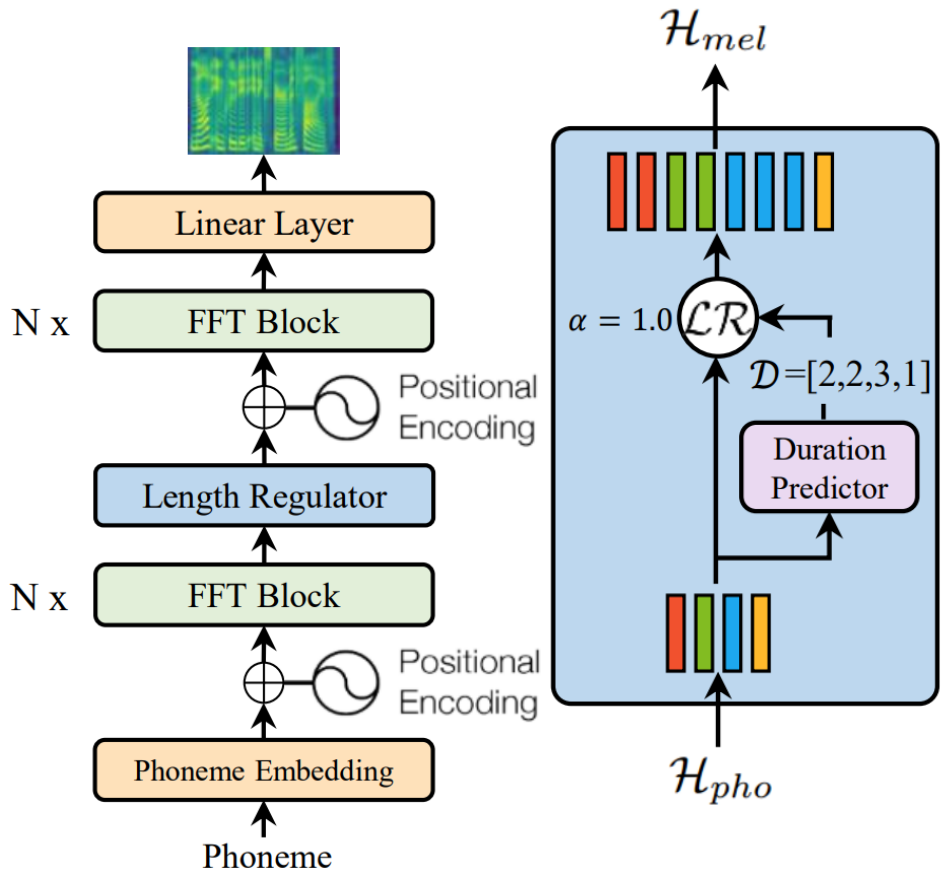
- Overview of NAR models in TTS
  - N is sequence length, T is iteration step

Modeling Paradigm	TTS Model	Training	Inference
AR (RNN)	Tacotron 1/2, SampleRNN, LPCNet	$\mathcal{O}(N)$	$\mathcal{O}(N)$
AR (CNN/Self-Att)	DeepVoice 3, TransformerTTS, WaveNet	$\mathcal{O}(1)$	$\mathcal{O}(N)$
NAR (CNN/Self-Att)	FastSpeech 1/2, ParaNet	$\mathcal{O}(1)$	$\mathcal{O}(1)$
NAR (GAN/VAE)	MelGAN, HiFi-GAN, FastSpeech 2s, EATS	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Flow (AR)	Par. WaveNet, ClariNet, Flowtron	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Flow (Bipartite)	WaveGlow, FloWaveNet, Glow-TTS	$\mathcal{O}(T)$	$\mathcal{O}(T)$
Diffusion	DiffWave, WaveGrad, Grad-TTS, PriorGrad	$\mathcal{O}(T)$	$\mathcal{O}(T)$

Text to Speech: FastSpeech

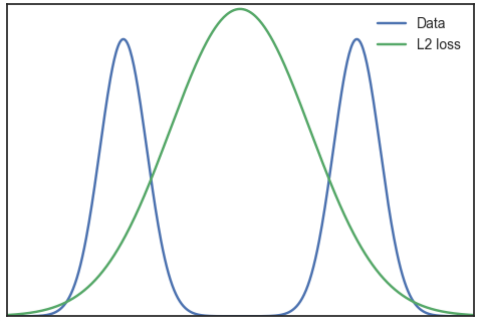
- Design of FastSpeech
  - Generate mel-spectrogram in parallel
    (for speedup, 270x)
  - Remove the text-speech attention mechanism
    (for robustness, no word skipping/repeating)
  - Feed-forward transformer with length regulator
    (for controllability, speed control)

https://speechresearch.github.io/fastspeech/

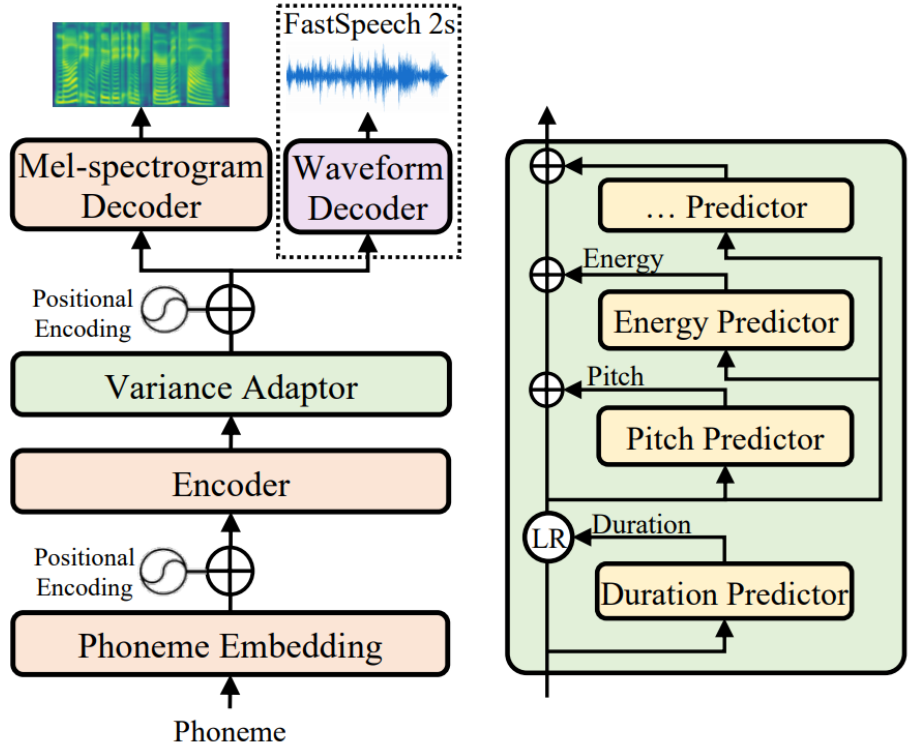


Text to Speech: FastSpeech 2

- FastSpeech 2
  - Improve FastSpeech (one-to-many mapping)
  - Use variance adaptor to predict duration, pitch, energy, etc
  - Simplify training pipeline of FastSpeech (KD)
  - FastSpeech 2s: a fully end-to-end parallel text to wave model



