Attention Mechanism

Lecture 5

Konstantin Yakovlev [1]

[1]MIPT
Moscow, Russia

MIPT 2024

Recap

- RNN for Language modeling
- RNN backpropagation and related issues
- LSTM and GRU networks
- Layer Normalization and dropout mechanisms
- Applications: Neural Nachine Translation, Image Captioning
- Mixture-of-Experts Layer
- State Space Models

Sequence copying problem[1]

Problem: train an RNN to copy a sequence $\mathbf{x}_{1:n}$. Here $\mathbf{s}_j$ are source hidden states.

Challenge: it is not guaranteed that the output sequence could be restored from a single vector.

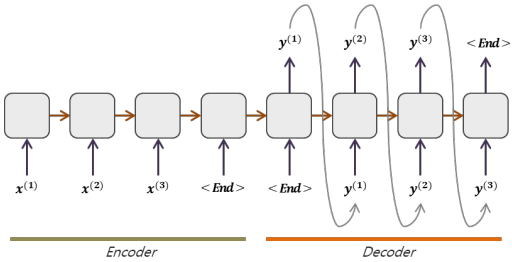
Solution: Attention mechanism.

Given a similarity function

$$\text{sim} : \mathbb{R}^{\dim \mathbf{h}} \times \mathbb{R}^{\dim \mathbf{h}} \to \mathbb{R}.$$

Define an alignment vector $\mathbf{a}_t \in \mathbb{R}^n$

$$\mathbf{a}_t(j) = \frac{\exp(\text{sim}(\mathbf{s}_j, \mathbf{h}_t))}{\sum_{s'=1}^{n} \exp(\text{sim}(\mathbf{s}_{s'}, \mathbf{h}_t))}, \quad j = \overline{1, n}.$$



Compute a context vector and the target distribution

$$\mathbf{c}_t = \sum_{j=1}^{n} \mathbf{a}_t(j)\mathbf{s}_j, \quad p(y_{t+1}|\mathbf{y}_{<t}, \mathbf{x}) = \mathbf{f}(\mathbf{c}_t, \mathbf{h}_t).$$

[1] Luong et. al, Effective Approaches to Attention-based Neural Machine Translation, 2015

Similarity function

Content-based similarity function:

$$\text{sim}(\mathbf{s}, \mathbf{h}) = \begin{cases} \frac{\mathbf{s}^\top \mathbf{h}}{\sqrt{\dim \mathbf{s}}} & \text{dot} \\ \mathbf{h}^\top \mathbf{W}_a \mathbf{s} & \text{general} \\ \mathbf{v}^\top \tanh(\mathbf{W}_a [\mathbf{h}; \mathbf{s}]) & \text{concat} \end{cases}$$

Here $\mathbf{W}_a \in \mathbb{R}^{\dim \mathbf{h} \times \dim \mathbf{s}}$ for the general case, $\mathbf{W}_a \in \mathbb{R}^{\dim \mathbf{v} \times (\dim \mathbf{h} + \dim \mathbf{s})}$ for the "concat" case.

Normalizing the attention scores:

Let $h_i, s_i \overset{i.i.d}{\sim} \mathcal{N}(0, 1)$

$$\mathbb{E}[\mathbf{h}^\top \mathbf{s}] = \sum_{i=1}^{\dim \mathbf{h}} \mathbb{E} h_i s_i = 0,$$

$$\mathbb{V}\left[\frac{\mathbf{h}^\top \mathbf{s}}{\sqrt{\dim \mathbf{h}}}\right] = \frac{1}{\dim \mathbf{h}} \sum_{i=1}^{\dim \mathbf{h}} \mathbb{E} h_i^2 s_i^2 = 1.$$

Local Attention

Challenge: The global attention has to attend to all words in the source sentence which is expansive.

Solution: The local attention mechanism.

