Advanced Models for Language

EE807: Recent Advances in Deep Learning

Lecture 19

Slide made by

Sangwoo Mo

KAIST EE

Table of Contents

Table of Contents

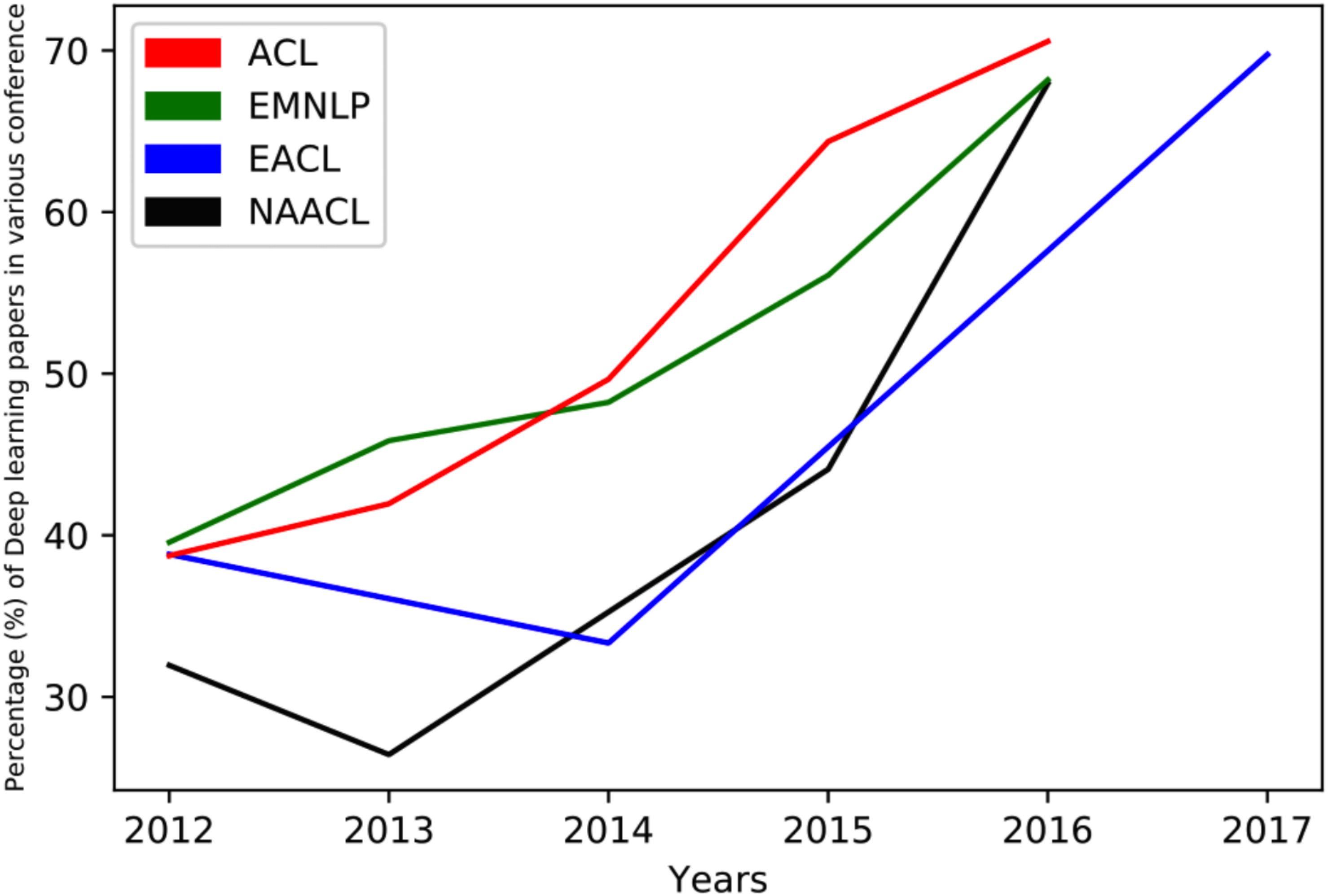
1. **Introduction**
   - Why deep learning for NLP?
   - Overview of the lecture

2. **Network Architecture**
   - Learning long-term dependencies
   - Improve softmax layers

3. **Training Methods**
   - Reduce exposure bias
   - Reduce loss/evaluation mismatch
   - Extension to unsupervised setting

Why Deep Learning for Natural Language Processing (NLP)?

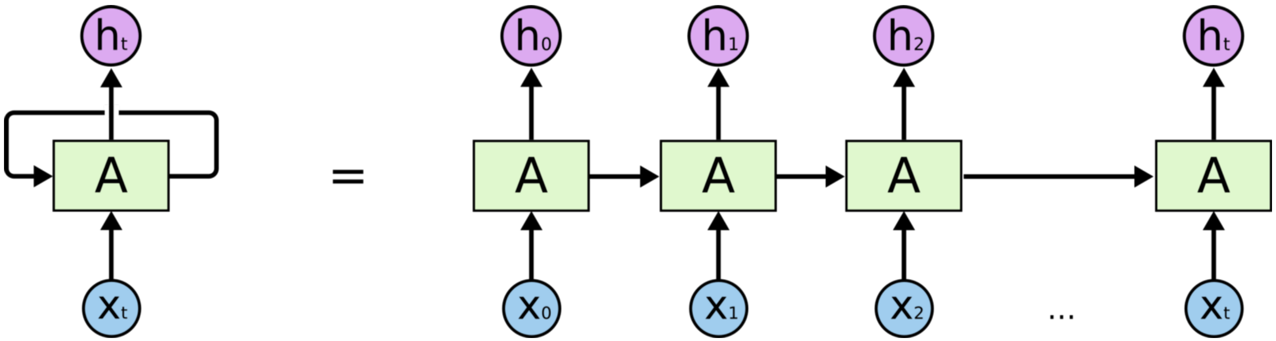
- **Deep learning** is now commonly used in **natural language processing (NLP)**



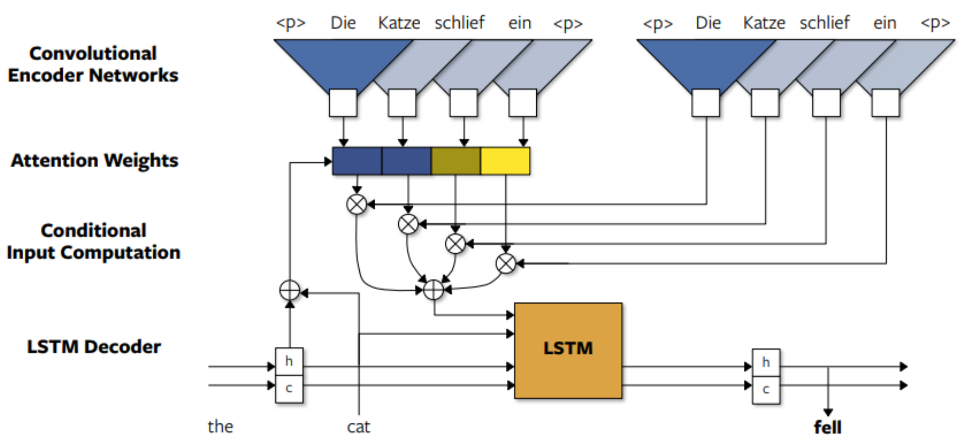
*Source: Young et al. "Recent Trends in Deep Learning Based Natural Language Processing", arXiv 2017

Recap: RNN & CNN for Sequence Modeling

- **Language** is sequential: It is natural to use RNN architectures
  - **RNN** (or LSTM variants) is a natural choice for sequence modelling



- **Language** is translation-invariant: It is natural to use CNN architectures
  - One can use **CNN** [Gehring et al., 2017] for parallelization



*Source: https://towardsdatascience.com/introduction-to-recurrent-neural-network-27202c3945f3
Gehring et al. "Convolutional Sequence to Sequence Learning", ICML 2017

Limitations of prior works

- However, **prior works** have several limitations…

- **Network architecture**
  - Long-term dependencies: Network **forgets** previous information as it summarizes inputs into a **single** feature vector
  - Limitations of softmax: **Computation** linearly increases to the vocabulary size, and **expressivity** is bounded by the feature dimension

- **Training methods**
  - Exposure bias: Model only sees **true** tokens at training, but it sees **generated** tokens at inference (and noise accumulates sequentially)
  - Loss/evaluation mismatch: Model uses **MLE** objective at training, but use **other evaluation metrics** (e.g., BLEU score [Papineni et al., 2002]) at inference
  - Unsupervised setting: How to train models if there are **no paired** data?

Table of Contents

Attention [Bahdanau et al., 2015]

- Motivation:
  - Previous models **summarize** inputs into a single feature vector
  - Hence, the model forgets old inputs, especially for long sequences

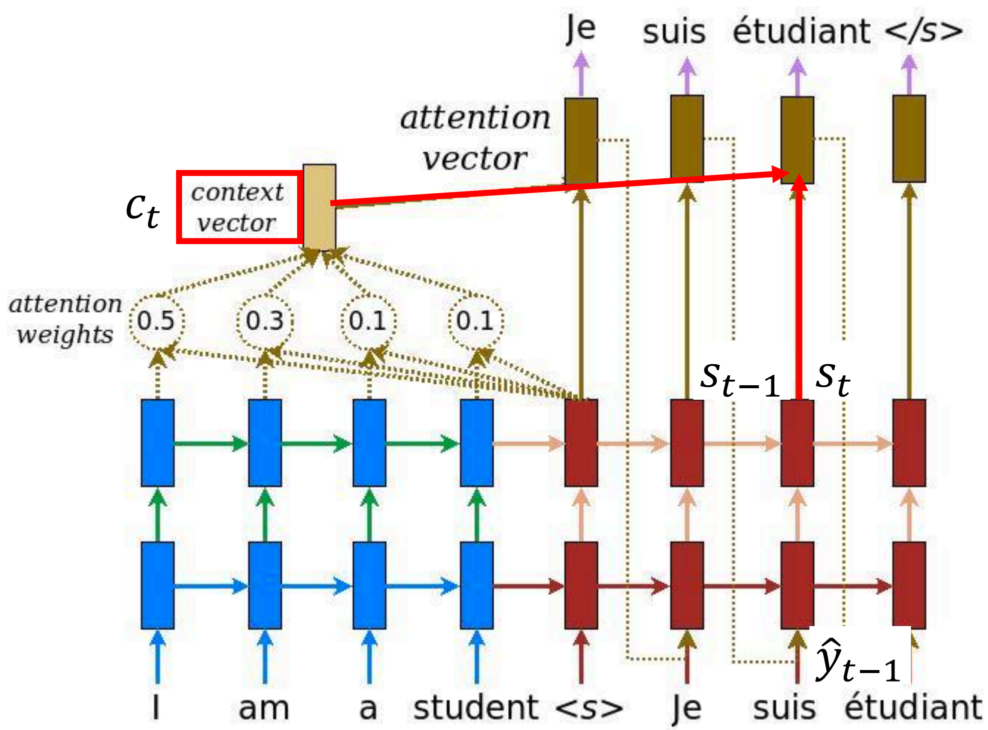
- Idea:
  - Use input features, but attend on the most importance features