p(x|y) multimodal distribution

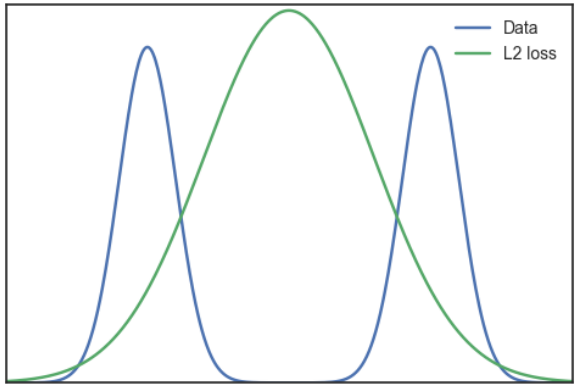


(a) FastSpeech 2 (b) Variance adaptor

https://speechresearch.github.io/fastspeech2/

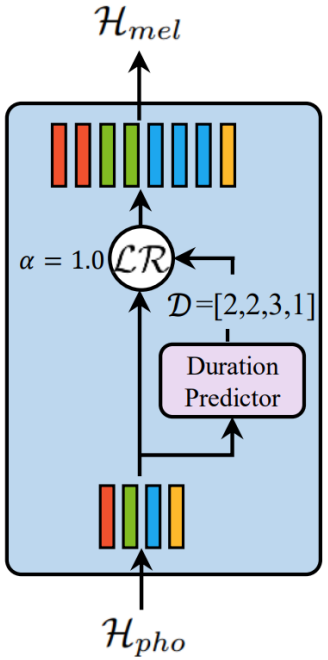
Text to Speech: Multi-modal

- **How to model multi-modal distribution p(x|y)**
  - Simplify the multimodal distribution p(x|y)
    - Simplify target: Data distillation: lossy, Data transformation: Short Time Fourier Transformation (STFT), DCT, Wavelet
    - More input information: Pitch, duration, energy, speaker ID, prosody tag, etc..
    - **Better alignment: duration/alignment modeling**
  - Advanced modeling for multimodal distribution
    - L1: Laplace distribution, L2: Gaussian distribution
    - Mixture of Gaussian/Laplace/Logistic: multimodal distribution
    - High-order statistics loss: high-order moment, SSIM
    - Model-based loss (any distribution): classifier, discriminator in GAN
    - **Advanced generative models (~~AR~~/Flow/VAE/Diffusion/GAN, etc)**

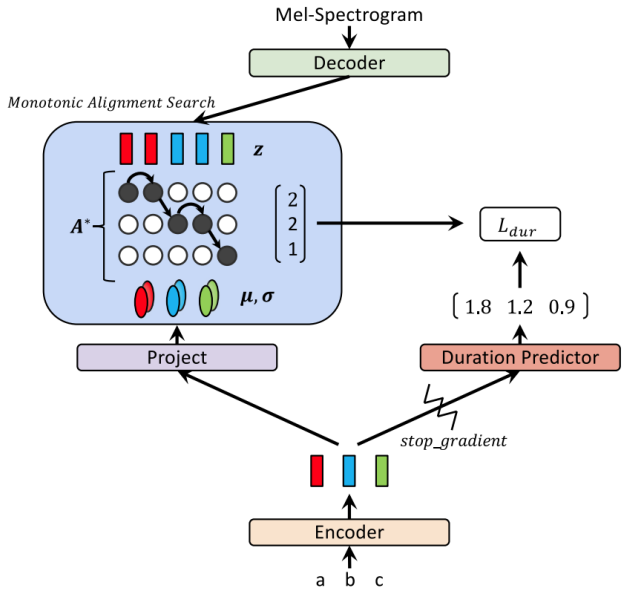


Text to Speech: Better alignment

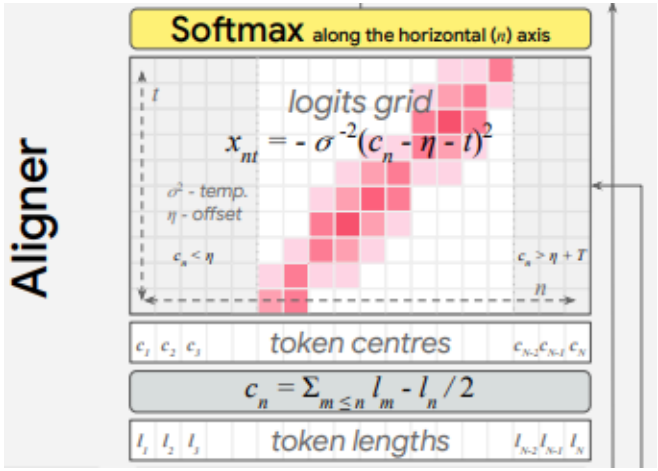
- Duration modeling
  - Statistic parametric speech synthesis→ AR model → NAR
  - Duration → attention, no duration → duration prediction (**technique renaissance!)**



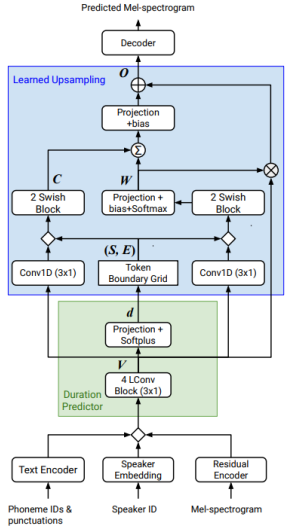
FastSpeech



Glow-TTS



EATS



Parallel Tacotron 2

Meta Microsoft

Text to Speech: Advanced generative models

- Flow/VAE/Diffusion/GAN
  - A comparison among different generative models for TTS
    - Simplicity in math formulation and optimization
    - Support parallel generation
    - Support latent manipulation
    - Support likelihood estimation

Generative Model	AR	VAE	Flow/AR	Flow/Bipartite	Diffusion	GAN
Simple	Y	N	N	N	N	N
Parallel	N	Y	Y	Y	Y	Y
Latent Manipulate	N	Y	Y	Y	Y	Y*
Likelihood Estimate	Y	Y	Y	Y	Y	N

GAN is weak in latent manipulation, since the condition in TTS is so strong, P(y|x) is not that much multi-modal compared to image synthesis, and some GAN based model do not add random noise

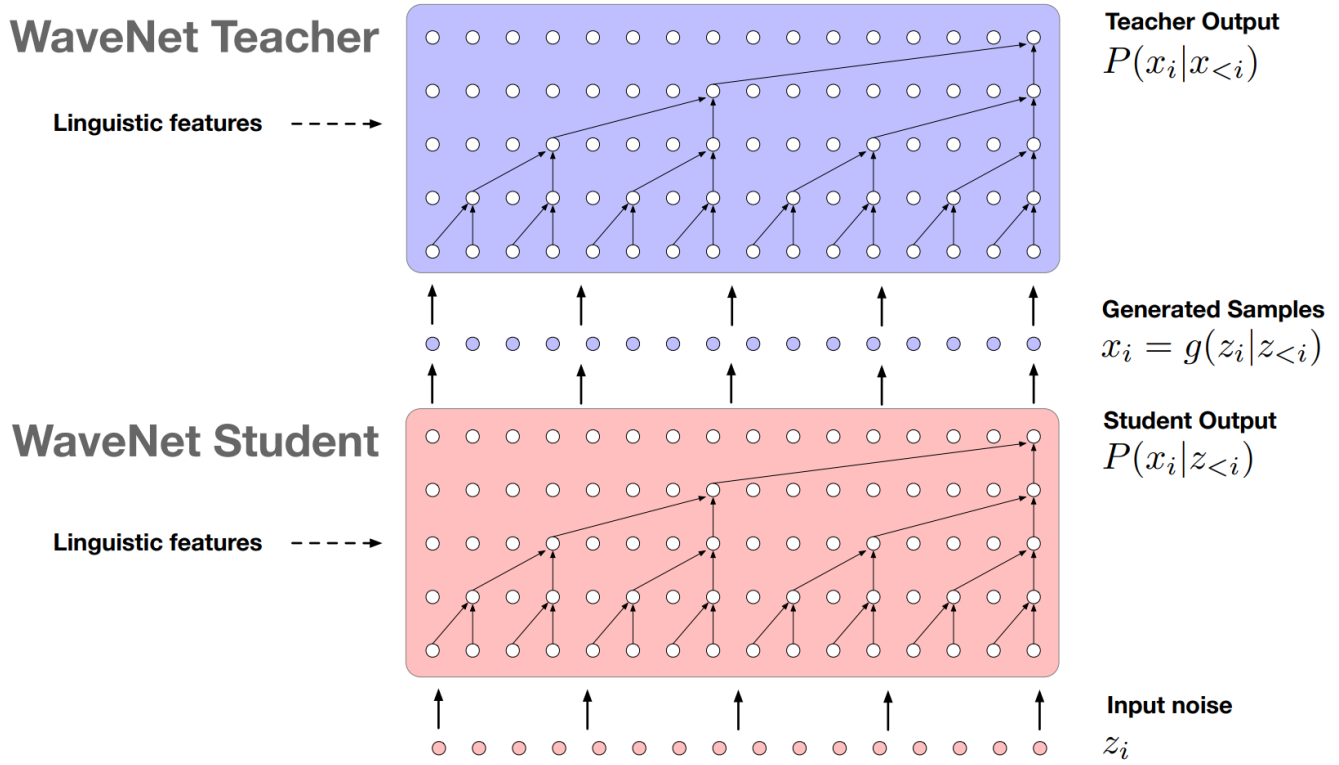
Text to Speech: Flow

- Map between data distribution p(x) and standard (normalizing) prior distribution p(z) Evaluation $z = f^{-1}(x)$ Synthesis $x = f(z)$
- Category of normalizing flow
  - AR (autoregressive): AF (autoregressive flow) and IAF (inverse autoregressive flow)
  - Bipartite: RealNVP and Glow