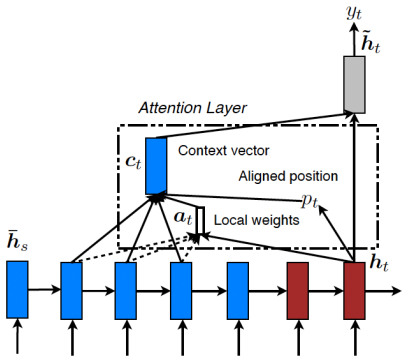
1) *Monotonic* alignment (**local-m**): $p_t = t$, $j \in \overline{p_t - D, p_t + D}$.

$$\mathbf{a}_t(j) = \frac{\exp(\mathrm{sim}(\mathbf{s}_j, \mathbf{h}_t))}{\sum\limits_{p_t - D \le s' \le p_t + D} \exp(\mathrm{sim}(\mathbf{s}_{s'}, \mathbf{h}_t))};$$

2) *Predictive* alignment (**local-p**)

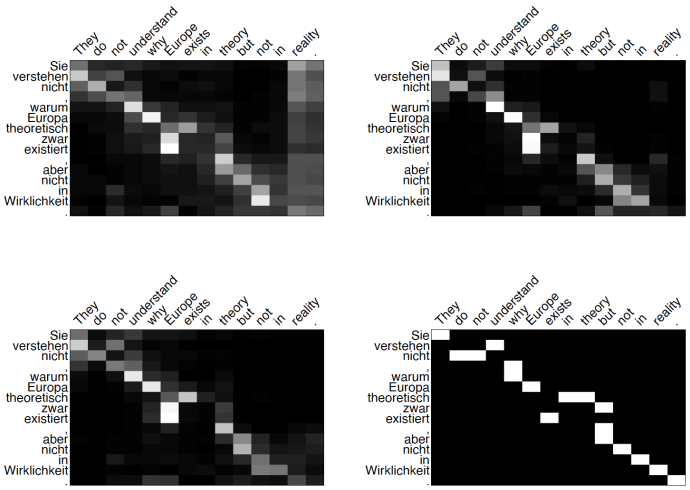
$$p_t = |\mathbf{x}| \cdot \sigma(\mathbf{v}_p^\top \tanh(\mathbf{W}_p \mathbf{h}_t)),$$

$$\tilde{\mathbf{a}}_t(j) = \mathbf{a}_t(j) \exp\left(\frac{-(j - p_t)^2}{2\sigma^2}\right), \quad \sigma = \frac{D}{2}.$$



System	Ppl	BLEU	
		Before	After unk
global (location)	6.4	18.1	19.3 (+1.2)
global (dot)	6.1	18.6	20.5 (+1.9)
global (general)	6.1	17.3	19.1 (+1.8)
local-m (dot)	>7.0	x	x
local-m (general)	6.2	18.6	20.4 (+1.8)
local-p (dot)	6.6	18.0	19.6 (+1.9)
local-p (general)	**5.9**	**19**	**20.9 (+1.9)**

Alignment visualization



Attention weights: top left – global, top right – **local-m**, bottom left – **local-p**, bottom-right – *gold*

Copy Mechanism[2]

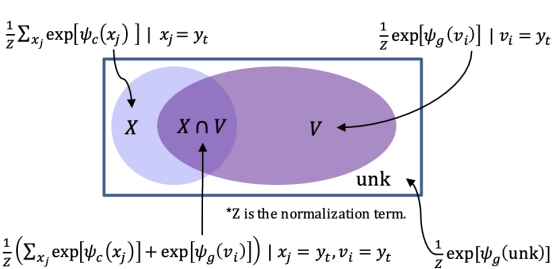
Challenge: the encoder-decoder architectures heavily rely on the representation of "meaning", which may lead to inaccurate outputs when the model needs to copy an input subsequence

Solution: copy mechanism Let $\mathcal{X} = \{x_1, \ldots, x_{T_s}\}$, $\mathcal{V} = \{v_1, \ldots, v_N\}$.

$$p(y_t|\mathbf{h}_t, y_{<t}, \mathbf{c}_t) = \frac{1}{2}p_g(y_t|.) + \frac{1}{2}p_c(y_t|.),$$

$$p_c(y_t|.) = \frac{1}{Z}\sum_{j:x_j=y_t} e^{\psi_c(x_j)} I[y_t \in \mathcal{X}].$$

$$p_g(y_t|.) = \begin{cases} \frac{1}{Z}e^{\psi_g(y_t)}, & y_t \in \mathcal{V} \\ 0, & y_t \in \mathcal{X} \cap \overline{\mathcal{V}} \\ \frac{1}{Z}e^{\psi_g(\text{UNK})}, & \text{otherwise} \end{cases}$$