  - Example) Translate **“Ich mochte ein bier”** ⇔ **“I’d like a beer”**
  - Here, when the model generates **“beer”**, it should attend on **“bier”**

*Source: https://ratsgo.github.io/from%20frequency%20to%20semantics/2017/10/06/attention/

Attention [Bahdanau et al., 2015]

- Method:
  - **Task:** Translate source sequence $[x_1, \dots, x_n]$ to target sequence $[y_1, \dots, y_m]$
  - Now the **decoder hidden state** s_t is a function of previous state s_{t-1}, current input $\hat{y}_{t-1}$, and context vector c_t, i.e., $s_t = f(s_{t-1}, \hat{y}_{t-1}, c_t)$



*Source: https://medium.com/syncedreview/a-brief-overview-of-attention-mechanism-13c578ba9129

- Method:
  - **Task:** Translate source sequence $[x_1, \dots, x_n]$ to target sequence $[y_1, \dots, y_m]$
  - Now the **decoder hidden state** s_t is a function of previous state s_{t-1}, current input $\hat{y}_{t-1}$, and context vector c_t, i.e., $s_t = f(s_{t-1}, \hat{y}_{t-1}, c_t)$
  - The context vector c_t is **linear combination** of input hidden features $[h_1, \dots, h_n]$

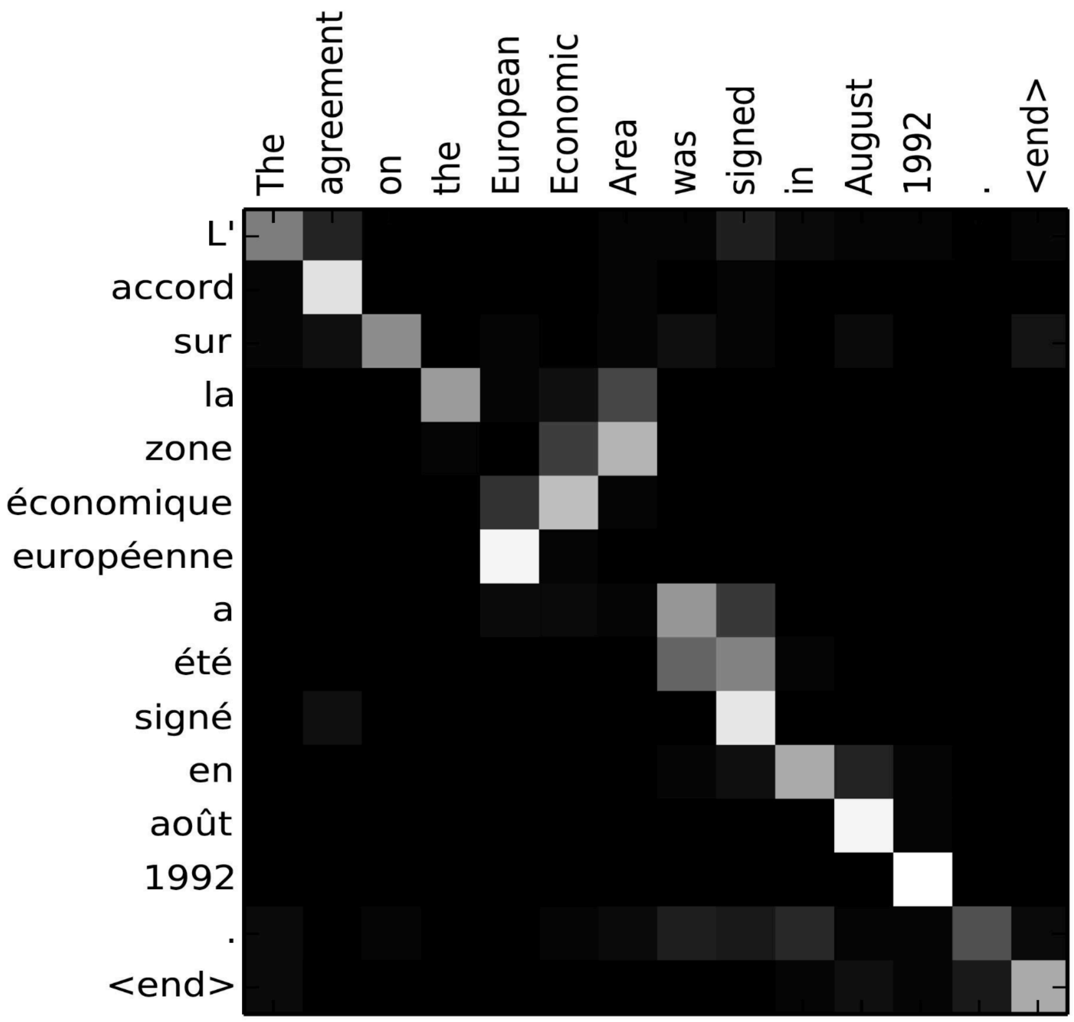
$$c_t = \sum_{i=1}^{n} \alpha_{t,i} h_i$$

  - Here, the weight $\alpha_{t,i}$ is alignment score of two words y_t and x_i

$$\alpha_{t,i} = \text{align}(y_t, x_i) = \frac{\text{score}(s_{t-1}, h_i)}{\sum_{i'} \text{score}(s_{t-1}, h_{i'})}$$

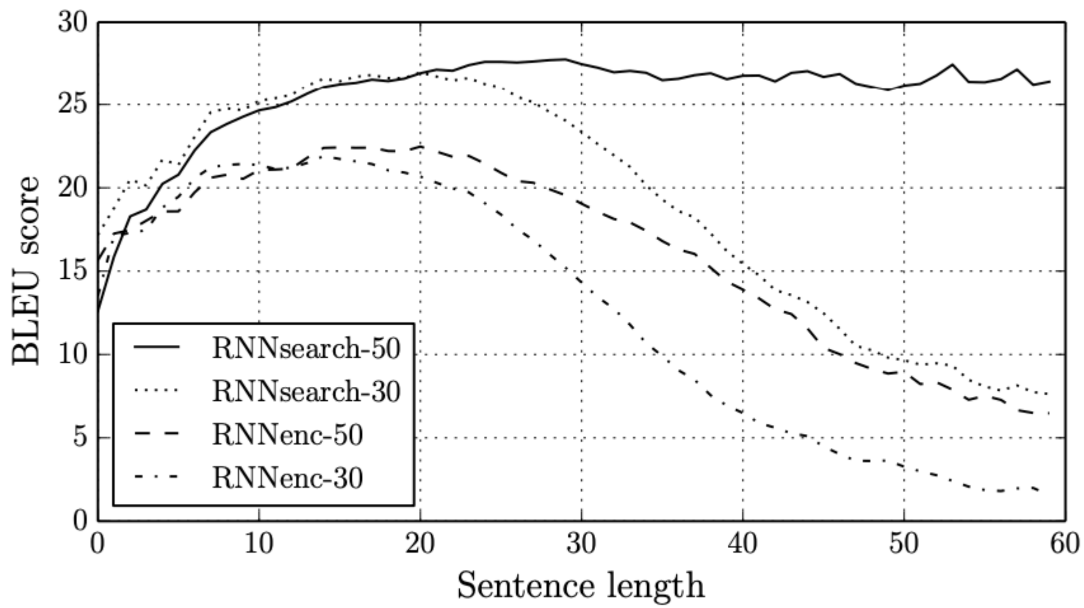
where score is also jointly trained, e.g., $\text{score}(s_t, h_i) = \mathbf{v}^T \tanh(\mathbf{W}[s_t; h_i])$

- Results: Attention shows good correlation between source and target



Attention [Bahdanau et al., 2015]

- Results: Attention improves machine translation performance
  - **RNNenc**: no attention / **RNNsearch**: with attention / **#**: max length of train data



Model	All	No UNK°
RNNencdec-30	13.93	24.19
RNNsearch-30	21.50	31.44
RNNencdec-50	17.82	26.71
RNNsearch-50	26.75	34.16
RNNsearch-50*	28.45	36.15
Moses	33.30	35.63

No UNK: omit unknown words
*: longer train until converge

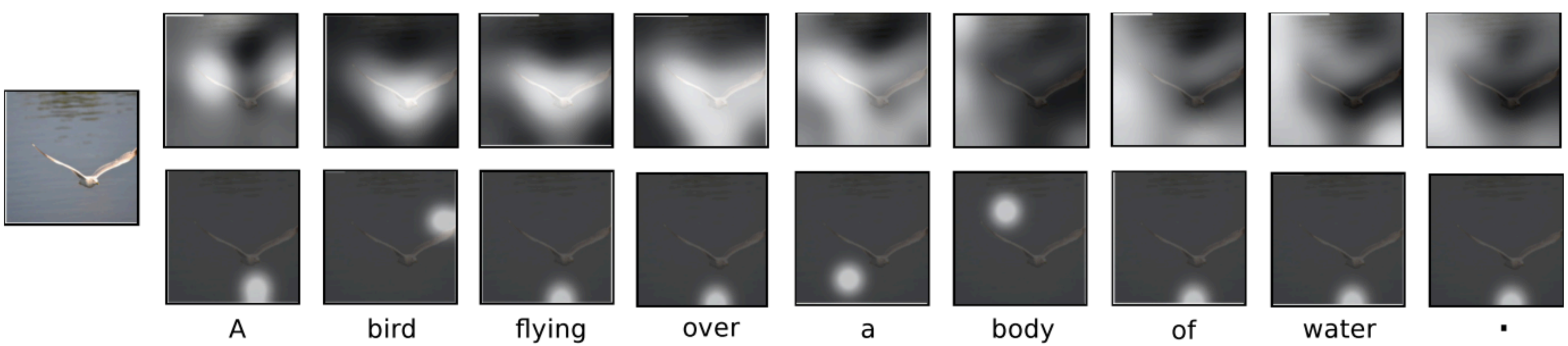
Show, Attend, and Tell [Xu et al., 2015]

- Motivation: Can apply **attention** for image captioning?
  - **Task:** Translate source image $[x]$ to target sequence $[y_1, \dots, y_m]$
  - Now attend on specific **location** on the image, not the words

- Idea: Apply attention to convolutional features $[h_1, \dots, h_K]$ (with K channels)
  - Apply deterministic **soft** attention (as previous one) and stochastic **hard** attention (pick one h_i by sampling multinomial distribution with parameter α)

  - Hard attention picks more **specific** area and shows **better** results, but training is less stable due to the **stochasticity** and **differentiability**

Up: hard attention / Down: soft attention



Show, Attend, and Tell [Xu et al., 2015]

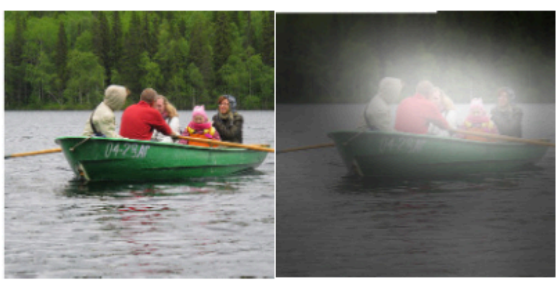
- Results: Attention picks visually plausible locations

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

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

A little girl sitting on a bed with a teddy bear.



