A COMPARISON OF AUTOREGRESSIVE & NON-AUTOREGRESSIVE APPROACHES USING TRANSFORMER MODEL FOR MACHINE TRANSLATION TASK

Javed Roshan & Gabriel Ohaike
W266: Natural Language Processing with Deep Learning
Professor: Sid J Reddy

Berkeley
UNIVERSITY OF CALIFORNIA

Abstract

- Autoregressive (AR) models are dependent on tokens from previous time-step
- Non-Autoregressive (NAR) models tokens are independent
- AR have high accuracy
- NAR have low latency
- To narrow accuracy gap
  - Using CRF and modifying decoder architecture
- IWSLT dataset for Machine Translation Task
- BLEU: AR (16.07) & NAR (8.79)
- Inference Latency: NAR is 2x faster than AR
- Related Work: Fast Structured Decoding for Sequence Models

  Zhiqing Sun, Zhuohan Li, Haoqing Wang, Di He, Zi Lin and Zhi-Hong Deng

Introduction: Autoregressive Model

- $x = (x_1, x_2, \dots, x_n)$, $y = (y_1, y_2, \dots, y_m)$
- Output tokens are based on chain of conditional probabilities

$$p(y|x) = \Pi_{i=1}^{m} p(y_i | y_{<i}, x)$$

- $y_{<i}$ represents the the tokens before the ith token
- Inference
  - Starts with <bos>
  - Ends when <eos> encountered

Introduction: Non-Autoregressive Model

- $x = (x_1, x_2, \ldots, x_n)$, $y = (y_1, y_2, \ldots, y_m)$
- Output tokens are based on chain of conditional probabilities

$$\mathrm{p}(y|x) = p(m|x)\, p(z|x)\, \Pi_{\mathrm{i=1}}^{\mathrm{m}}\, p(y_i|z, x)$$

- ‘**m**’ represents length of output, y
- ‘**z**’ is the deterministic input to the decoder
- Decoder uses
    - z
    - encoder contextual embedding

Introduction: Non-Autoregressive Model

- Multi-modality problem in output
  - Eg: Thank you (EN)
    - Danke (DE)
    - Danke Schon (DE)
    - Vielen Schon (DE)
  - Can generate - Danke Dank, Danke Schon etc.
- CRF solution

$$p(y|x) = p(m|x)\, p(z|x)\, .\, softmax\left(\Sigma_{i=2}^{m}\, \theta_{i-1,i}(y_{i-1}, y_i) \middle|\, z, x\right)$$

$\theta_{i-1,i}$ is the pairwise potential for (y_{i-1}, y_i).