Generate-Mode:

$$\psi_g(y_t = v_i) = \mathbf{v}_i^\top \mathbf{W}_o \mathbf{h}_t, \quad v_i \in \mathcal{V} \cup \{\text{UNK}\}$$

Copy-Mode:

$$\psi_c(y_t = x_j) = \tanh(\mathbf{s}_j^\top \mathbf{W}_c)\mathbf{h}_t, \quad x_j \in \mathcal{X}$$

[2] Gu et. al, Incorporating Copying Mechanism in Sequence-to-Sequence Learning, 2016

Copy Mechanism

Rule-type	x → ∅	x → x	x → xx	xy → x	xy → xy
Enc-Dec	**100**	3.3	1.5	2.9	0.0
RNNSearch	99.0	69.4	22.3	40.7	2.6
CopyNet	97.3	**93.7**	**98.3**	**68.2**	**77.5**

Enc-Dec – the RNN Encoder-Decoder without attention mechanism, RNNSearch – with attention mechanism.

Models		ROUGE scores on LCSTS (%) R-1	R-2	R-L
RNN (Hu et al., 2015)	+C	21.5	8.9	18.6
	+W	17.7	8.5	15.8
RNN context (Hu et al., 2015)	+C	29.9	17.4	27.2
	+W	26.8	16.1	24.1
CopyNet	+C	**34.4**	**21.6**	**31.3**
	+W	**35.0**	**22.3**	**32.0**

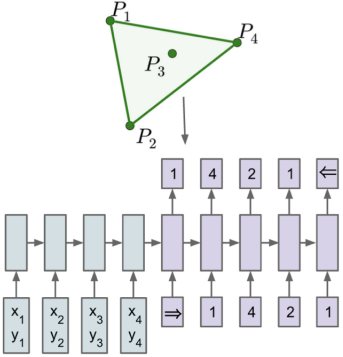
Results

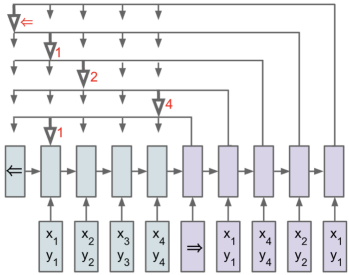
It can be clearly seen that the proposed approach outperforms naive baselines by a huge margin.

Pointer Networks[3]

Challenge: problems such as sorting variable sized sequences require the size of the output dictionary to be fixed a priori.

Solution: a new Pointer Net architecture.





[3] Vinyals O. et. al, Pointer Networks, 2015

Pointer Networks

Content Based Input Attention: produces the entire output sequence using the fixed vocabulary size

$$\mathbf{u}_j^t = \mathbf{v}^\top \tanh(\mathbf{W}_1 \mathbf{s}_j + \mathbf{W}_2 \mathbf{h}_t), \quad j = \overline{1, n}$$
$$\mathbf{a}^t = \mathrm{softmax}(\mathbf{u}^t),$$
$$\mathbf{h}'_t = \sum_{j=1}^{n} a_j^t \mathbf{s}_j,$$
$$p_\theta(y_t | y_{<t}, x_{1:n}) = \mathrm{softmax}(\mathbf{W}_o[\mathbf{h}_t, \mathbf{h}'_t] + \mathbf{d}_o).$$

Sequence-to-sequence model

Ptr-Net: the output dictionary size depends on the number of elements in the input sequence:

$$\mathbf{u}_j^t = \mathbf{v}^\top \tanh(\mathbf{W}_1 \mathbf{s}_j + \mathbf{W}_2 \mathbf{h}_t),$$
$$p_\theta(y_t | y_{<t}, x_{1:n}) = \mathrm{softmax}(\mathbf{u}^t), \quad j = \overline{1, n}.$$

$$\boldsymbol{\theta}^* = \arg\max \sum_{(x,y) \in \mathfrak{D}} \log p(y_{1:T} | x_{1:n}, \boldsymbol{\theta})$$