A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Show, Attend, and Tell [Xu et al., 2015]

- Results: Attention improves the image captioning performance

		BLEU				
Dataset	Model	B-1	B-2	B-3	B-4	METEOR
Flickr8k	Google NIC(Vinyals et al., 2014)$^{\dagger\Sigma}$	63	41	27	—	—
	Log Bilinear (Kiros et al., 2014a)$^{\circ}$	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$^{\dagger\circ\Sigma}$	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)$^{\dagger a}$	—	—	—	—	20.71
	BRNN (Karpathy & Li, 2014)$^{\circ}$	64.2	45.1	30.4	20.3	—
	Google NIC$^{\dagger\circ\Sigma}$	66.6	46.1	32.9	24.6	—
	Log Bilinear$^{\circ}$	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

Transformer [Vaswani et al., 2017]

- Motivation:
  - Prior works use RNN/CNN to solve **sequence-to-sequence** problems
  - **Attention** already handles *arbitrary length* of sequences, *easy to parallelize*, and not suffer from *forgetting* problems… Why should one use RNN/CNN modules?

- Idea:
  - Design architecture **only using** attention modules
  - To extract features, the authors use self-attention, that features **attend on itself**
    - Self-attention has many advantages over RNN/CNN blocks

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

n: sequence length, d: feature dimension, k: (conv) kernel size, r: window size to consider
Maximum path length: maximum traversal between any two input/outputs **(lower is better)**

*Cf. Now self-attention is widely used in other architectures, e.g., CNN [Wang et al., 2018] or GAN [Zhang et al., 2018]

Transformer [Vaswani et al., 2017]

- Multi-head attention: The building block of the Transformer
  - In previous slide, we introduced ***additive* attention** [Bahdanau et al., 2015]
  - Here, the context vector is a **linear combination** of
    - weight $\alpha_{t,i}$, a function of inputs $[x_i]$ and output y_t
    - and input hidden states $[h_i]$

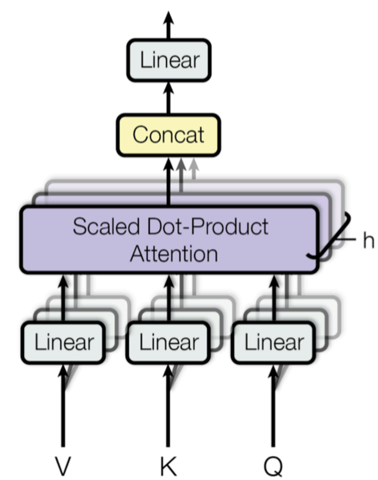
$$c_t = \sum_{i=1}^{n} \alpha_{t,i} h_i$$

  - In general, attention is a function of **key** K, **value** V, and **query** Q
    - **key** $[x_i]$ and **query** y_t defines weights $\alpha_{t,i}$, which are applied to **value** $[h_i]$
    - For sequence length T and feature dimension d, (K, V, Q) are $T \times d$, $T \times d$, and $1 \times d$ matrices

  - Transformer use ***scaled dot-product* attention**

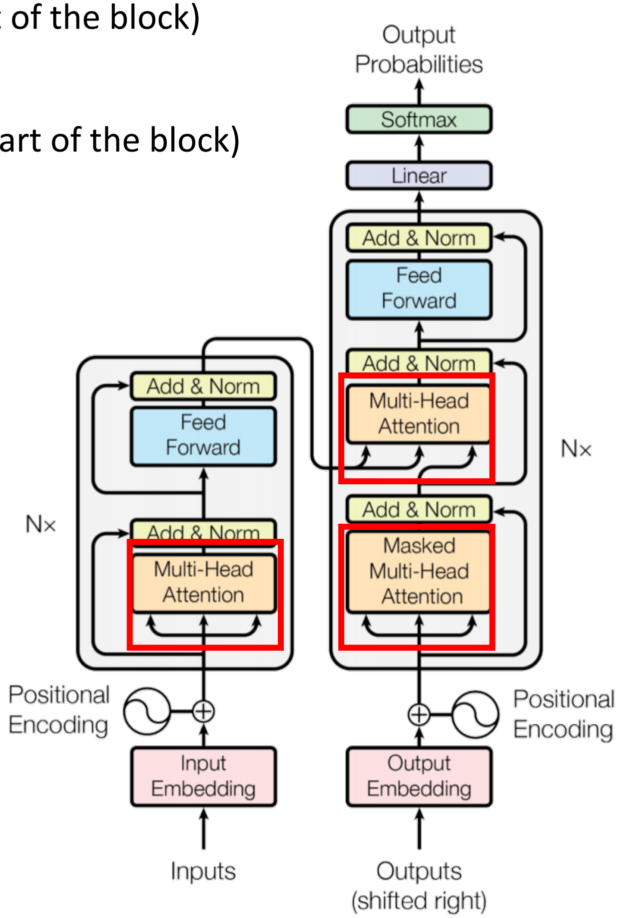
$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d}}\right) V$$

  - In addition, transformer use **multi-head attention**, ***ensemble*** of attentions



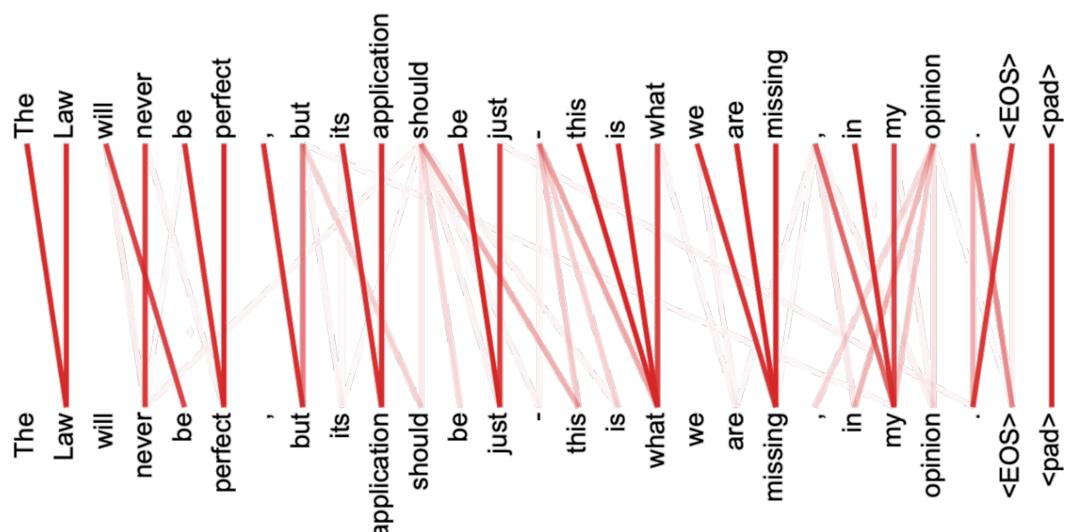
Transformer [Vaswani et al., 2017]

- Transformer:
  - The final **transformer** model is built upon the (multi-head) attention blocks
  - First, extract features with **self-attention** (see lower part of the block)
  - Then decode feature with usual **attention** (see middle part of the block)
  - Since the model don't have a sequential structure, the authors give **position embedding** (some handcrafted feature that represents the location in sequence)



- Results: Transformer architecture shows good performance for languages

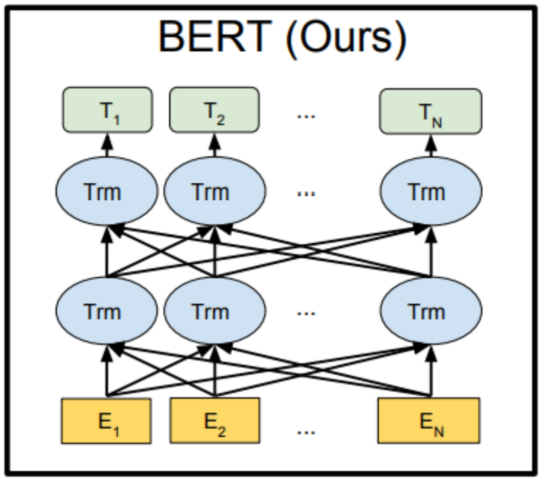
Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

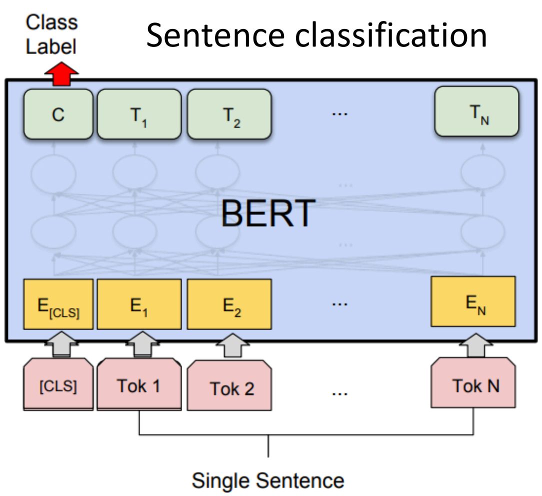


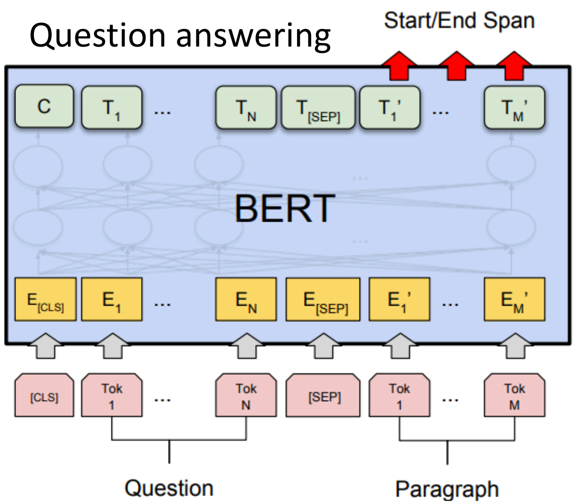
BERT [Delvin et al., 2018]

- Motivation:
  - Many **success of CNN** comes from ImageNet-pretrained networks
  - Can train a universal encoder for natural languages?