Architecture: Autoregressive Model

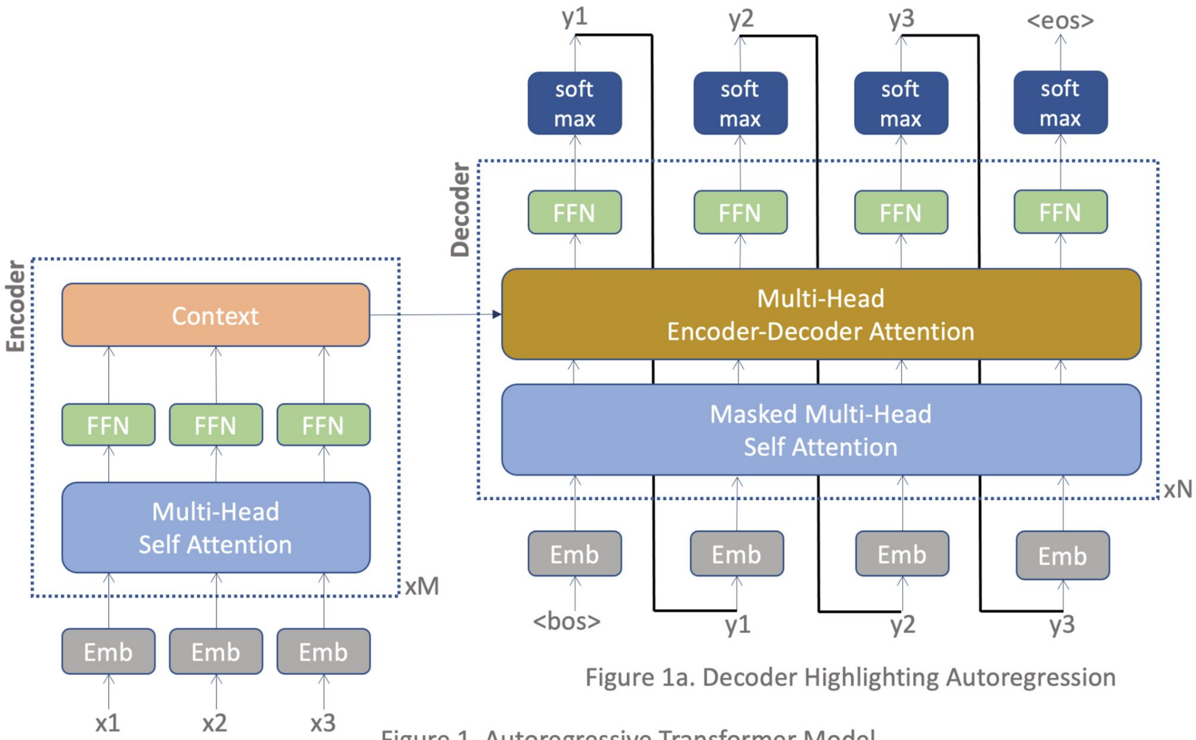

Figure 1a. Decoder Highlighting Autoregression

Figure 1. Autoregressive Transformer Model

Architecture: Non-Autoregressive Model

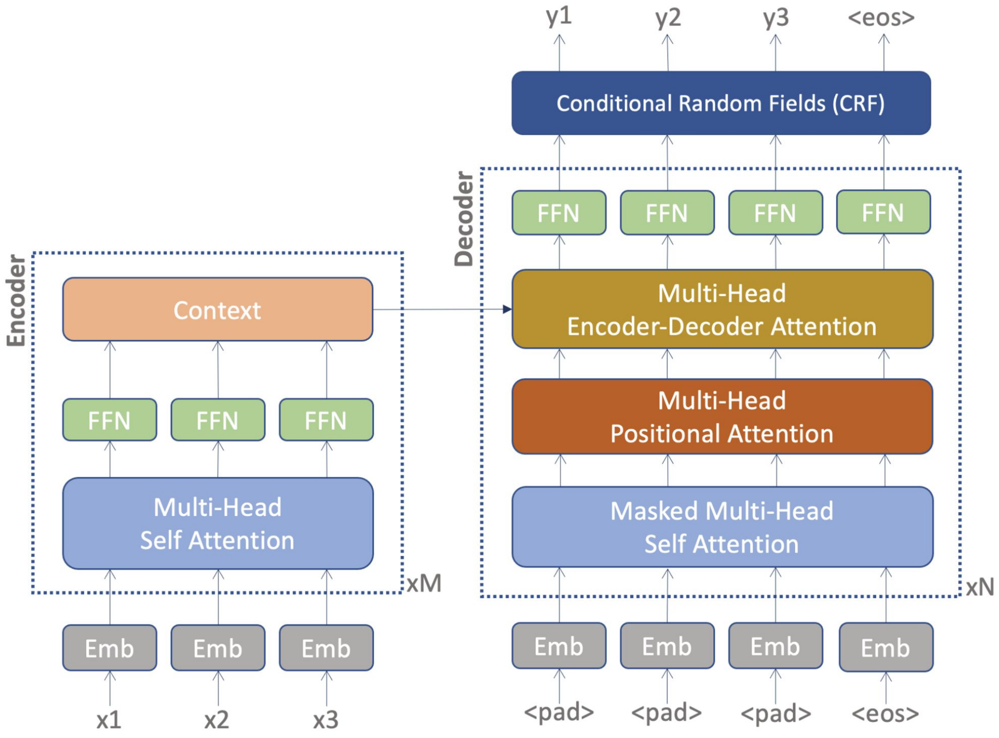

Figure 2. Non-autoregressive Transformer Model
with Conditional Random Fields

Architecture: Non-Autoregressive Model

$$Attention(Q,K,V) = softmax\left(\frac{Q.K^T}{\sqrt{d_{model}}}\right).V$$

d_{model} represents the dimensions of the hidden representations

- Self Attention
  - Q = K = V = x
- Encoder-Decoder Attention
  - Q = hidden representation of previous layer
  - K = V = encoder context
- Positional Attention
  - Q = K = positional embedding
  - V = hidden representation of previous layer

Architecture: Conditional Random Fields

- $x = (x_1, x_2, \ldots., x_n)$

$$\mathrm{p}(y|x) = \frac{1}{Z(x)} \exp[\Sigma_{i=1}^{n} s(y_i, x, i) + \Sigma_{i=2}^{n} t(y_{i-1}, y_i, x, i)]$$