Flow		Evaluation $z = f^{-1}(x)$	Synthesis $x = f(z)$
AR	AF [261]	$z_t = x_t \cdot \sigma_t(x_{<t};\theta) + \mu_t(x_{<t};\theta)$	$x_t = \frac{z_t - u_t(x_{<t};\theta)}{\sigma_t(x_{<t};\theta)}$
	IAF [169]	$z_t = \frac{x_t - \mu_t(z_{<t};\theta)}{\sigma_t(z_{<t};\theta)}$	$x_t = z_t \cdot \sigma_t(z_{<t};\theta) + \mu_t(z_{<t};\theta)$
Bipartite	RealNVP [66]	$z_a = x_a,$	$x_a = z_a,$
	Glow [167]	$z_b = x_b \cdot \sigma_b(x_a;\theta) + \mu_b(x_a;\theta)$	$x_b = \frac{z_b - \mu_b(x_a;\theta)}{\sigma_b(x_a;\theta)}$

Text to Speech: Flow (vocoder)

- Parallel WaveNet (AR)
  - Knowledge distillation: Student (IAF), Teacher (AF)
  - Combine the best of both worlds
    - Parallel inference of IAF student
    - Parallel training of AF teacher
  - Other works
    - ClariNet



Text to Speech: Flow (vocoder)

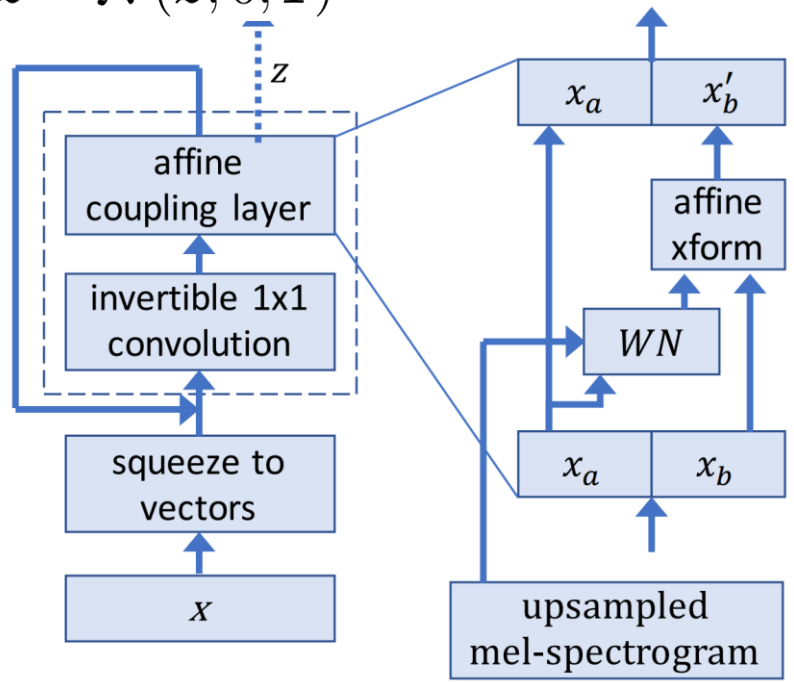
- WaveGlow (Bipartite)
  - Flow based transformation

$$z = f_k^{-1} \circ f_{k-1}^{-1} \circ \ldots f_0^{-1}(x) \qquad x = f_0 \circ f_1 \circ \ldots f_k(z) \qquad z \sim \mathcal{N}(z; 0, I)$$

  - Affine Coupling Layer

$$x_a, x_b = split(x)$$
$$(\log s, t) = WN(x_a, mel\text{-}spectrogram)$$
$$x_b\prime = s \odot x_b + t$$
$$f_{coupling}^{-1}(x) = concat(x_a, x_b\prime)$$

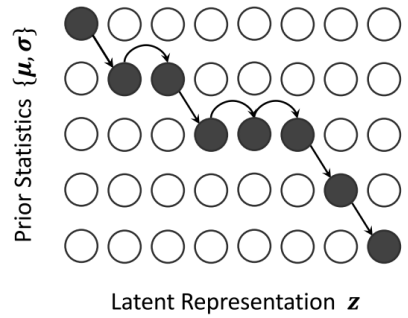
×12



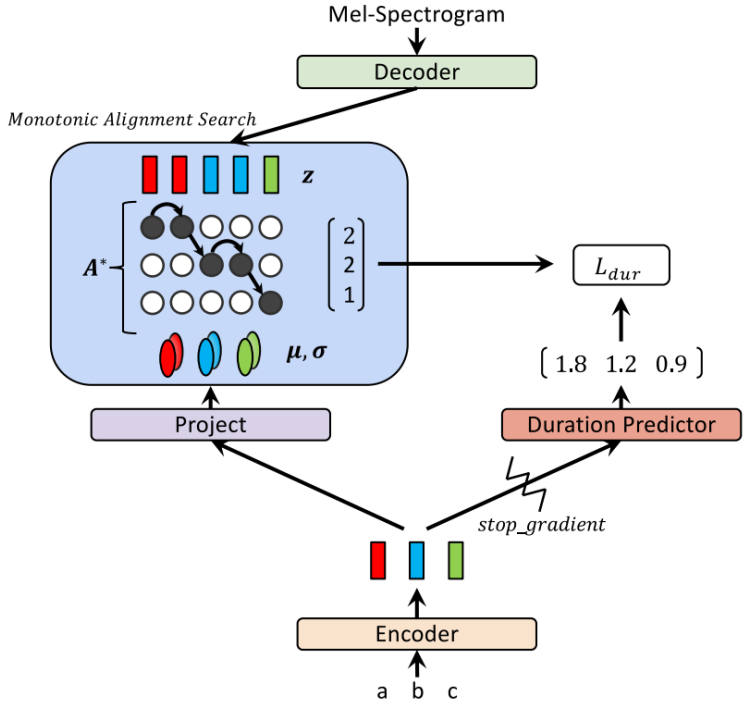
  - Other works
    - FloWaveNet
    - WaveFlow

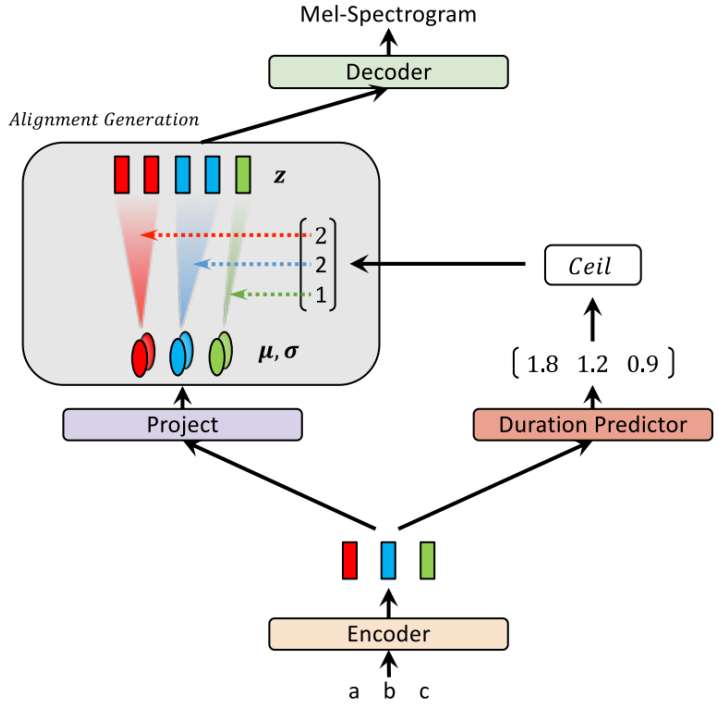
Text to Speech: Flow (acoustic model)