- Method:
  - **BERT (bidirectional encoder representations from transformers)**: Design a neural network based on bidirectional transformer, and use it as a pretraining model
    - Pretrain with **two tasks** (masked language model, next sentence prediction)
  - Use fixed BERT encoder, and fine-tune simple 1-layer decoder for each task







BERT [Delvin et al., 2018]

- Results:
  - Even without task-specific complex architectures, BERT achieves **SOTA** for **11 NLP tasks**, including classification, question answering, tagging, etc.

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
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	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

System	Dev F1	Test F1
ELMo+BiLSTM+CRF	95.7	92.2
CVT+Multi (Clark et al., 2018)	-	92.6
$\text{BERT}_{\text{BASE}}$	96.4	92.4
$\text{BERT}_{\text{LARGE}}$	**96.6**	**92.8**

System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
$\text{BERT}_{\text{BASE}}$	81.6	-
$\text{BERT}_{\text{LARGE}}$	**86.6**	**86.3**
Human (expert)†	-	85.0
Human (5 annotations)†	-	88.0

Table of Contents

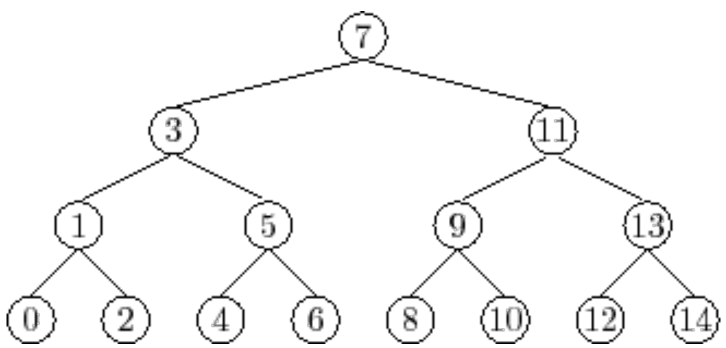
Adaptive Softmax [Grave et al., 2017]

- Motivation:
  - Computation of softmax is expensive, especially for **large vocabularies**

- Hierarchical softmax [Mnih & Hinton, 2009]:
  - Cluster k words into *balanced* $\sqrt{k}$ groups, which reduces the complexity to $O(\sqrt{k})$
  - For hidden state h, word w, and cluster $C(w)$,

$$p(w|h) = p(C(w)|h) \times p(w|C(w), h)$$

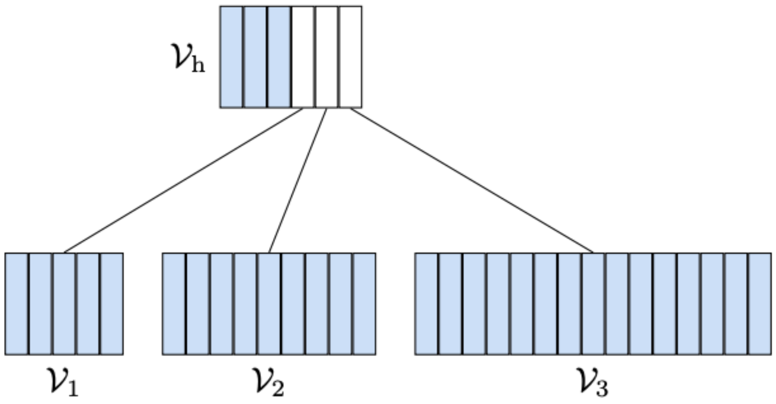
  - One can repeat clustering for subtrees (i.e., build a *balanced n-ary tree*), which reduces the complexity to $O(\log k)$



*Source: http://opendatastructures.org/versions/edition-0.1d/ods-java/node40.html

Adaptive Softmax [Grave et al., 2017]

- Limitation of prior works & Proposed idea:
  - Cluster k words into *balanced* $\sqrt{k}$ groups, which reduces the complexity to $O(\sqrt{k})$
  - One can repeat clustering for subtrees, which reduces the complexity to $O(\log k)$
  - However, **putting all words to the leaves** drop the performance (around 5-10%)
  - Instead, one can put frequent words in front (similar to Huffman coding)
  - Put **top $\boldsymbol{k_h}$** words (p_h of frequencies) and token **"NEXT-$\boldsymbol{i}$"** in the first layer, and put k_i words (p_i of frequencies) in the next layers



Adaptive Softmax [Grave et al., 2017]

- Limitation of prior works & Proposed idea:
  - Put **top $\boldsymbol{k_h}$** words (p_h of frequencies) and token **"NEXT-$\boldsymbol{i}$"** in the first layer, and put k_i words (p_i of frequencies) in the next layers
  - Let $g(k, B)$ be the computation time for k words and batch size B
  - Then the **computation time** of adaptive softmax (with J clusters) is
    - For k, B larger than some threshold, one can simply assume $g(k, B) = kB$ (see paper for details)

$$C = g(J + k_h, B) + \sum_i g(k_i, p_i B)$$
$$= (J + 1)c + \lambda B[J + k_h + \sum_i p_i k_i]$$

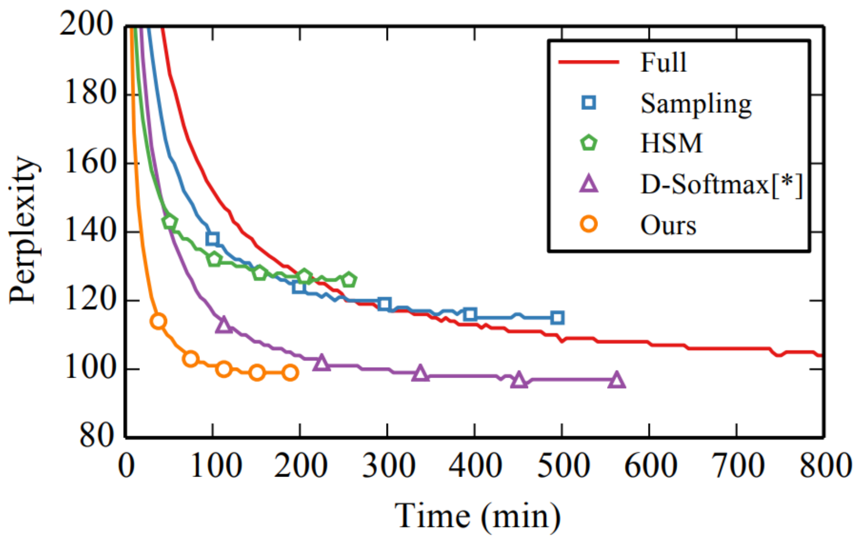
  - By solving the optimization problem (for k_i and J), the model is 3-5x faster than the original softmax (in practice, $J = 5$ works well)

Adaptive Softmax [Grave et al., 2017]

- Results: Adaptive softmax shows comparable results to the original softmax (while much faster)

ppl: perplexity (lower is better)

	ppl	training time
full softmax	144	83 min
sampling	166	41 min
HSM (freq)	166	34 min
HSM (sim)	155	41 min
D-softmax	195	53 min
D-softmax [*]	147	54 min
Ours	147	30 min

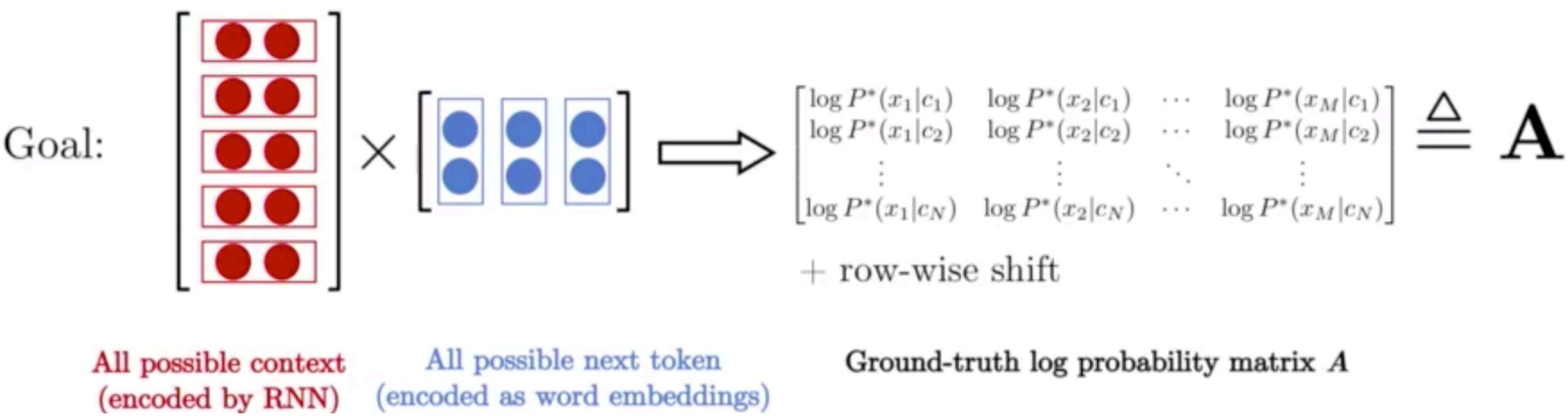


Language:	bg		cs		da		de		el		es	
k=	50k		83k		128k		143k		100k		87k	
Method	ppl	*t*	ppl	*t*	ppl	*t*	ppl	*t*	ppl	*t*	ppl	*t*
Full	37	58	62	132	37	713	42	802	38	383	30	536
Sampling	40	29	70	53	40	247	45	262	41	144	32	217
HSM (freq)	43	17	78	29	42	114	51	124	45	73	34	110
HSM (sim)	39	25	68	43	38	150	43	154	39	98	30	147
D-softmax	47	36	82	75	46	369	56	397	50	211	38	296
D-softmax [*]	37	36	62	76	36	366	41	398	37	213	29	303
Ours	37	18	62	30	35	105	40	110	36	72	29	103

Mixture of Softmax [Yang et al., 2018]