Pointer Networks: experimental results

Convex Hull Problem

Method	trained n	n	Accuracy	Area
LSTM [1]	50	50	1.9%	FAIL
+attention [5]	50	50	38.9%	99.7%
Ptr-Net	50	50	72.6%	99.9%
LSTM [1]	5	5	87.7%	99.6%
Ptr-Net	5-50	5	92.0%	99.6%
LSTM [1]	10	10	29.9%	FAIL
Ptr-Net	5-50	10	87.0%	99.8%
Ptr-Net	5-50	50	69.6%	99.9%
Ptr-Net	5-50	100	50.3%	99.9%
Ptr-Net	5-50	200	22.1%	99.9%
Ptr-Net	5-50	500	1.3%	99.2%

It can be seen that Ptr-Net outperforms LSTM+Attention baseline.

Travelling Salesman Problem

n	Optimal	A1	A2	A3	Ptr-Net
5	2.12	2.18	2.12	2.12	2.12
10	2.87	3.07	2.87	2.87	2.88
50 (A1 trained)	N/A	6.46	5.84	5.79	6.42
50 (A3 trained)	N/A	6.46	5.84	5.79	6.09
5 (5-20 trained)	2.12	2.18	2.12	2.12	2.12
10 (5-20 trained)	2.87	3.07	2.87	2.87	2.87
20 (5-20 trained)	3.83	4.24	3.86	3.85	3.88
25 (5-20 trained)	N/A	4.71	4.27	4.24	4.30
30 (5-20 trained)	N/A	5.11	4.63	4.60	4.72
40 (5-20 trained)	N/A	5.82	5.27	5.23	5.91
50 (5-20 trained)	N/A	6.46	5.84	5.79	7.66

Interestingly, when using the worst algorithm (A1) data to train the Ptr-Net, the model outperforms the algorithm that is trying to imitate.

Image Captioning[4]

Encoder: Convolutional Features

$$a = \{\mathbf{a}_1, \ldots, \mathbf{a}_L\}, \quad \mathbf{a}_i \in \mathbb{R}^D.$$

Decoder: LSTM with the attention mechanism

$$e_{ti} = f_{\text{att}}(\mathbf{a}_i, \mathbf{h}_{t-1}), \quad \boldsymbol{\alpha}_t = \text{softmax}(\mathbf{e}_t),$$

$$\hat{\mathbf{z}}_t = \sum_{i=1}^{L} \alpha_{ti} \mathbf{a}_i,$$

$$p(y_t | y_{<t}, \mathbf{a}) \propto$$

$$\exp(\text{Linear}(\mathbf{h}_t, \hat{\mathbf{z}}_t, \text{Emb}(y_{t-1}))).$$

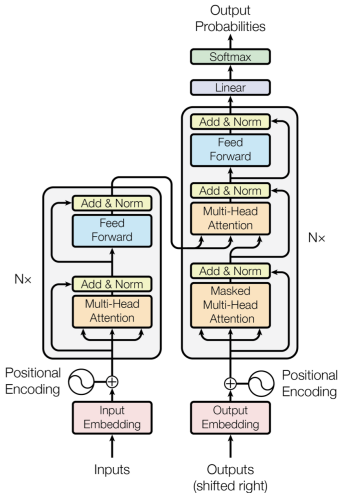
Dataset	Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR
Flickr8k	Google NIC(Vinyals et al., 2014)†♯	63	41	27	—	—
	Log Bilinear (Kiros et al., 2014a)°	65.6	42.4	27.7	17.7	17.31
	Soft-Attention	**67**	44.8	29.9	19.5	18.93
	Hard-Attention	**67**	**45.7**	**31.4**	**21.3**	**20.30**
Flickr30k	Google NIC†°♯	66.3	42.3	27.7	18.3	—
	Log Bilinear	60.0	38	25.4	17.1	16.88
	Soft-Attention	66.7	43.4	28.8	19.1	**18.49**
	Hard-Attention	**66.9**	**43.9**	**29.6**	**19.9**	18.46
COCO	CMU/MS Research (Chen & Zitnick, 2014)^a	—	—	—	—	20.41
	MS Research (Fang et al., 2014)†^a	—	—	—	—	20.71
	BRNN (Karpathy & Li, 2014)°	64.2	45.1	30.4	20.3	—
	Google NIC†°♯	66.6	46.1	32.9	24.6	—
	Log Bilinear°	70.8	48.9	34.4	24.3	20.03
	Soft-Attention	70.7	49.2	34.4	24.3	**23.90**
	Hard-Attention	**71.8**	**50.4**	**35.7**	**25.0**	23.04

A stop sign is on a road with a mountain in the background.