- Glow-TTS
  - Log likelihood $\log P_X(x|c) = \log P_Z(z|c) + \log \left| \det \frac{\partial f_{dec}^{-1}(x)}{\partial x} \right|$
  - Prior is learnt from phoneme text $\log P_Z(z|c;\theta,A) = \sum_{j=1}^{T_{mel}} \log \mathcal{N}(z_j; \mu_{A(j)}, \sigma_{A(j)})$
  - Alignment A is obtained by monotonic alignment search



- Other works
  - FlowTTS, Flowtron
  - EfficientTTS





Meta Microsoft

Text to Speech: GAN

- Adversarial loss

$$\mathcal{L}_{Adv}(D;G) = \mathbb{E}_{(x,s)}\left[(D(x)-1)^2 + (D(G(s)))^2\right]$$

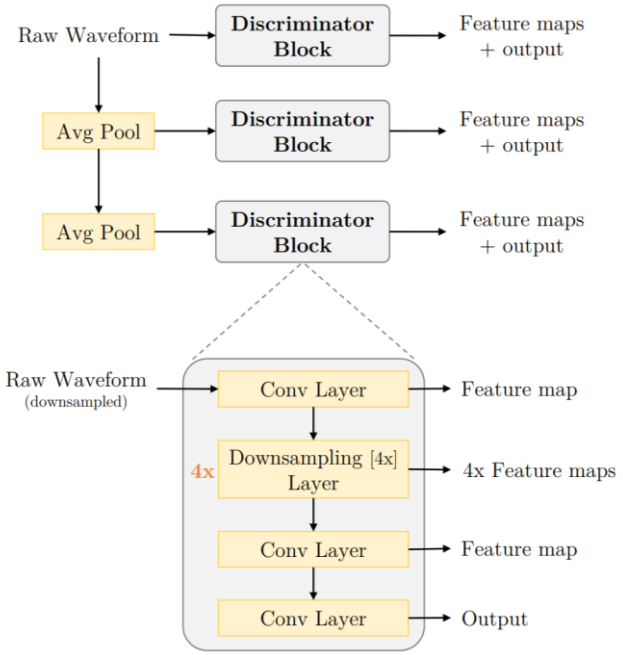
$$\mathcal{L}_{Adv}(G;D) = \mathbb{E}_{s}\left[(D(G(s))-1)^2\right]$$

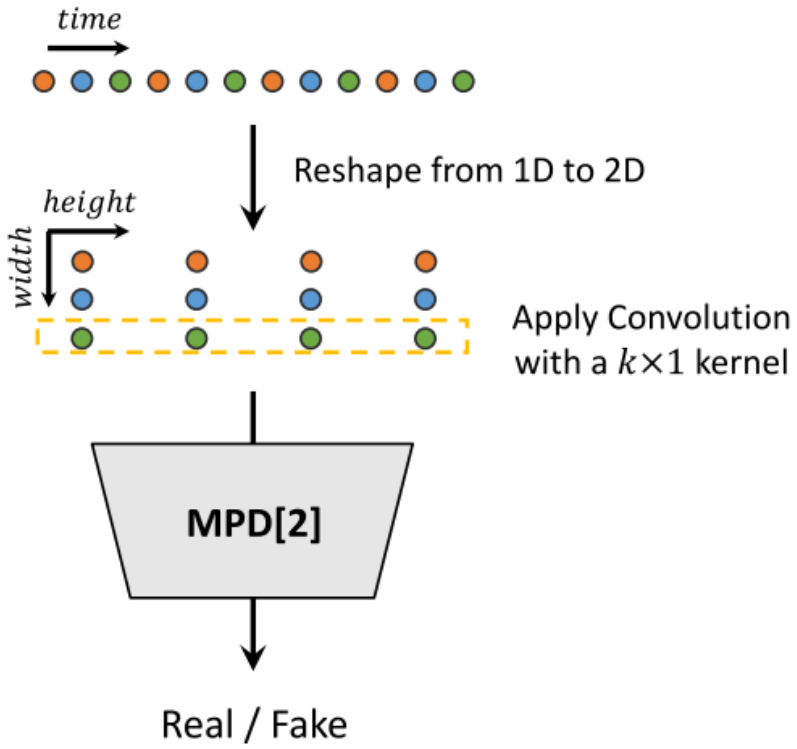
- Category of GAN based vocoders

GAN	Generator	Discriminator	Loss
WaveGAN [68]	DCGAN [287]	/	WGAN-GP [97]
GAN-TTS [23]	/	Random Window D	Hinge-Loss GAN [198]
MelGAN [178]	/	Multi-Scale D	LS-GAN [231] Feature Matching Loss [182]
Par.WaveGAN [402]	WaveNet [254]	/	LS-GAN, Multi-STFT Loss
HiFi-GAN [174]	Multi-Receptive Field Fusion	Multi-Period D, Multi-Scale D	LS-GAN, STFT Loss, Feature Matching Loss
VocGAN [408]	Multi-Scale G	Hierarchical D	LS-GAN, Multi-STFT Loss, Feature Matching Loss
GED [96]	/	Random Window D	Hinge-Loss GAN, Repulsive loss

Text to Speech: GAN (vocoder)

- HiFiGAN
  - Multi-Scale Discriminator (MSD)
  - Multi-Period Discriminator (MPD)





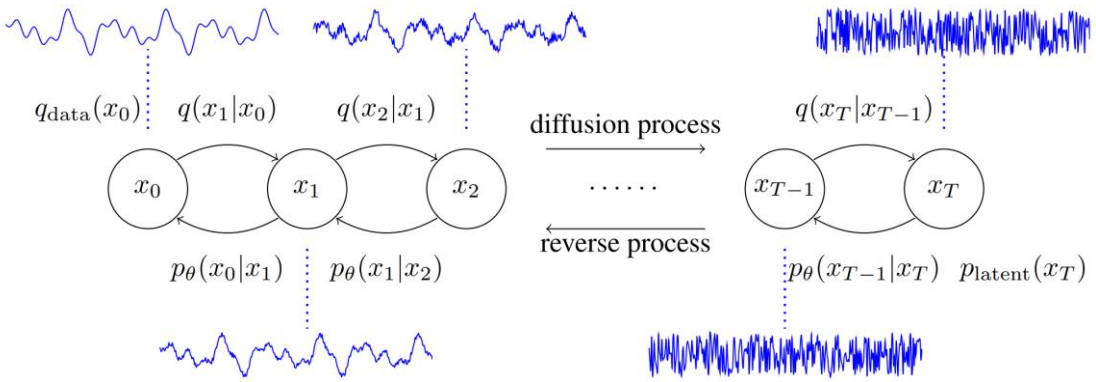
Meta Microsoft

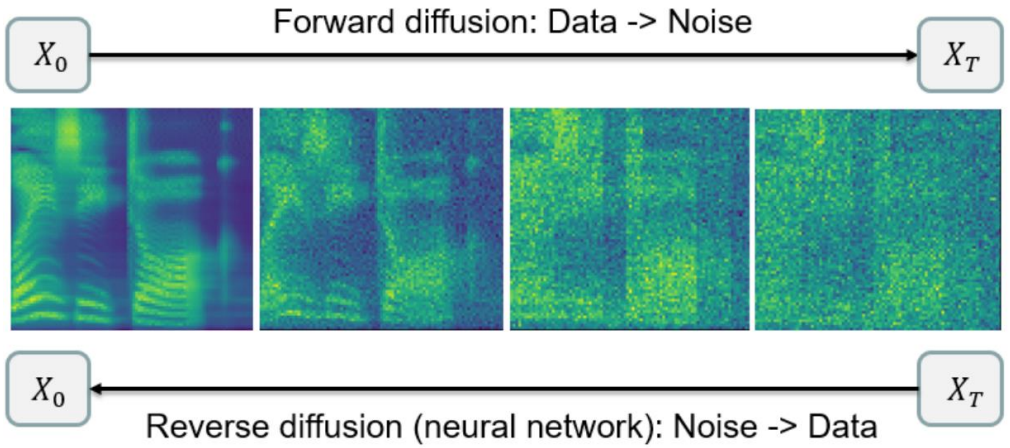
Text to Speech: Diffusion

- Diffusion probabilistic model
  - Forward (diffusion) process:
  - Reverse (denoising) process