- Motivation:
  - **Rank** of softmax layer is bounded by the **feature dimension** d
  - **Recall:** By definition of softmax $P(x|c) = \frac{\exp(\mathbf{h}_c^\top \mathbf{w}_x)}{\sum_{x'} \exp(\mathbf{h}_c^\top \mathbf{w}_{x'})}$

    we have $\mathbf{h}_c^\top \mathbf{w}_x = \log P^*(x|c) + \text{const}$ (which is called *logit*)
  - Let N be number of possible contexts, and M be vocabulary size, then



which implies that softmax can represent **at most rank d** (real $\mathbf{A}$ can be larger)

*Source: https://www.facebook.com/iclr.cc/videos/2127071060655282/

Mixture of Softmax [Yang et al., 2018]

- Motivation:
  - **Rank** of softmax layer is bounded by the **feature dimension** d
  - Naïvely **increasing dimension** d to vocab size M is inefficient

- Idea:
  - Use mixture of softmaxes (MoS)

$$P^{\mathrm{MoS}}(x|c) = \sum_{k=1}^{K} \pi_{c,k} \frac{\exp(\mathbf{h}_{c,k}^\top \mathbf{w}_x)}{\sum_{x'} \exp(\mathbf{h}_{c,k}^\top \mathbf{w}_{x'})} \quad \text{s.t.} \quad \sum_{k=1}^{K} \pi_{c,k} = 1$$

  - It is easily implemented by defining $\pi_{c,k}$ and $\mathbf{h}_{c,k}$ as a function of original $\mathbf{h}_c$
  - Note that now $$\log P^{\mathrm{MoS}}(x|c) = \log \sum_{k=1}^{K} \pi_{c,k} \exp(\mathbf{h}_{c,k}^\top \mathbf{w}_x) + \mathrm{const}$$

    is a nonlinear (log-sum-exp) function of $\mathbf{h}$ and $\mathbf{w}$, hence can represent high rank

Mixture of Softmax [Yang et al., 2018]

- Results: MoS learns full rank (= vocab size) while softmax is bounded by d
  - Measured *empirical rank*, collect every empirical contexts & outputs

Model	Validation	Test
Softmax	400	400
MoC	280	280
MoS	**9981**	**9981**

MoC: mixture of contexts
(mixture *before* softmax)

$d = 400, 280, 280$ for
Softmax, MoC, MoS, respectively

#Softmax	Rank	Perplexity
3	6467	58.62
5	8930	57.36
10	9973	56.33
15	9981	55.97
20	9981	56.17

Note that 9981 is full rank
as vocab size = 9981

Mixture of Softmax [Yang et al., 2018]

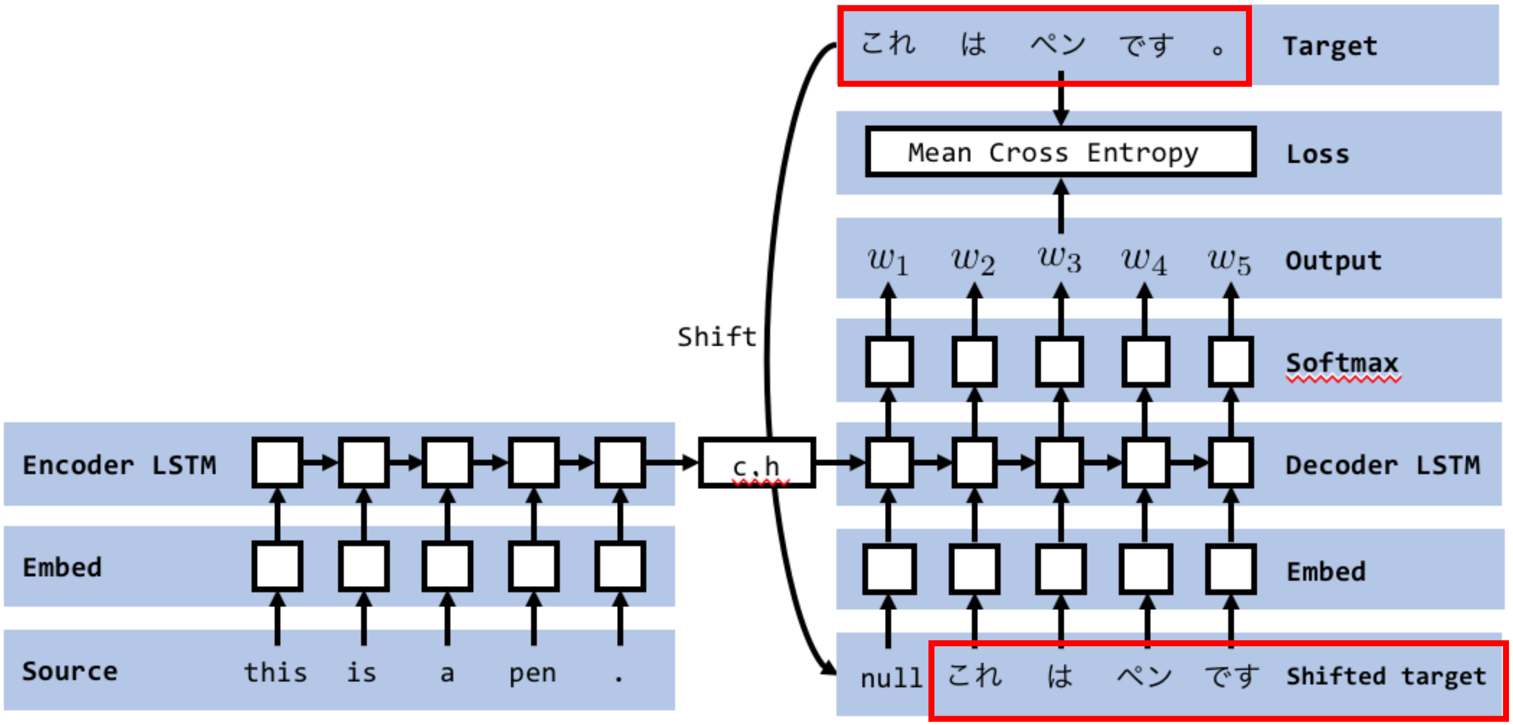
- Results: Simply changing Softmax to MoS improves the performance
  - By applying MoS to SOTA models, the authors achieved new SOTA records

Model	#Param	Validation	Test
Mikolov & Zweig (2012) – RNN-LDA + KN-5 + cache	9M‡	-	92.0
Zaremba et al. (2014) – LSTM	20M	86.2	82.7
Gal & Ghahramani (2016) – Variational LSTM (MC)	20M	-	78.6
Kim et al. (2016) – CharCNN	19M	-	78.9
Merity et al. (2016) – Pointer Sentinel-LSTM	21M	72.4	70.9
Grave et al. (2016) – LSTM + continuous cache pointer†	-	-	72.1
Inan et al. (2016) – Tied Variational LSTM + augmented loss	24M	75.7	73.2
Zilly et al. (2016) – Variational RHN	23M	67.9	65.4
Zoph & Le (2016) – NAS Cell	25M	-	64.0
Melis et al. (2017) – 2-layer skip connection LSTM	24M	60.9	58.3
Merity et al. (2017) – AWD-LSTM w/o finetune	24M	60.7	58.8
Merity et al. (2017) – AWD-LSTM	24M	60.0	57.3
Ours – AWD-LSTM-MoS w/o finetune	22M	58.08	55.97
Ours – AWD-LSTM-MoS	22M	**56.54**	**54.44**
Merity et al. (2017) – AWD-LSTM + continuous cache pointer†	24M	53.9	52.8
Krause et al. (2017) – AWD-LSTM + dynamic evaluation†	24M	51.6	51.1
Ours – AWD-LSTM-MoS + dynamic evaluation†	22M	**48.33**	**47.69**

Table of Contents

Scheduled Sampling [Bengio et al., 2015]

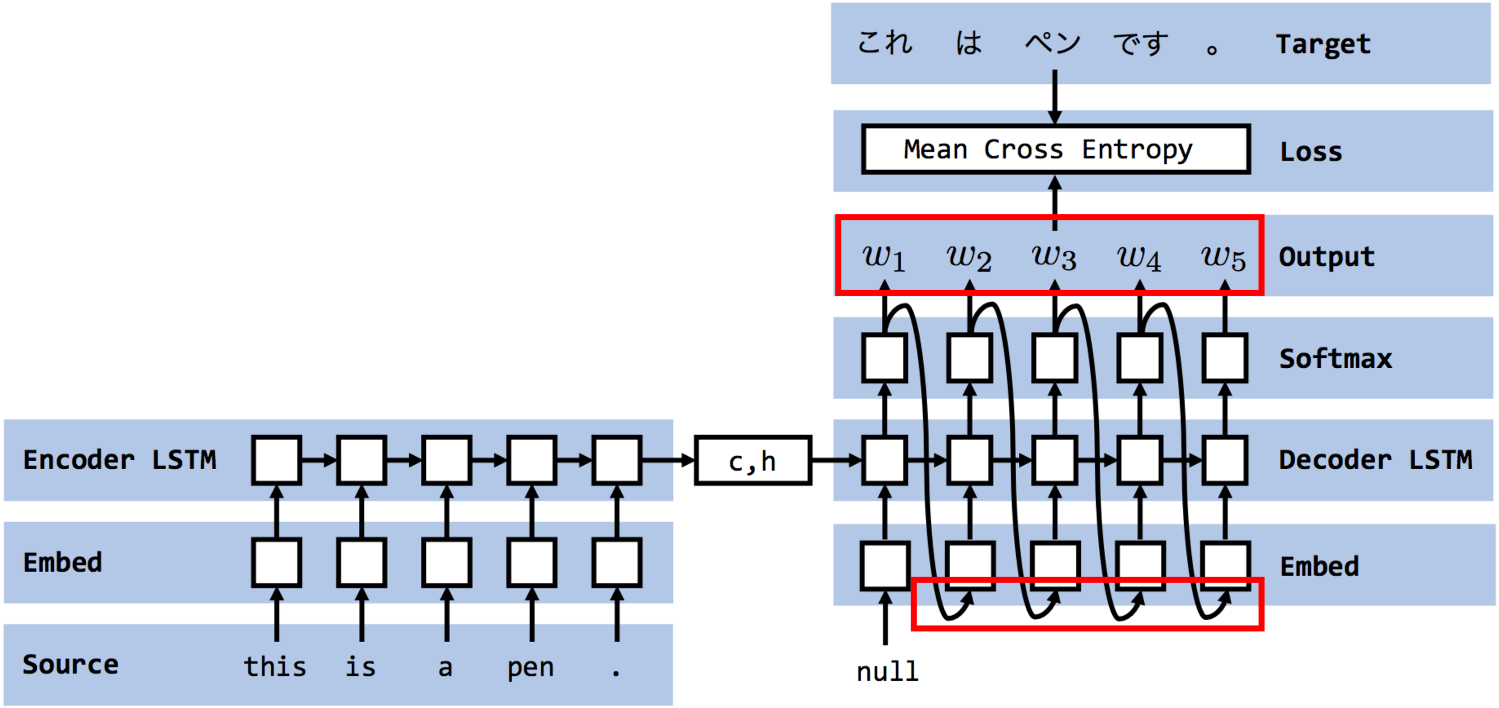
- Motivation:
  - **Teacher forcing** [Williams et al., 1989] is widely used for sequential training
  - It use real previous token and current state to predict current output