[4] Xu K. et. al, Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, 2016

Transformer Architecture [5]

Challenge: The LSTM is based on recurrent connections which can be hard to parallelize.
Solution: The Transformer architecture
General overview: The encoder maps an input sequence $(x_1, \ldots, x_n)$ to a sequence of contiguous representations $\mathbf{H} = (\mathbf{h}_1, \ldots, \mathbf{h}_n)$. The decoder then generates an output sequence $(y_1, \ldots, y_m)$ given $\mathbf{H}$.



[5] Vaswani A. et. al, Attention Is All You Need, 2017

Transformer Encoder

Given an input sequence $(x_1, \ldots, x_n)$, for each $t = \overline{1, n}$ define

$$\mathbf{h}_t^0 = \mathrm{Embedding}(x_t) + \mathrm{PosEmbedding}(t),$$

where $\mathrm{PosEmbedding} : \mathbb{N} \to \mathbb{R}^d$,

$$\mathrm{PosEmbedding}(t)_{2i} = \sin(t/10000^{2i/d}),$$
$$\mathrm{PosEmbedding}(t)_{2i+1} = \cos(t/10000^{2i/d}).$$

Now consider

$$\begin{aligned}\mathrm{PosEmbedding}(t+k)_{2i} &= \sin((t+k)/c_i) \\ &= \sin(t/c_i)\cos(k/c_i) + \cos(t/c_i)\sin(k/c_i) \\ &= \mathrm{Linear}(\mathrm{PosEmbedding}(t))\end{aligned}$$

Next, feed $\mathbf{H}^0 = [\mathbf{h}_t^0]_{t=1}^n \in \mathbb{R}^{n \times d}$ to the stack of L Transformer Layers:

$$\tilde{\mathbf{H}}^l = \mathrm{Norm}(\mathrm{MHA}(\mathbf{H}^{l-1}, \mathbf{H}^{l-1}, \mathbf{H}^{l-1}) + \mathbf{H}^{l-1}),$$
$$\mathbf{h}_t^l = \mathrm{Norm}(\mathrm{FFN}(\tilde{\mathbf{h}}_t^l) + \tilde{\mathbf{h}}_t^l)$$

where $\mathrm{Norm} \equiv \mathrm{LayerNorm}$, FFN – two linear transformations with a ReLU activation.

$$\mathrm{Attention}(\underbrace{\mathbf{Q}}_{n\times d}, \underbrace{\mathbf{K}}_{n\times d}, \underbrace{\mathbf{V}}_{n\times d}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V},$$

$$\mathrm{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\mathbf{head}_i]_{i=1}^h \underbrace{\mathbf{W}^o}_{d\times d}, \ \mathbf{head}_i \in \mathbb{R}^{n \times d/h}$$

$$\mathbf{head}_i = \mathrm{Attention}(\mathbf{Q}\underbrace{\mathbf{W}_i^Q}_{d\times d/h}, \mathbf{K}\underbrace{\mathbf{W}_i^K}_{d\times d/h}, \mathbf{V}\underbrace{\mathbf{W}_i^V}_{d\times d/h}).$$

Transformer Decoder

Given an output sequence $(\underbrace{y_0}_{[BOS]}, y_1, \ldots, y_m)$.

For each $t = \overline{1, m}$ define

$$\mathbf{h}_t^0 = \mathrm{Embedding}(y_t) + \mathrm{PosEmbedding}(t).$$

Next apply a stack of L Transofrmer layers with causal mask:

$$\tilde{\mathbf{H}}^l = \mathrm{Norm}(\mathrm{MaskMHA}(\mathbf{H}^{l-1}, \mathbf{H}^{l-1}, \mathbf{H}^{l-1}) + \mathbf{H}^{l-1}),$$
$$\hat{\mathbf{H}}^l = \mathrm{Norm}(\mathrm{MaskMHA}(\tilde{\mathbf{H}}^l, \mathbf{H}^l_{\text{enc}}, \mathbf{H}^l_{\text{enc}}) + \tilde{\mathbf{H}}^l),$$
$$\mathbf{h}_t^l = \mathrm{Norm}(\mathrm{FFN}(\hat{\mathbf{h}}_t^l) + \hat{\mathbf{h}}_t^l),$$