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \quad q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)\right)$$





Text to Speech: Diffusion

- Loss derived from ELBO: $L_{\text{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\mathbf{x}_t, t\right)\right\|^2\right]$
- Training and inference process

Algorithm 1 Training

for $i = 1, 2, \cdots, N_{\text{iter}}$ **do**
  Sample $x_0 \sim q_{\text{data}}$, $\epsilon \sim \mathcal{N}(0, I)$, and $t \sim \text{Uniform}(\{1, \cdots, T\})$
  Take gradient step on $\nabla_\theta \|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon,\ t)\|_2^2$ according to Eq. (7)
end for

Algorithm 2 Sampling

Sample $x_T \sim p_{\text{latent}} = \mathcal{N}(0, I)$
for $t = T, T-1, \cdots, 1$ **do**
  Compute $\mu_\theta(x_t, t)$ and $\sigma_\theta(x_t, t)$ using Eq. (5)
  Sample $x_{t-1} \sim p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_\theta(x_t, t)^2 I)$
end for
return x_0

Text to Speech: Diffusion

- Diffusion model for vocoder: DiffWave, WaveGrad
- Diffusion model for acoustic model: Diff-TTS, Grad-TTS
- Improving diffusion model for TTS
  - PriorGrad, SpecGrad, DiffGAN-TTS, WaveGrad 2, etc

- With sufficient diffusion steps, the quality is good enough, but latency is high
- How to reduce inference cost while maintaining the quality is challenging, and has a long way to go

Text to Speech: NAR with human-level quality

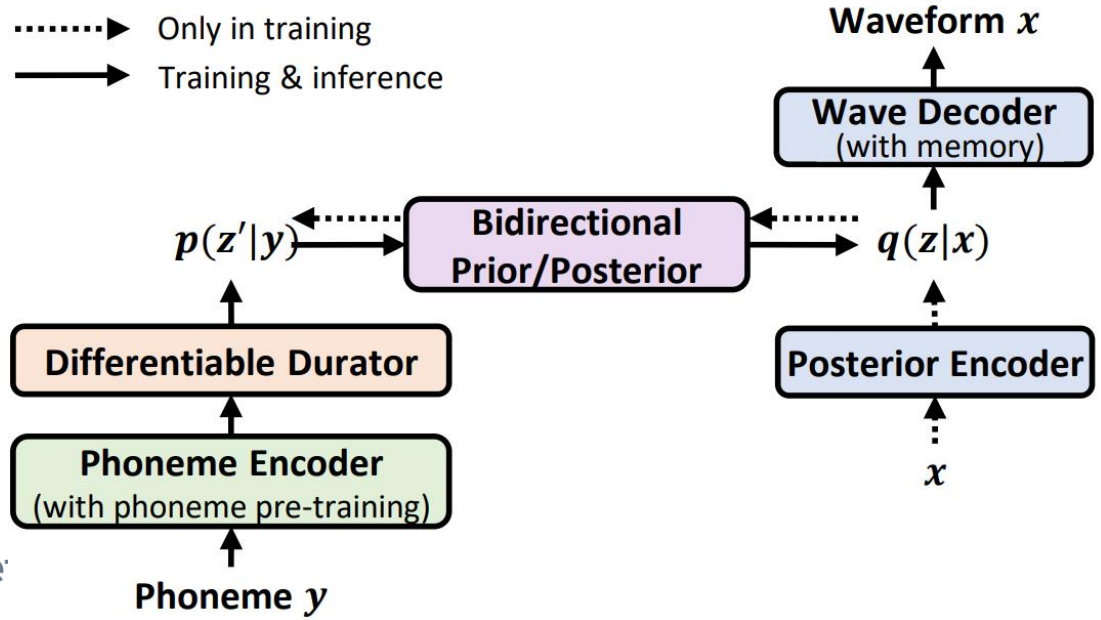
- NaturalSpeech: achieving human-level quality on LJSpeech dataset (CMOS)
- Leverage VAE to compress high-dimensional waveform x into frame-level representations $z \sim q(z|x)$, and is used to reconstruct waveform $x \sim p(x|z)$
- To enable text to waveform synthesis, z is predicted from y, $z \sim p(z|y)$

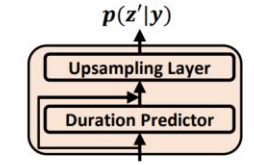
- However, the posterior $z \sim q(z|x)$ is more complicated than the prior $z \sim p(z|y)$.

Tan X, Chen J, Liu H, et al. NaturalSpeech: End-to-End Text to Speech Synthesis with Human-Level Quality[J]. arXiv 2022

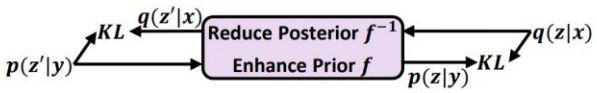
Text to Speech: NAR with human-level quality

- Solutions in NaturalSpeech
  - Phoneme encoder with large-scale phoneme pre-training
  - Differentiable durator
  - Bidirectional prior/posterior
  - Memory based VAE

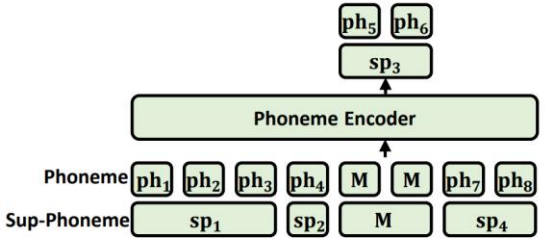




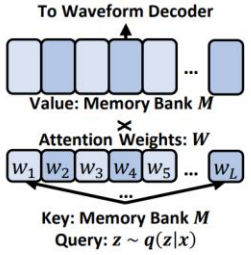
(a) Differentiable durator.



(b) Bidirectional prior/posterior.



(c) Phoneme pre-training.



(d) Memory mechanism in VAE.

Text to Speech: NAR with human-level quality

- Evaluations of NaturalSpeech
  - MOS and CMOS on par with recordings, p-value >> 0.05

Human Recordings	NaturalSpeech	Wilcoxon p-value
4.58 ± 0.13	4.56 ± 0.13	0.7145

Human Recordings	NaturalSpeech	Wilcoxon p-value
0	-0.01	0.6902

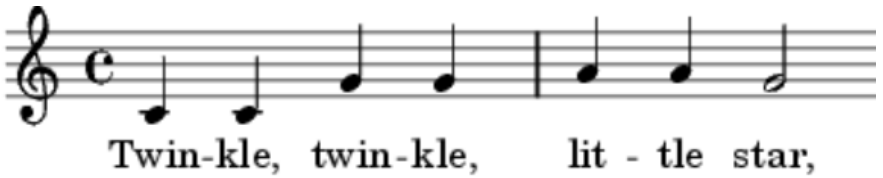

Achieving human-level quality on LJSpeech dataset for the first time!

Modality	Task	Source	Target	Target-Source vs Target-Target	Difficulty of NAR
Text Generation	**Neural Machine Translation**	Source language	Target language	$\approx$?	****
	Text Summarization	Long text	Short Summarization	$>$	***
	Text Error Correction	Error Text	Correct Text	$>$	***
	Text Style Transfer	Source Text	Target text	$>$	***
	Dialogue Generation	Dialogue	Response	$<$	*****
	Speech Recognition	Speech	Text	$\geq$?	***
Speech Generation	**Text to Speech**	Text	Speech	$>$	***
	Singing Voice Synthesis	Score	Singing Voice	$>>$	***
	Voice Conversion	Source Voice	Target Voice	$\gg$	**
	Speech Enhancement	Noisy Speech	Clean Speech	$\gg$	*
Image Generation	Pixel Generation	Class ID	Image Pixel	-	*
	Discrete Token Generation		Image Token	-	**

The values in the last two columns are just for reference

Singing voice synthesis

- Input: lyric/score (phoneme/pitch/duration); output: singing voice



- Target-source dependency is even stronger than target-target dependency when compared with text to speech synthesis
  - Duration and pitch in score can decide the duration/pitch in singing voice in a large extent
  - NAR is preferred
- Generative models are similar to that used in speech synthesis
- Only slight difference in alignment modeling
  - Given a rough duration and pitch in music score, predict more accurate duration and pitch in the singing voice
  - Since human cannot always sing according to music score, both in duration and pitch