*Source: https://satopirka.com/2018/02/encoder-decoder%E3%83%A2%E3%83%87%E3%83%AB%E3%81%A8teacher-forcingscheduled-samplingprofessor-forcing/

Scheduled Sampling [Bengio et al., 2015]

- Motivation:
  - **Teacher forcing** [Williams et al., 1989] is widely used for sequential training
  - It use real previous token and current state to predict current output
  - However, the model use predicted token at inference (a.k.a. exposure bias)

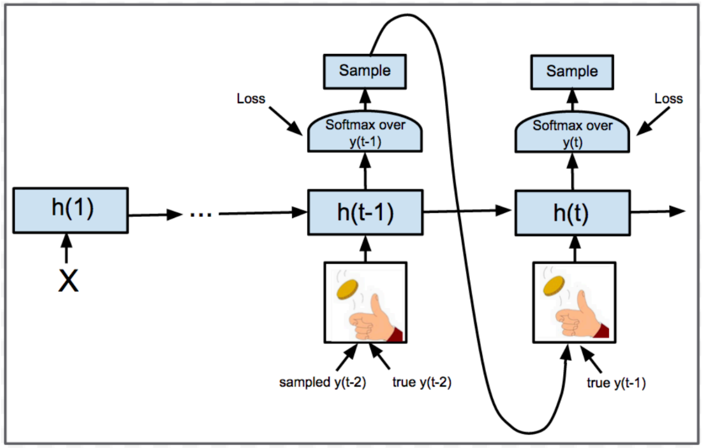


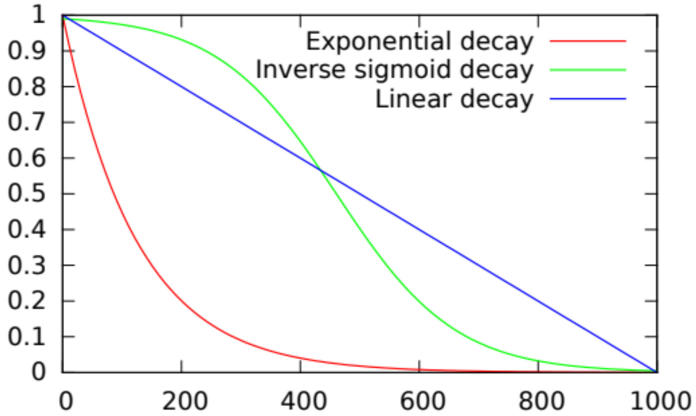
*Source: https://satopirka.com/2018/02/encoder-decoder%E3%83%A2%E3%83%87%E3%83%AB%E3%81%A8teacher-forcingscheduled-samplingprofessor-forcing/

Scheduled Sampling [Bengio et al., 2015]

- Motivation:
  - **Teacher forcing** [Williams et al., 1989] is widely used for sequential training
  - It use real previous token and current state to predict current output
  - However, the model use predicted token at inference (a.k.a. exposure bias)
  - Training with predicted token is **not trivial**, since (a) training is unstable, and (b) as previous token is changed, target also should be changed

- Idea: Apply curriculum learning
  - At beginning, use **real** tokens, and slowly move to **predicted** tokens





Scheduled Sampling [Bengio et al., 2015]

- Results: **Scheduled sampling** improves baseline for many tasks

Image captioning

Approach vs Metric	BLEU-4	METEOR	CIDER
Baseline	28.8	24.2	89.5
Baseline with Dropout	28.1	23.9	87.0
Always Sampling	11.2	15.7	49.7
Scheduled Sampling	**30.6**	**24.3**	**92.1**
Uniform Scheduled Sampling	29.2	24.2	90.9
Baseline ensemble of 10	30.7	25.1	95.7
Scheduled Sampling ensemble of 5	**32.3**	**25.4**	**98.7**

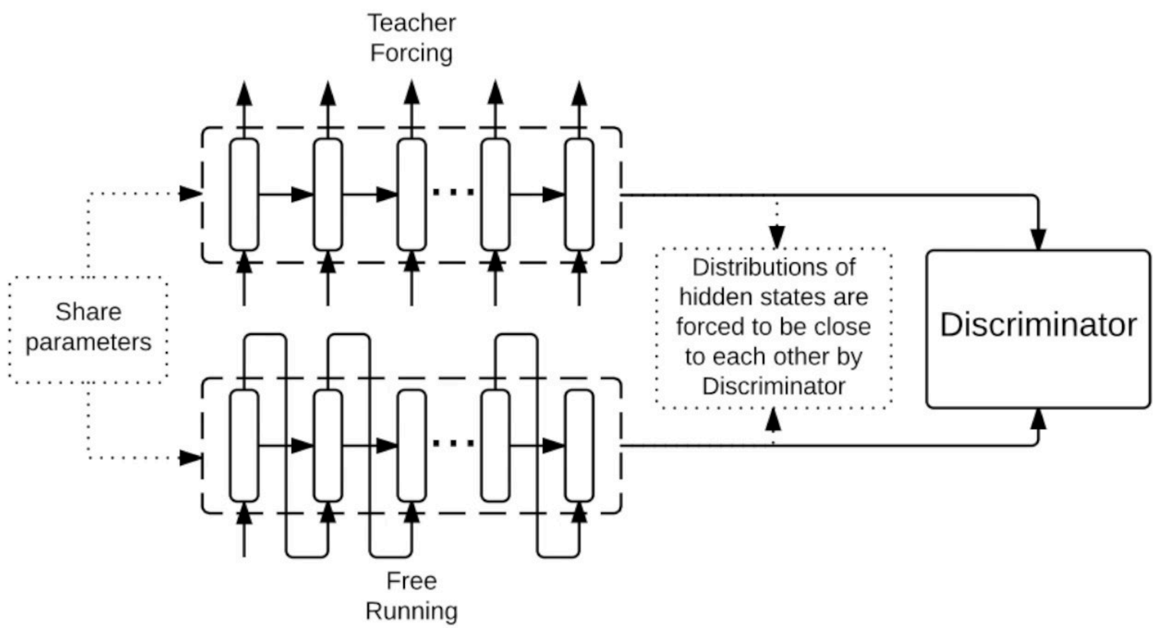
Constituency parsing

Approach	F1
Baseline LSTM	86.54
Baseline LSTM with Dropout	87.0
Always Sampling	-
Scheduled Sampling	**88.08**
Scheduled Sampling with Dropout	**88.68**

Professor Forcing [Lamb et al., 2016]

- Motivation:
  - Scheduled sampling (SS) is known to optimize wrong objective [Huszár et al., 2015]

- Idea:
  - Make features of **predicted** tokens be similar to the features of **true** tokens
  - To this end, train a discriminator classifies features of true/predicted tokens
    - **Teacher forcing:** use real tokens / **Free running:** use predicted tokens



Professor Forcing [Lamb et al., 2016]

- Results:
  - Professor forcing improves the generalization performance, especially for the long sequences (test samples are much longer than training samples)

Method	MNIST NLL
DBN 2hl (Germain *et al.*, 2015)	≈ 84.55
NADE (Larochelle and Murray, 2011)	88.33
EoNADE-5 2hl (Raiko *et al.*, 2014)	84.68
DLGM 8 leapfrog steps (Salimans *et al.*, 2014)	≈ 85.51
DARN 1hl (Gregor *et al.*, 2015)	≈ 84.13
DRAW (Gregor *et al.*, 2015)	≤ 80.97
Pixel RNN (van den Oord *et al.*, 2016)	**79.2**
Professor Forcing (ours)	79.58

NLL for MNIST generation

Response	Percent	Count
Professor Forcing Much Better	19.7	151
Professor Forcing Slightly Better	57.2	439
Teacher Forcing Slightly Better	18.9	145
Teacher Forcing Much Better	4.3	33
Total	100.0	768

Human evaluation for handwriting generation

Table of Contents

MIXER [Ranzato et al., 2016]

- Motivation:
  - Prior works use **word-level** objectives (e.g., cross-entropy) for training, but use **sequence-level** objectives (e.g., BLEU [Papineni et al., 2002]) for evaluation

- Idea: **Directly optimize** model with sequence-level objective (e.g., BLEU)
  - Q. How to backprop (usually not differentiable) sequence-level objective?

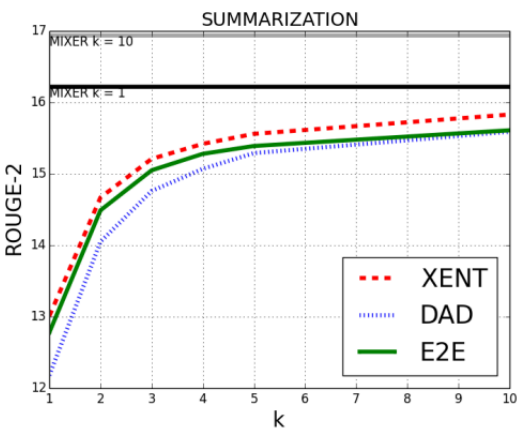
  - Sequence generation is a kind of **RL problem**
    - state: hidden state, action: output, policy: generation algorithm
  - Sequence-level objective is the **reward** of current algorithm
  - Hence, one can use policy gradient (e.g., REINFORCE) algorithm

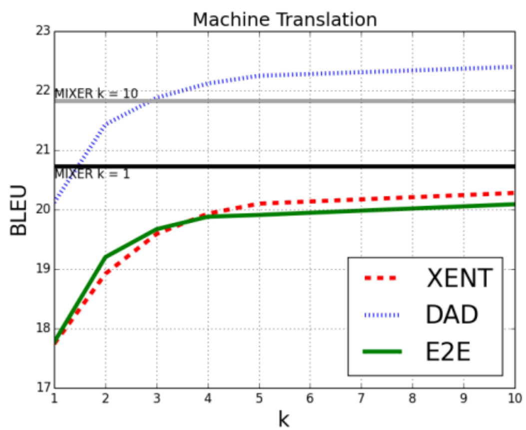
  - However, the gradient estimator of REINFORCE has **high variance**
  - To reduce variance, **MIXER (mixed incremental cross-entropy reinforce)** use MLE for first T' steps and REINFOCE for next $T - T'$ steps (T' goes to zero)
    - Cf. One can also use other variance reduction techniques, e.g., actor-critic [Bahdanau et al., 2017]