where $\mathrm{MaskedMHA}$ is as follows

$$\mathrm{MaskMHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\mathbf{head}_i]_{i=1}^h \underbrace{\mathbf{W}^o}_{d \times d},$$

$$\mathbf{head}_i = \mathrm{MaskAtt}(\mathbf{Q}\underbrace{\mathbf{W}_i^Q}_{d \times d/h}, \mathbf{K}\underbrace{\mathbf{W}_i^K}_{d \times d/h}, \mathbf{V}\underbrace{\mathbf{W}_i^V}_{d \times d/h}),$$

$$\mathrm{MaskAtt}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}} + \mathbf{M}\right)\mathbf{V},$$

$$M_{ij} = \begin{cases} 0, & i \geq j \\ -\infty, & \text{otherwise} \end{cases}$$

Finally, probability of the next token:

$$\log p(y_t | y_{<t}, x_{1:n}) = \mathrm{softmax}(\underbrace{\mathbf{W}_o}_{|\mathcal{V}| \times d}\mathbf{h}_t^l + \mathbf{b}_o)$$

Autoregressive Generation[6]

Given an input sequence $(x_1, \ldots, x_n)$. We need to generate an output sequence $\hat{y}$, $y_m = [\text{EOS}]$.

Sampling

$$\hat{y}_t \sim p(y_t | y_{<t}, x_{1:n}), \quad y_0 = [\text{BOS}].$$

Greedy Decoding

An exact decoding is intractable:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \sum_{t=1}^{|\mathbf{y}|} \log p(y_t | y_{<t}, x_{1:n})$$

Approximate it with a greedy decoding:

$$\hat{y}_t = \arg\max_{y \in \mathcal{V}} p(y | y_{<t}, x_{1:n}), \quad y_0 = [\text{BOS}].$$

Beam Search

Let b denote the beam size. Then the beam seach operates as follows:

$$\mathcal{B}_0 = \{([\text{BOS}], 1)\},$$
$$\mathcal{B}_i = \text{top}_b\{([y', y_i], s \cdot p(y_i | y, x)) \mid (y, s) \in \mathcal{B}_{i-1}, y_i \in \mathcal{V}\},$$

Finally, we choose the sequence from $\mathcal{B}_N$ with highest model score, where N is the maximum length.

[6] Yang Y. et. al, Breaking the Beam Search Curse, 2018

Transformer performance on the WMT 2014

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

It is crystal clear that the proposed architecture requires up to 100 times less computational cost and is the most effective.

Non-Autoregressive Neural Machine Translation[7]

Challenge: an autoregressive generation is slow, since it requires sequential forward passes.
Solution: generate tokens in parallel with a non-autoregressive decoder.
Remove autoregressive connections:

$$p_{\text{nar}}(y_{1:T}, T|x_{1:n}) = p_L(T|x_{1:n}) \prod_{t=1}^{T} p(y_t|x_{1:n}, T).$$

Multimodality Problem: Let $x =$ Thank you.

$$\mathfrak{D} = \{(x, \text{Danke schon}), (x, \text{Vielen Dank})\}$$
$$p^*_{\text{ar}}(y|x) = 0.5 \cdot I[(x, y) \in \mathfrak{D}],$$
$$p^*_{\text{nar}}(\text{Vielen shon}|x) = p^*_{\text{nar}}(\text{Danke Dank}|x) = \frac{1}{4}.$$

Informative Latent Variables:

$$p_{\text{nar}}(y_{1:T}|x_{1:n}) = \sum_{f_1,\dots,f_n \in \mathcal{F}} \left(\prod_{i=1}^{n} p_F(f_i|x_{1:n}) \cdot \prod_{t=1}^{T} p(y_t|x_{1:n}, f_{1:n}) \right),$$

where $\mathcal{F} = \{(f_1, \dots, f_n) \mid \sum_{i=1}^{n} f_i = T;\ f_i \geq 0\ \forall i\}$ is the set of all fertility sequences.
Training:

$$\mathcal{L}_{\text{nar}} \geq \underbrace{\sum_{i=1}^{n} \log p_F(f^*_i|x_{1:n})}_{\text{fertility loss}} + \underbrace{\sum_{t=1}^{T} \log p(y_t|x_{1:n}, f^*_{1:n})}_{\text{translation loss}},$$