Singing voice synthesis

- But singing has its distinctive characteristics other than speaking voice
  - Pitch/duration range is wider than speaking
  - Data space is larger (#phoneme * #pitch * #duration) than speech
  - Many singing techniques: trill, glide, opera singing, etc
  - High expressiveness with high fidelity (e.g., 48kHz)
- e.g., HiFiSinger, a NAR singing model for high-fidelity 48kHz voice


Method	MOS
Recording (48kHz)	4.03 ± 0.06
Recording (24kHz)	3.70 ± 0.08
XiaoiceSing (48kHz)	2.93 ± 0.06
Baseline (24kHz)	3.32 ± 0.09
Baseline (48kHz)	3.44 ± 0.08
HiFiSinger (24kHz)	3.47 ± 0.06
HiFiSinger (48kHz)	$\mathbf{3.76 \pm 0.06}$

Modality	Task	Source	Target	Target-Source vs Target-Target	Difficulty of NAR
Text Generation	**Neural Machine Translation**	Source language	Target language	$\approx$?	****
	Text Summarization	Long text	Short Summarization	$>$	***
	Text Error Correction	Error Text	Correct Text	$>$	***
	Text Style Transfer	Source Text	Target text	$>$	***
	Dialogue Generation	Dialogue	Response	$<$	*****
	Speech Recognition	Speech	Text	$\geq$?	***
Speech Generation	**Text to Speech**	Text	Speech	$>$	***
	Singing Voice Synthesis	Score	Singing Voice	$>>$	***
	Voice Conversion	Source Voice	Target Voice	$\gg$	**
	Speech Enhancement	Noisy Speech	Clean Speech	$\gg$	*
Image Generation	Pixel Generation	Class ID	Image Pixel	-	*
	Discrete Token Generation		Image Token	-	**

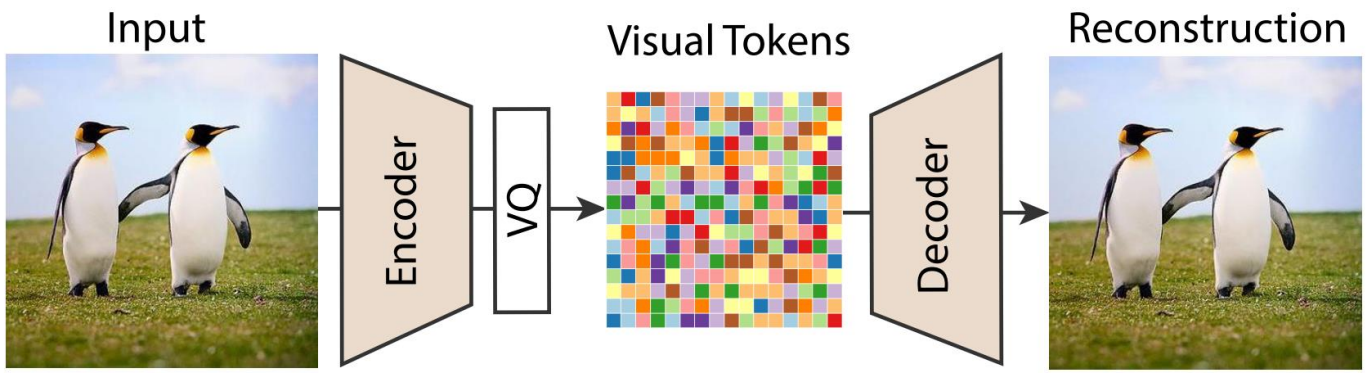
The values in the last two columns are just for reference

Image Generation

- Traditional image generation is inherently non-autoregressive based on generative models, like GAN, VAE, Flow and Diffusion model
  - GAN suffers from training instability and mode collapse
  - VAE suffers from blurriness
  - Flow and diffusion model require multiple iterative steps
  - Importantly, high-resolution image generation is costly for these methods

Image Generation

- A recent trend on image generation (VQ-VAE/VQ-GAN/DALL-E)
  - Step 1: use VQ-VAE 1/2 or VQ-GAN to quantize the high-resolution image into discrete tokens with encoder
  - Step 2: use autoregressive model such as Transformer/GPT to generate these discrete tokens autoregressively
  - Step 3: Use decoder in VQ-VAE/VQ-GAN to generate high-resolution image from these discrete tokens



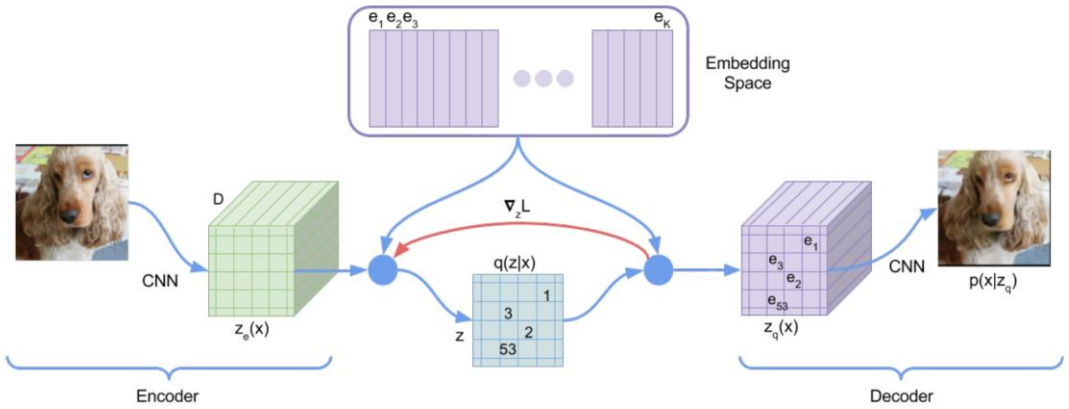


Image Generation

- A recent trend on image generation (VQ-VAE/VQ-GAN/DALL-E)
  - Step 1: use VQ-VAE 1/2 or VQ-GAN to quantize the high-resolution image into discrete tokens with encoder
  - Step 2: use autoregressive model such as Transformer/GPT to generate these discrete tokens autoregressively
  - Step 3: Use decoder in VQ-VAE/VQ-GAN to generate high-resolution image from these discrete tokens

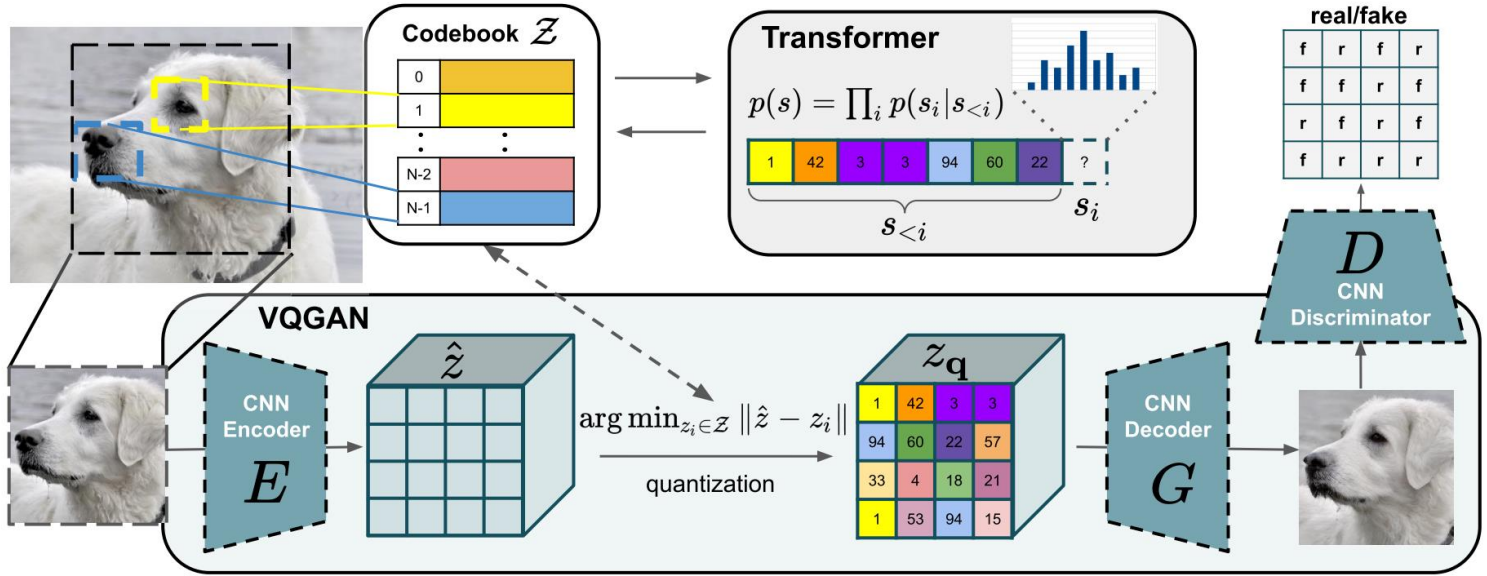


Image Generation

- Advantages and disadvantages
  - Advantages: first learn composition then render the details, very reasonable
  - Disadvantages
    - AR generation for discrete tokens, should have no order bias like in language
    - Slow in long sequence

Use NAR for discrete token generation!