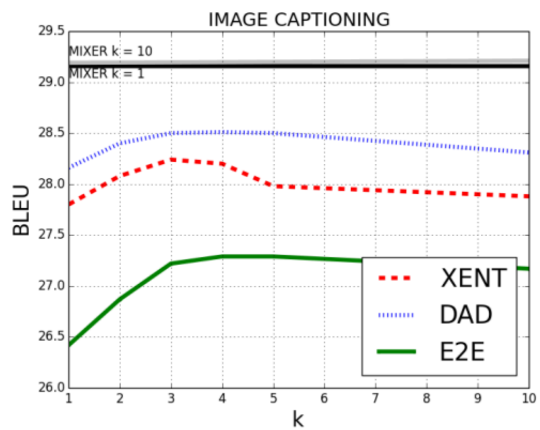
MIXER [Ranzato et al., 2016]

- Results:
  - **MIXER** shows better performance than other baselines
    - XENT (= cross entropy): another name of maximum likelihood estimation (MLE)
    - DAD (= data as demonstrator): another name of scheduled sampling
    - E2D (= end-to-end backprop): use top-K vector as input (approx. beam search)

TASK	XENT	DAD	E2E	MIXER
summarization	13.01	12.18	12.78	**16.22**
translation	17.74	20.12	17.77	**20.73**
image captioning	27.8	28.16	26.42	**29.16**



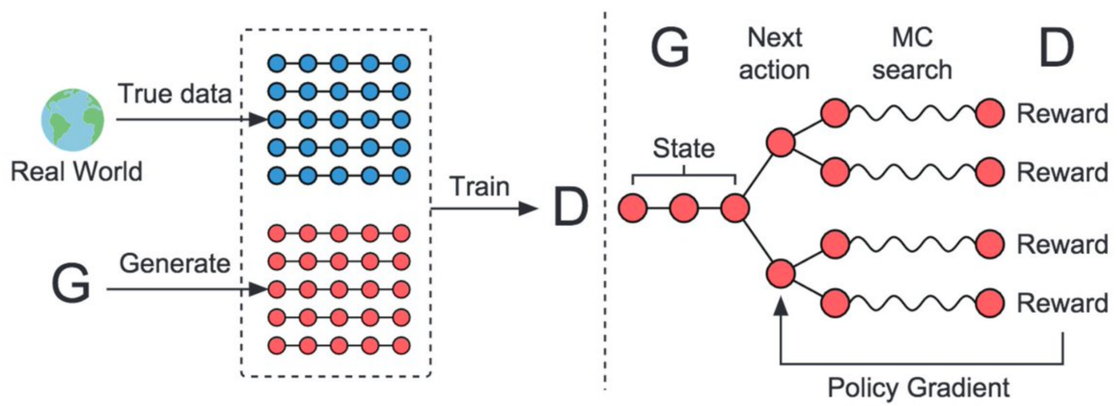




SeqGAN [Yu et al., 2017]

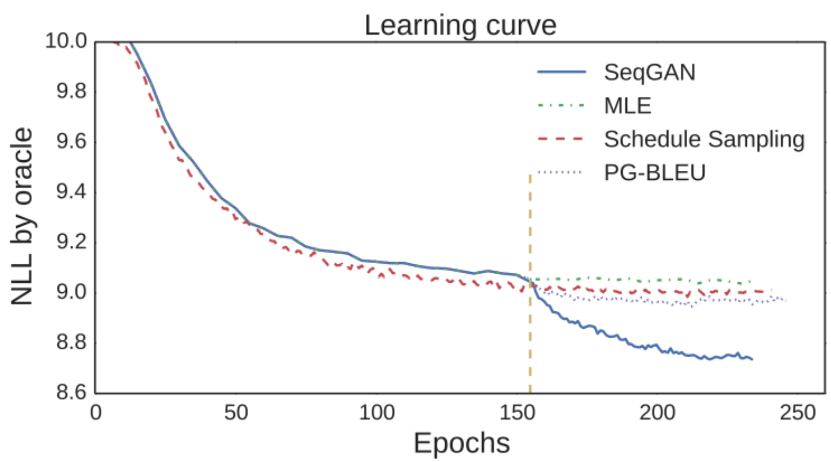
- Motivation:
  - RL-based method still relies on handcrafted objective (e.g., BLEU)
  - Instead, one can use GAN loss to generate realistic sequences
  - However, it is not trivial to apply GAN for natural languages, since data is **discrete** (hence not differentiable) and **sequence** (hence need new architecture)

- Idea: Backprop discriminator's output with policy gradient
  - Similar to **actor-critic**; only difference is now the reward is discriminator's output
  - Use LSTM-generator and CNN (or Bi-LSTM)-discriminator architectures



- Results:
  - **SeqGAN** shows better performance than prior methods

Algorithm	Random	MLE	SS	PG-BLEU	SeqGAN
NLL	10.310	9.038	8.985	8.946	**8.736**
p-value	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	$< 10^{-6}$	



Synthetic generation (follow the oracle)

Chinese poem generation

Algorithm	Human score	p-value	BLEU-2	p-value
MLE	0.4165	0.0034	0.6670	$< 10^{-6}$
SeqGAN	**0.5356**		**0.7389**	
Real data	0.6011		0.746	

Obama speech generation

Algorithm	BLEU-3	p-value	BLEU-4	p-value
MLE	0.519	$< 10^{-6}$	0.416	0.00014
SeqGAN	**0.556**		**0.427**	

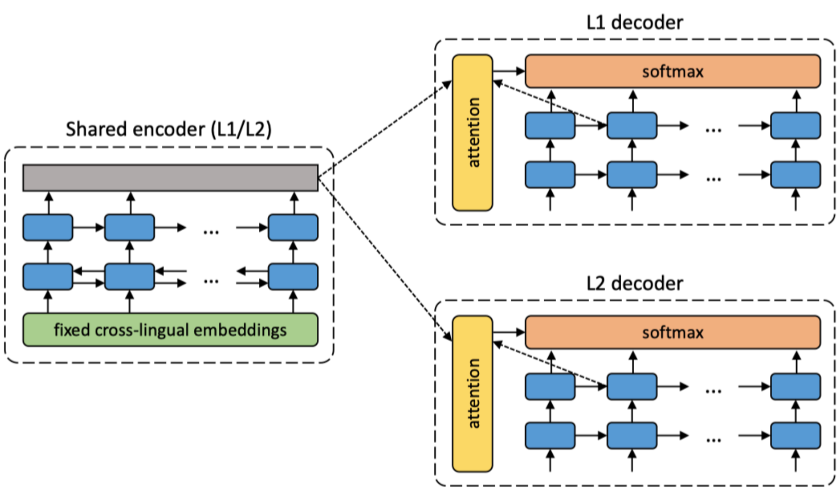
Table of Contents

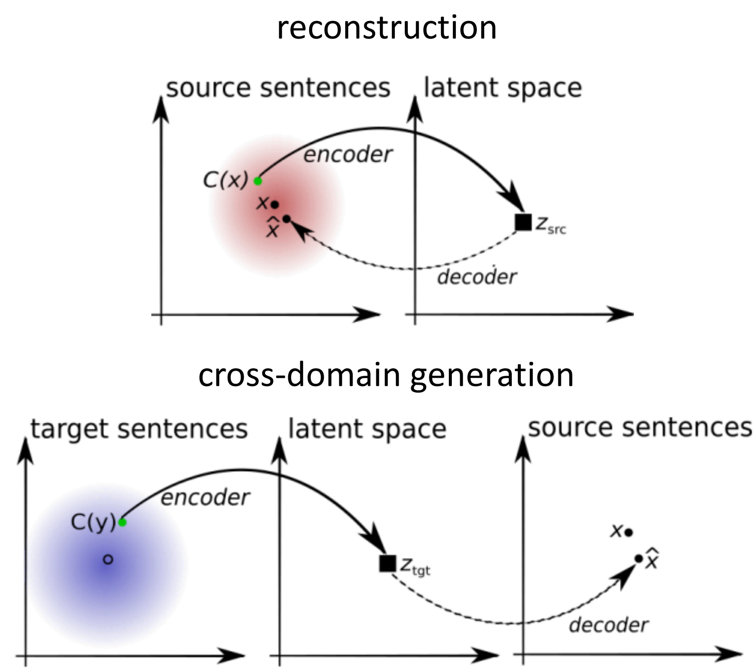
UNMT [Artetxe et al., 2018]

- Motivation:
  - Can train **neural machine translation** models in unsupervised way?

- Idea: Apply the idea of domain transfer in Lecture 12
  - Combine **two losses**: reconstruction loss and cycle-consistency loss
    - **Recall:** Cycle-consistency loss forces *twice* cross-domain generated (e.g., L1→L2→L1) data to become the original data

Model architecture (L1/L2: language 1, 2)





*Source: Lample et al. "Unsupervised Machine Translation Using Monolingual Corpora Only", ICLR 2018.

UNMT [Artetxe et al., 2018]

- Results: UNMT produces good translation results

BPE (byte pair encoding), a preprocessing method

		FR-EN	EN-FR	DE-EN	EN-DE
Unsupervised	1. Baseline (emb. nearest neighbor)	9.98	6.25	7.07	4.39
	2. Proposed (denoising)	7.28	5.33	3.64	2.40
	3. Proposed (+ backtranslation)	15.56	15.13	10.21	6.55
	4. Proposed (+ BPE)	15.56	14.36	10.16	6.89
Semi-supervised	5. Proposed (full) + 100k parallel	21.81	21.74	15.24	10.95
Supervised	6. Comparable NMT	20.48	19.89	15.04	11.05
	7. GNMT (Wu et al., 2016)	-	38.95	-	24.61

Source	Reference	Proposed system (full)
Une fusillade a eu lieu à l'aéroport international de Los Angeles.	There was a shooting in Los Angeles International Airport.	A shooting occurred at Los Angeles International Airport.
Cette controverse croissante autour de l'agence a provoqué beaucoup de spéculations selon lesquelles l'incident de ce soir était le résultat d'une cyber-opération ciblée.	Such growing controversy surrounding the agency prompted early speculation that tonight's incident was the result of a targeted cyber operation.	This growing scandal around the agency has caused much speculation about how this incident was the outcome of a targeted cyber operation.
Le nombre total de morts en octobre est le plus élevé depuis avril 2008, quand 1 073 personnes avaient été tuées.	The total number of deaths in October is the highest since April 2008, when 1,073 people were killed.	The total number of deaths in May is the highest since April 2008, when 1 064 people had been killed.
À l'exception de l'opéra, la province reste le parent pauvre de la culture en France.	With the exception of opera, the provinces remain the poor relative of culture in France.	At an exception, opera remains of the state remains the poorest parent culture.