[7] Gu J. et. al, Non-Autoregressive Neural Machine Translation, 2018

Non-Autoregressive Neural Machine Translation

Argmax Decoding

$$\hat{f}_i = \arg\max_{f_i} p_F(f_i|x_{1:n}), \quad \hat{y}_{1:T} = G(x_{1:n}, \hat{f}_{1:n}),$$

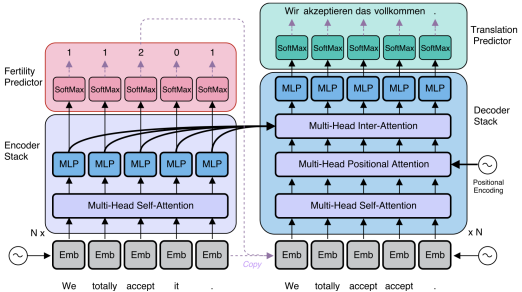
$$G(x_{1:n}, \hat{f}_{1:n})_t = \arg\max_{y_t \in \mathcal{V}} p(y_t|x_{1:n}, \hat{f}_{1:n}).$$

Noisy parallel decoding (NPD) Use the autoregressive teacher to identify the best translation:

$$\hat{y}_{1:T} = G(x_{1:n}, \arg\max_{f_i \sim p_F} p_{\text{ar}}(G(x_{1:n}, f_{1:n})|x_{1:n})).$$

Sequence-level knowledge distillation: Generate a new corpus with an autoregressive teacher to mitigate the multimodality problem.

The architecture of the NAT



Fine-Tuning[a]

$$\mathcal{L}_{FT} = -\mathcal{L}_{(x,y^{\text{teacher}})} \log p_{\text{nar}}(y^{\text{teacher}}_{1:T}|x_{1:n}).$$

[a] See eq. (11) in the original paper.

Non-autoregressive Neural Machine Translation: evaluation

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

The NAT performs between 2-5 BLEU points worse than its autoregressive teacher. There is a speedup of more than a factor of 10 over greedy autoregressive decoding, or a factor of 15 over beam search

Semi-Autoregressive Neural Machine Translation[8]

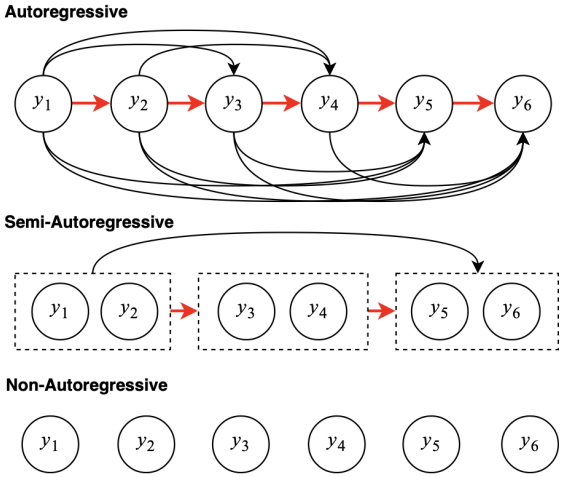
Challenge: while the parallelizability of the NAT is greatly improved, the translation quality encounter much decrease.
Solution: a new SAT architecture
Group-Level Chain Rule:

$$G_t = y_{(t-1)K+1:\,tK}, \quad t = \overline{1, [(T-1)/K] + 1},$$

$$p(y_{1:T}|x_{1:n}) = \prod_{t=1}^{[(T-1)/K]+1} p(G_t|G_{<t}, x_{1:n}).$$

Relaxed Causal Mask:

$$\mathbf{M}_{ij} = I[j < ([(i-1)/K] + 1) \times K]$$



[8] Wang C. et. al, Semi-Autoregressive Neural Machine Translation, 2018

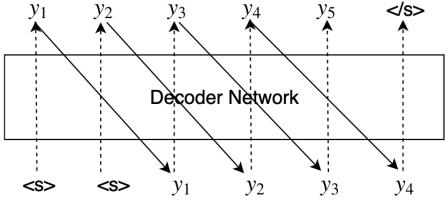
Semi-Autoregressive Neural Machine Translation: evaluation