- $Z(x)$ is the normalizing factor
- $s(y_i, x, i)$ is the label score of y_i at position i
- $t(y_{i-1}, y_i, x, i)$ is the transition score from y_{i-1} to y_i
- Optimizing techniques in CRF:
  - Low-rank approximation for transition matrix
  - Beam approximation to estimate normalizing factor
- Negative log-likelihood loss, $L_{CRF} = -\log P(y\,|x)$

Training & Inference - AR Model

- AR model built using pytorch's transformer implementation
  - TransformerEncoder, TransformerEncoderLayer
  - TransformerDecoder, TransformerDecoderLayer
  - PositionalEncoding, Transformer
- Tokenizer: spaCy
- IWSLT dataset
- Custom code
  - AR model wrapper for word and positional embeddings
  - Inference

Training Environment for AR Model

- https://gpu.land
- Tesla v100 single GPU with 16GB memory & 200GB disk space
- Hyper parameters

  number of epochs: ***1024;*** *learning rate: 3e-4; batch size: 32; embedding size: 512; number of heads in the attention layer: 8; number of encoder layers: 6; number of decoder layers: 6; activation function: reLu; dropout: 10% (0.1); optimizer: Adam; optimizer betas: (0.9, 0.98)*
- BLEU score of 15.67
- ~20 hours of training

Training & Inference - NAR Model

- NAR model with baseline code from Facebook’s fairseq library
- IWSLT & WMT dataset (DE to EN)
- Custom code
    - Modified decoder architecture
    - Positional Attention
- Pre-processed dataset; combined source & target dictionaries
- Leveraged checkpoints

Training Environment for NAR Model

- https://gpu.land
- Tesla v100 single GPU with 16GB memory & 200GB disk space
- Hyper parameters

  *number of epochs: **155**; learning rate: 0.0005; optimizer: Adam; optimizer betas: (0.9, 0.98); number of heads in the attention layer: 8; number of encoder layers: 6; number of decoder layers: 6; dropout: 0.3; CRF low rank: 32; CRF beam-approx.: 64*

- BLEU score of 9.26
- ~10 hours

Analysis

TABLE I
MODEL PERFORMANCE

Epochs	Transformer Models	BLEU Score training/validation	Inference BLEU Score	Latency (tokens/second)
1024	AR Transformer (pytorch based)	15.67	16.07	55
155	NAR Transformer (fairseq based)	9.26	8.79	117
50	AR Transformer (fairseq based)	35.23	34.71	210

- NAR is 2x faster than AR (pytorch impl.)
- Fairseq’s AR impl. is almost 80% faster than NAR

Analysis - 2

TABLE II
TRANSLATIONS

Transformer Models	Text
AR Transformer (pytorch based)	**Source:** ein mann rührt in einem topf in seiner küche . **Predicted:** a man is stirring a pot in the kitchen . **Actual:** a man stirs a pot in his kitchen.
	Source: ein mann in einem roten hemd betritt ein etablissement . **Predicted:** a man in a red inside a small glass . **Actual:** a man in a red shirt enters an establishment.
NAR Transformer (fairseq based)	**Source:** und weil uns nichts wichtiger ist als unser überleben , ist die erste haltestelle für all die informationen ein teil unseres temporallappens , die amygdala **Predicted:** and because nothing is more important to us than survival , the first stop of all of that data is an ancient sliver of the temporal lobe called the amygdala . **Actual:** and because nothing is more important to us than our survival, the first stop for all the information is a part of our temporal lobe, the amygdala

Challenges

- Baseline Transformer multiple implementations
- Fairseq debugging was non-trivial
  - pyx files & cythonize
  - debugging cpp
  - environment setup
  - model architecture changes
- Model training from scratch
  - explored all cloud providers before gpu.land

Conclusion

- Autoregressive are still accurate
- Fairseq has highly performant code
- Multiple other NAR models are available to research
- Fun to get into the nitty-gritty of transformers
- Did not replicate the BLEU scores from the papers. But, we successfully ran both models & became transformer savvy:-)
- https://github.com/jroshanucb/deep_learning