Image Generation

- Input: any conditional information, e.g., class tag
- Output: discrete tokens of image
- Target dependency comparison with NLP/Speech tasks
  - Language has more target dependency, since contextual symbol
  - Speech has correspondence with source text, target dependency is weak
  - Image tokens seem to be similar to language, but more similar to speech

NAR image token generation weaker target dependency than NLP, maybe stronger than text to speech

Image Generation——MaskGIT

- Use BERT-like mask-predict mechanism to iterative predict discrete tokens

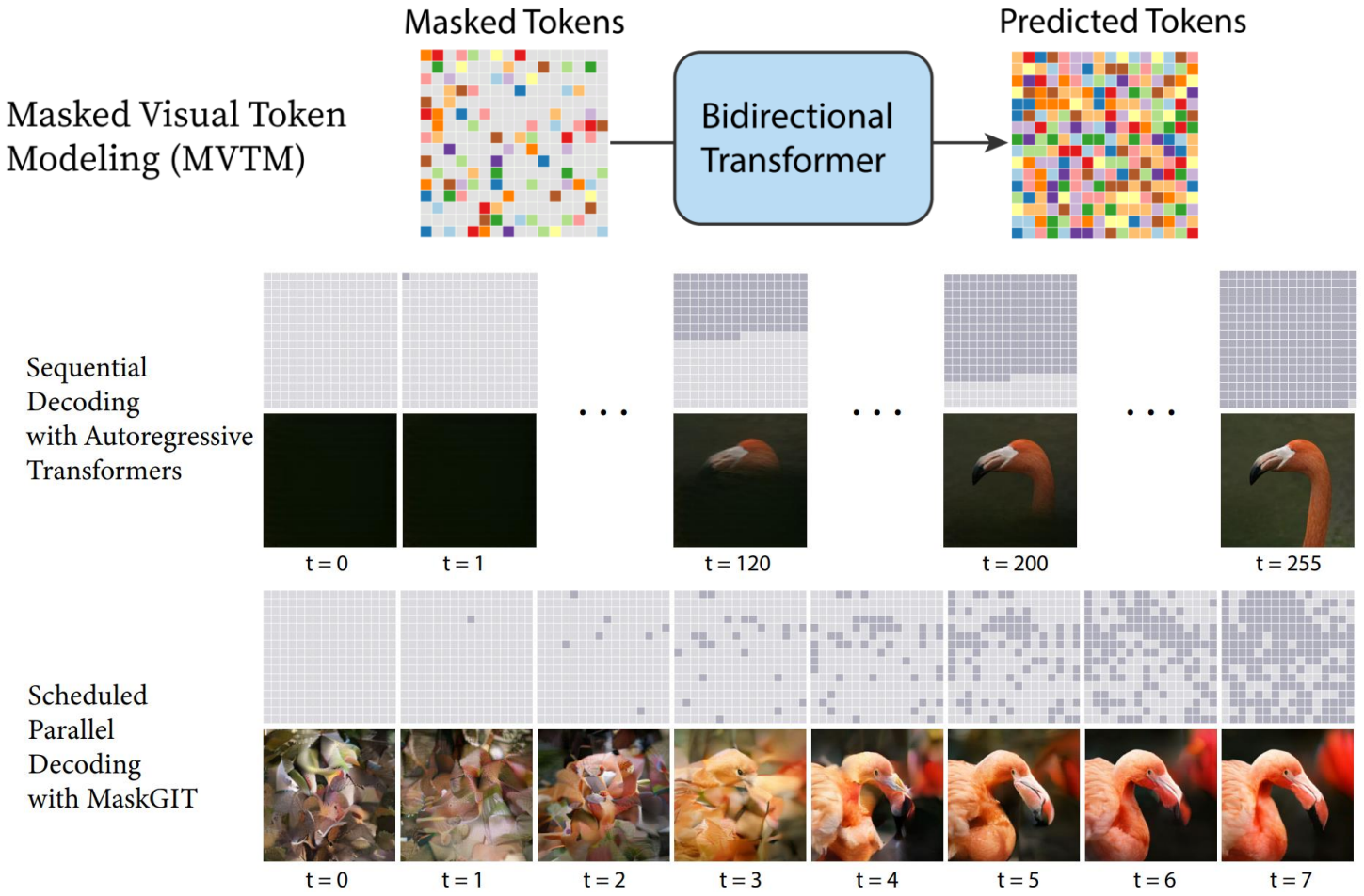
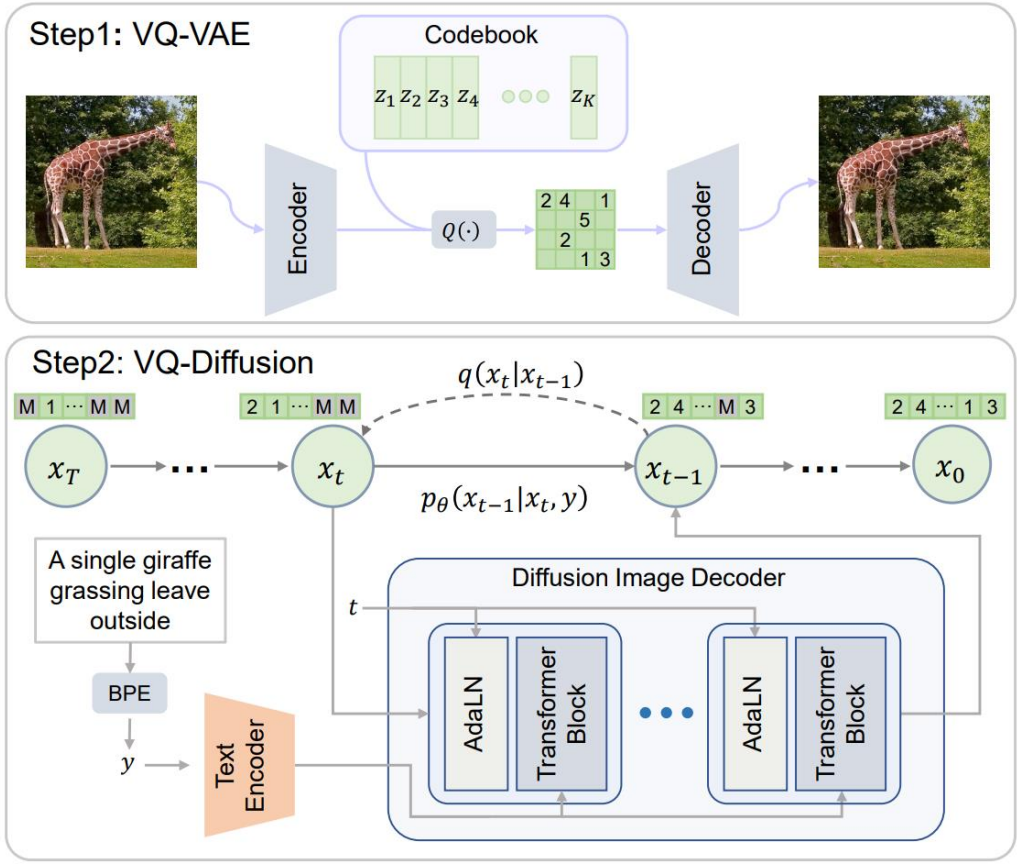