Evaluation on WMT14 (En-De)

Model	Beam Size	BLEU	Degeneration	Latency	Speedup
Transformer	4	27.11	0%	346ms	1.00×
	1	26.01	4%	283ms	1.22×
Transformer, N=2	4	24.30	10%	163ms	2.12×
	1	23.37	14%	113ms	3.06×
NAT (Gu et al., 2017)	-	17.69	25%	39ms	15.6×
NAT (rescoring 10)	-	18.66	20%	79ms	7.68×
NAT (rescoring 100)	-	19.17	18%	257ms	2.36×
LT (Kaiser et al., 2018)	-	19.80	27%	105ms	-
LT (rescoring 10)	-	21.00	23%	-	-
LT (rescoring 100)	-	22.50	18%	-	-
IRNAT (Lee et al., 2018)	-	18.91	22%	-	1.98×
This Work					
SAT, K=2	4	26.90	1%	229ms	1.51×
	1	26.09	4%	167ms	2.07×
SAT, K=4	4	25.71	5%	149ms	2.32×
	1	24.67	9%	91ms	3.80×
SAT, K=6	4	24.83	8%	116ms	2.98×
	1	23.93	12%	62ms	5.58×

The SAT achieves a better balance between translation quality and decoding speed.

Long-Distance Prediction: At the beginning of generation feed the model with K begin of sequence tokens $[\mathrm{BOS}]$ to predict $y_{1:K}$ in parallel. Subsequently, decode the next group.



BERT: Bidirectional Encoder Representations from Transformers[9]

Challenge: the major limitation is that standard language models are unidirectional which sub-optimal for sentence-level tasks.

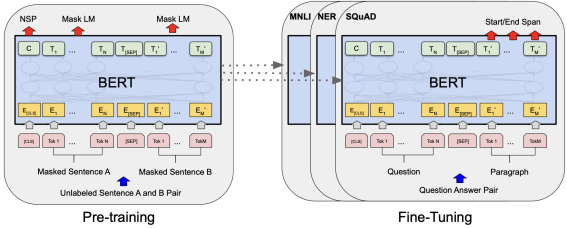
Solution: introduce BERT equipped with a Masked Language Modeling objective.

Pre-training: Masked Language Modeling: given a sequence $x_{1:n}$. MLM randomly selects a subset of tokens positions $\mathcal{M}$ and replaces them with a $[\mathrm{MASK}]$ token (80% of the time), random token (10%), same token (10%).

$$\mathcal{L}_{\text{MLM}} = -\mathbb{E}_x \sum_{i \in \mathcal{M}} \log p_\theta(x_i | \mathrm{Mask}(x)).$$

Next Sentence Prediction: given a sentences A and B, where 50% of the time B is the actual next sentence, and 50% of the time is randomly taken from a collection.

$$\mathcal{L}_{\text{NSP}} = -\mathbb{E}_x \log p_\theta(y_{\text{NSP}} | [\mathrm{A}, [\mathrm{SEP}], \mathrm{B}]).$$



[9] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019

BERT: evaluation

Evaluation on GLUE benchmark

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Both $BERT_{\text{base}}$ and $BERT_{\text{large}}$ outperform all systems on all tasks by a substantial margin.

Ablation on different masking strategies

Masking Rates			Dev Set Results		
MASK	SAME	RND	MNLI Fine-tune	NER Fine-tune	NER Feature-based
80%	10%	10%	84.2	95.4	94.9
100%	0%	0%	84.3	94.9	94.0
80%	0%	20%	84.1	95.2	94.6
80%	20%	0%	84.4	95.2	94.7
0%	20%	80%	83.7	94.8	94.6
0%	0%	100%	83.6	94.9	94.6

Fine-tuning is surprisingly robust to different masking strategies.

Summary

- Global and local attention mechanisms
- Copy Mechanism
- Pointer Networks
- Image Captioning
- Transformer architecture
- Non-autoregressive Neural Machine Translation
- Semi-Autoregressive Neural Machine Translation
- BERT