Architecture: Non-Autoregressive Model

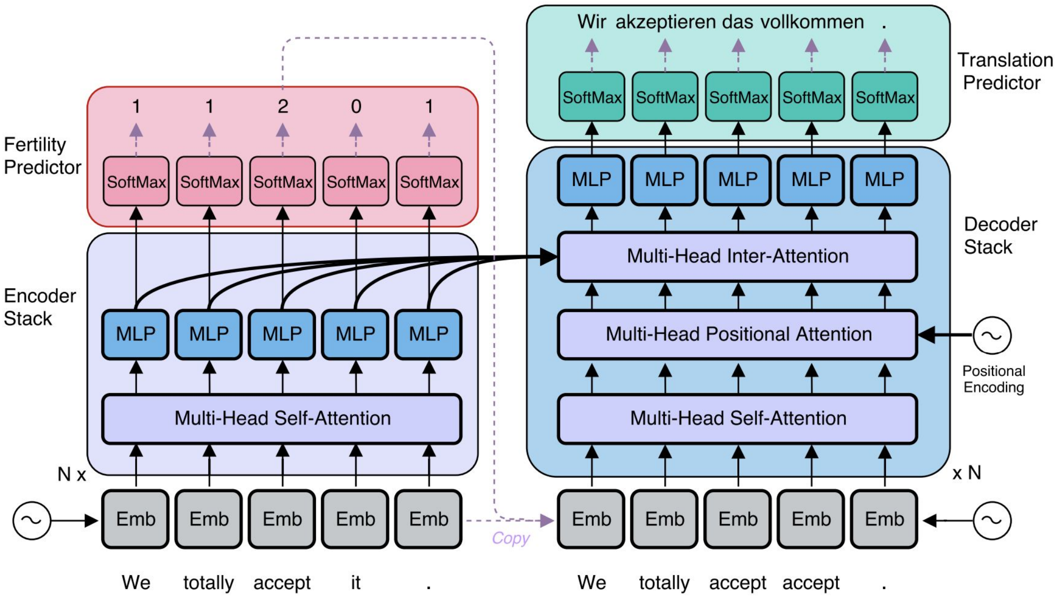

Figure 2: The architecture of the NAT, where the black solid arrows represent differentiable connections and the purple dashed arrows are non-differentiable operations. Each sublayer inside the encoder and decoder stacks also includes layer normalization and a residual connection.

[Non-Autoregressive Neural Machine Translation]

Jiatao Gu, et al.

Architecture: Non-Autoregressive Model

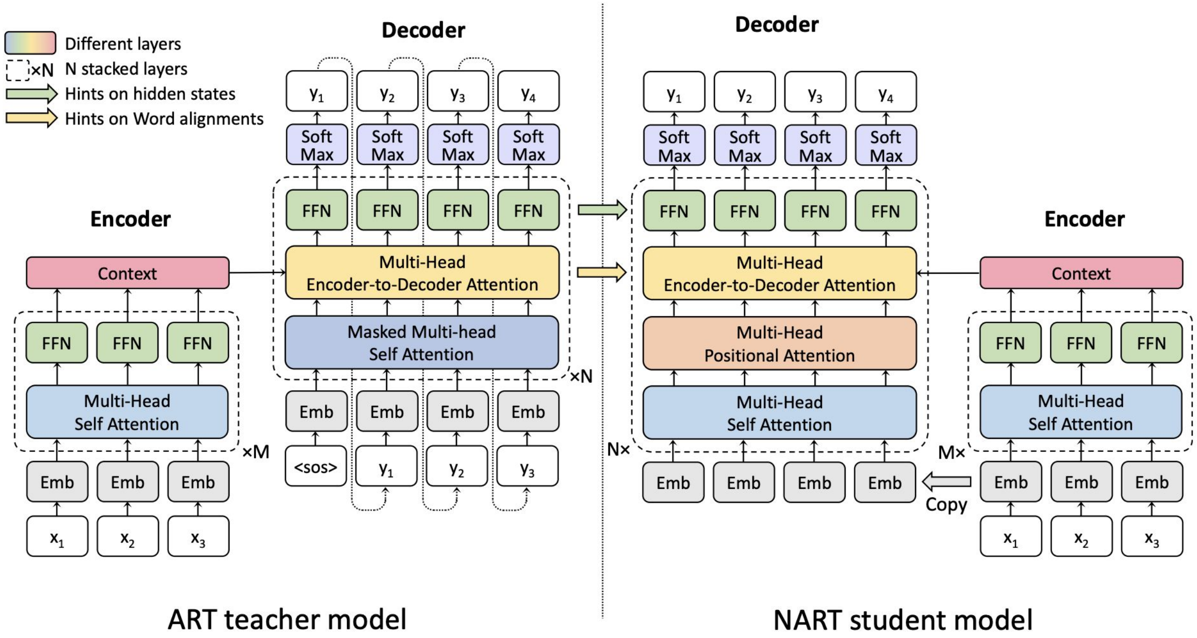

Figure 3: Hint-based training from ART model to NART model.

[Hint Based Training For Non-Autoregressive Translation](#)
Zhuohan Li et al.