Image Generation——VQ-Diffusion

- Use diffusion model with a mask-and-replace diffusion strategy to model the discrete tokens in parallel



Modality	Task	Source	Target	Target-Source vs Target-Target	Difficulty of NAR
Text Generation	**Neural Machine Translation**	Source language	Target language	$\approx$?	****
	Text Summarization	Long text	Short Summarization	$>$	***
	Text Error Correction	Error Text	Correct Text	$>$	***
	Text Style Transfer	Source Text	Target text	$>$	***
	Dialogue Generation	Dialogue	Response	$<$	*****
	Speech Recognition	Speech	Text	$\geq$?	***
Speech Generation	**Text to Speech**	Text	Speech	$>$	***
	Singing Voice Synthesis	Score	Singing Voice	$>>$	***
	Voice Conversion	Source Voice	Target Voice	$\gg$	**
	Speech Enhancement	Noisy Speech	Clean Speech	$\gg$	*
Image Generation	Pixel Generation	Class ID	Image Pixel	-	*
	Discrete Token Generation		Image Token	-	**

The values in the last two columns are just for reference

Outline

- Overview of NAR generation tasks in NLP/Speech/CV
  - Target-target vs target-source dependency
- Key tasks
  - Neural machine translation
  - Text error correction
  - Speech to text recognition
  - Text to speech/Singing voice synthesis
  - Image generation
- Summary of NAR applications
  - Benefits of NAR for different tasks
  - Addressing target-target dependency (model multimodal distributions)
  - Addressing target-source dependency (learn source-target alignment)
  - Data difficulty vs model capacity
  - Streaming vs NAR, AR vs Iterative NAR

Benefits of NAR: Inference speedup

- Ideal speedup for one pass generation
  - Text sequence: 10~100
  - Speech sequence: spectrogram/500, waveform/80K
  - Image sequence: token/256, pixel/65536
- Iterative based method
  - N/iter, speedup achieved when iter < N
  - For example, in TTS, for a speech/spectrogram sequence with 500 frames, a diffusion model with 1000 steps (e.g., Grad-TTS), then 500/1000 no speedup!
- Inference speedup does not necessarily mean low computation/memory
  - Computation/memory should be similar as AR model
  - NAR leverages parallel computation (e.g., GPU) for speedup

Benefits of NAR: Beyond inference speedup

- Avoid error propagation
  - AR has exposure bias and error propagation, later tokens will be affected by the accumulated errors in previous tokens
  - NAR has no such exposure bias and propagation
- Avoid order bias
  - Image has no left-to-right or right-to-left inductive bias
- Avoid label bias
- Avoid attention collapse
  - AR usually leverages encoder-decoder attention to extract source information
  - However, enc-dec attention is originally designed for text, may not be suitable for other modalities, such as in text-to-speech, attention collapse and word skipping/repeating/error
- Improve controllability
  - AR generates token one-by-one, cannot well control the length or other factors
  - NAR can well control length, and generative models like VAE/Flow/GAN/Diffusion support latent manipulation!

AR is not the only way for generative modeling
Embrace Flow/VAE/Diffusion/GAN and other generative models in various tasks!

Handle multimodal p(x|y)

- Addressing target-source dependency to simplify mutimodal in p(x|y)
  - Learn better target-source alignment
- Addressing target-target dependency to better model multimodal
  - With advanced generative models (Flow/VAE/Diffusion/GAN, etc)

Handle multimodal p(x|y)

- Learn source-target alignment or provide more information to reduce multimodality in p(x|y)

Alignment Method	Task	
Attention	Machine translation	Alignment is vague, not monotonic, and implicit
	Text summarization	
	Text style transfer	
	Dialogue generation	
CTC	Text error correction	Alignment is monotonic and flexible
	Speech recognition	
	Machine translation	
Duration	Text to speech/Singing voice synthesis	Alignment is monotonic and deterministic
	Text error correction	
No	Voice conversion/Speech enhancement	Already aligned well
N/A	Image pixel/token generation	No fine-grained condition

Handle multimodal p(x|y)

- Learn source-target alignment or provide more information to reduce multimodality in p(x|y)
  - Alignment methods: attention, CTC, duration
  - Providing more information
    - e.g., the detailed class tag of an image, a specific class of dog, instead of simply a dog
    - e.g. language ID, speaker ID, more variance information (pitch, prosody) in speech
    - e.g., more context information in text, such as long sequence processing or document translation

Handle multimodal p(x|y)

- Addressing target-target dependency with advanced generative models
  - Flow/VAE/Diffusion/GAN can be well adopted in image/speech generation
  - More works are trying to apply these generative model in text generation
    - Iterative refinement (e.g., masked predict) is similar to diffusion model (VQ-Diffusion)
    - CV/Speech inspire NLP
  - There is a trend to discretize high-resolution continuous data (e.g., image, waveform) into low-resolution discrete tokens (VQ-VAE or Wav2vec)
    - NLP inspire CV/Speech

Data difficulty vs model capacity

- Tradeoff between the data difficulty (the degree of multimodality in p(x|y)) and the model capacity (representation power or model size)
  - Sometimes increase the model capacity can handle the multimodality to some extent
    - Extremely case: one model to learn them all!
  - But in some cases, data difficulty cannot be well addressed by simply increasing model capacity
    - e.g, NAR in NMT, need a teacher model to distill the data to reduce the data difficulty

Streaming vs NAR, AR vs iterative NAR

- Streaming vs NAR: Streaming is a scenario that a model processes in real-time when an input is coming, but not wait for the whole sequence
  - Streaming ASR, NMT in online scenario
  - In this case, two aspects:
    - The advantage of NAR is reduced, since we can process an input chunk very fast even using AR generation
    - The difficulty of NAR is reduced, since we can generate the current chunk in parallel, conditioned on the output of previous chunk
- AR vs iterative NAR
  - AR can be regarded as an extremely case of NAR
  - So all is about tradeoff between accuracy and efficiency

Outline

- Part I: Introduction (Jiatao Gu)
- Part II: Methods (Jiatao Gu)
- Part III: Applications (Xu Tan)
- Part IV: Open Problems (Xu Tan)

Open problems and future directions

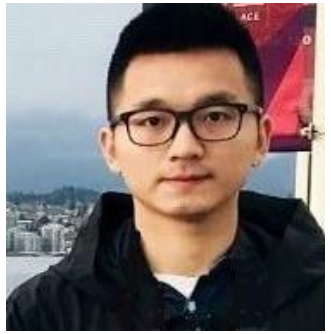
- How to address the multi-modal problem in NAR more effectively and efficiently
  - Improve the accuracy while maintain the advantage of inference speedup
  - Instead of sacrificing the inference speedup with multiple iterations to trade off for accuracy

  - Learn better target-source alignment to provide more target-source dependency to reduce target-target dependency
  - Better model target-target dependency with advanced methods

Open problems and future directions

- Unify all the modality (text, speech, image) in one NAR model?
  - Bridge the modality gap among CV/NLP/Speech
  - Currently we already see some trend, quantize image/speech into discrete tokens, maybe can unify in a single discrete token generation?
- For iterative generation, what connection/relationship can we build between diffusion model (continuous diffusion or VQ-Diffusion), flow model, and iterative refinement methods like Mask-Predict in NLP? What insight we can get to further inspire new methods?
- What is the challenges and opportunities in other sequence generation tasks beyonf NLP/Speech/CV?

Non-Autoregressive Sequence Generation

Jiatao Gu and Xu Tan
Meta AI Research and Microsoft Research Asia

https://github.com/NAR-tutorial/acl2022

Hiring at Microsoft Research Asia!

- Research FTE (social/campus hire)
  - Generative Models and Data Generation
  - Machine Learning, Deep Learning
  - NLP (NMT, Summarization, Conversation, Pre-training, etc)
  - Speech (TTS/ASR)
- Research Intern
  - Speech, Music, NLP, ML

Machine Learning Group, Microsoft Research Asia
Xu Tan xuta@microsoft.com

https://www.microsoft.com/en-us/research/people/xuta/

Thank You!