Conclusion

- **Deep learning is widely used for natural language processing (NLP)**
  - RNN and CNN were popular in 2014-2017
  - Recently, self-attention based methods are widely used

- **Many new ideas are proposed to solve language problems**
  - New architectures (e.g., self-attention, softmax)
  - New training methods (e.g., loss, algorithm, unsupervised)

- **Research for natural languages are now just began**
  - Deep learning (especially GAN) is not widely used in NLP as computer vision
  - Transformer and BERT are just published in 2017-2018
  - There are still many research opportunities in NLP

References

Introduction

- [Papineni et al., 2002] BLEU: a method for automatic evaluation of machine translation. ACL 2002.
  link : https://dl.acm.org/citation.cfm?id=1073135
- [Cho et al., 2014] Learning Phrase Representations using RNN Encoder-Decoder for Statistical... EMNLP 2014.
  link : https://arxiv.org/abs/1406.1078
- [Sutskever et al., 2014] Sequence to Sequence Learning with Neural Networks. NIPS 2014.
  link : https://arxiv.org/abs/1409.3215
- [Gehring et al., 2017] Convolutional Sequence to Sequence Learning. ICML 2017.
  link : https://arxiv.org/abs/1705.03122
- [Young et al., 2017] Recent Trends in Deep Learning Based Natural Language Processing. arXiv 2017.
  link : https://arxiv.org/abs/1708.02709

Extension to unsupervised setting

- [Artetxe et al., 2018] Unsupervised Neural Machine Translation. ICLR 2018.
  link : https://arxiv.org/abs/1710.11041
- [Lample et al., 2018] Unsupervised Machine Translation Using Monolingual Corpora Only. ICLR 2018.
  link : https://arxiv.org/abs/1711.00043

References

Learning long-term dependencies

- [Bahdanau et al., 2015] Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015.
  link : https://arxiv.org/abs/1409.0473
- [Weston et al., 2015] Memory Networks. ICLR 2015.
  link : https://arxiv.org/abs/1410.3916
- [Xu et al., 2015] Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. ICML 2015.
  link : https://arxiv.org/abs/1502.03044
- [Sukhbaatar et al., 2015] End-To-End Memory Networks. NIPS 2015.
  link : https://arxiv.org/abs/1503.08895
- [Kumar et al., 2016] Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. ICML 2016.
  link : https://arxiv.org/abs/1506.07285
- [Vaswani et al., 2017] Attention Is All You Need. NIPS 2017.
  link : https://arxiv.org/abs/1706.03762
- [Wang et al., 2018] Non-local Neural Networks. CVPR 2018.
  link : https://arxiv.org/abs/1711.07971
- [Zhang et al., 2018] Self-Attention Generative Adversarial Networks. arXiv 2018.
  link : https://arxiv.org/abs/1805.08318
- [Peters et al., 2018] Deep contextualized word representations. NAACL 2018.
  link : https://arxiv.org/abs/1802.05365
- [Delvin et al., 2018] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv 2018.
  link : https://arxiv.org/abs/1810.04805

References

Improve softmax layers

- [Mnih & Hinton, 2009] A Scalable Hierarchical Distributed Language Model. NIPS 2009.
  link : https://papers.nips.cc/paper/3583-a-scalable-hierarchical-distributed-language-model
- [Grave et al., 2017] Efficient softmax approximation for GPUs. ICML 2017.
  link : https://arxiv.org/abs/1609.04309
- [Yang et al., 2018] Breaking the Softmax Bottleneck: A High-Rank RNN Language Model. ICLR 2018.
  link : https://arxiv.org/abs/1711.03953

Reduce exposure bias

- [Williams et al., 1989] A Learning Algorithm for Continually Running Fully Recurrent... Neural Computation 1989.
  link : https://ieeexplore.ieee.org/document/6795228
- [Bengio et al., 2015] Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. NIPS 2015.
  link : https://arxiv.org/abs/1506.03099
- [Huszár et al., 2015] How (not) to Train your Generative Model: Scheduled Sampling, Likelihood... arXiv 2015.
  link : https://arxiv.org/abs/1511.05101
- [Lamb et al., 2016] Professor Forcing: A New Algorithm for Training Recurrent Networks. NIPS 2016.
  link : https://arxiv.org/abs/1610.09038

References

Reduce loss/evaluation mismatch

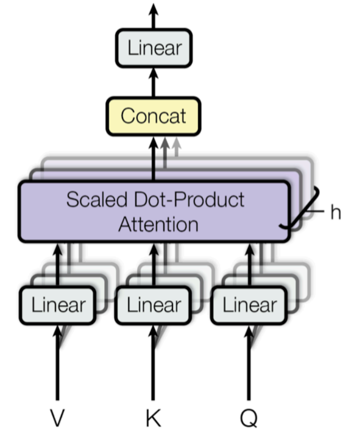
- [Ranzato et al., 2016] Sequence Level Training with Recurrent Neural Networks. ICLR 2016.
  link : https://arxiv.org/abs/1511.06732
- [Bahdanau et al., 2017] An Actor-Critic Algorithm for Sequence Prediction. ICLR 2017.
  link : https://arxiv.org/abs/1607.07086
- [Yu et al., 2017] SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. AAAI 2017.
  link : https://arxiv.org/abs/1609.05473
- [Rajeswar et al., 2017] Adversarial Generation of Natural Language. arXiv 2017.
  link : https://arxiv.org/abs/1705.10929
- [Maddison et al., 2017] The Concrete Distribution: A Continuous Relaxation of Discrete Random... ICLR 2017.
  link : https://arxiv.org/abs/1611.00712
- [Jang et al., 2017] Categorical Reparameterization with Gumbel-Softmax. ICLR 2017.
  link : https://arxiv.org/abs/1611.01144
- [Kusner et al., 2016] GANS for Sequences of Discrete Elements with the Gumbel-softmax... NIPS Workshop 2016.
  link : https://arxiv.org/abs/1611.04051
- [Tucker et al., 2017] REBAR: Low-variance, unbiased gradient estimates for discrete latent variable... NIPS 2017.
  link : https://arxiv.org/abs/1703.07370
- [Hjelm et al., 2018] Boundary-Seeking Generative Adversarial Networks. ICLR 2018.
  link : https://arxiv.org/abs/1702.08431
- [Zhao et al., 2018] Adversarially Regularized Autoencoders. ICML 2018.
  link : https://arxiv.org/abs/1706.04223

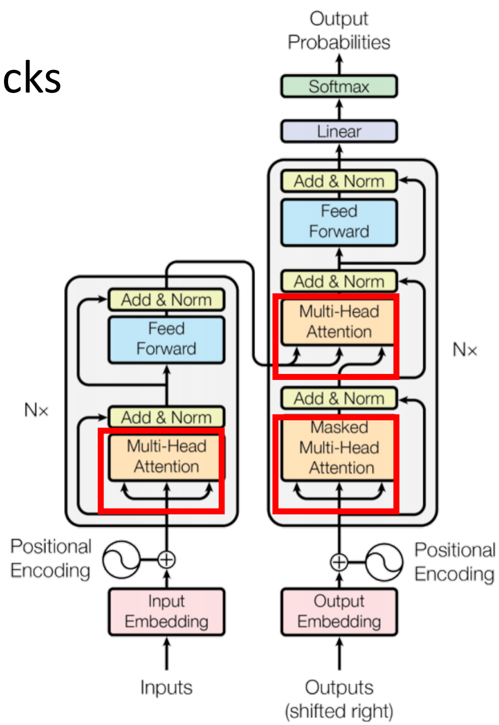
Transformer [Vaswani et al., 2017]

- Method:
  - (Scaled dot-product) **attention** is given by

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{n}}\right)\mathbf{V}$$

  - Use **multi-head attention** (i.e., ensemble of attentions)
  - The final **transformer** model is built upon the attention blocks
    - First, extract features with **self-attention**
    - Then decode feature with usual **attention**
    - Since the model don't have a sequential structure, the authors give **position embedding** (some handcrafted feature that represents the location in sequence)





*Notation: $(\mathbf{K},\mathbf{V})$ is (key, value) pair, and $\mathbf{Q}$ is query

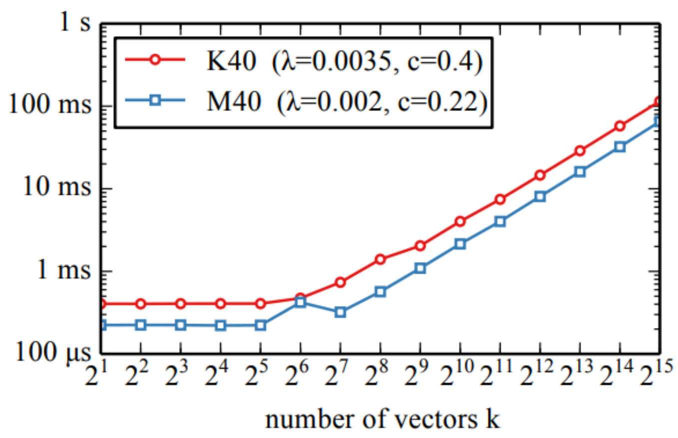
Adaptive Softmax [Grave et al., 2017]

- Limitation of prior works & Proposed idea:
  - Put **top $\boldsymbol{k_h}$** words (p_h of frequencies) and a token **"NEXT"** in the first layer, and put $k_t = k - k_h$ words ($p_t = 1 - p_h$ of frequencies) in the next layer
  - Let $g(k, B)$ be a computation time for k vocabularies and batch size B
  - Then the **computation time** of the proposed method is

$$C = g(k_h + 1, B) + g(k_t, p_t B)$$

  - Here, $g(k, B)$ is a **threshold function** (due to the initial setup of GPU)

$$g(k, B) = \max(c + \lambda k_0 B_0, c + \lambda k B)$$



Adaptive Softmax [Grave et al., 2017]

- Limitation of prior works & Proposed idea:
  - The **computation time** of the proposed method is

$$C = g(k_h + 1, B) + g(k_t, p_t B)$$
$$g(k, B) = \max(c + \lambda k_0 B_0, c + \lambda k B)$$

  - Hence, give a **constraint** that $kB \geq k_0 B_0$ (for efficient usage for GPU)

  - Also, extend the model to **multi-cluster** setting (with J clusters):

$$C = g(J + k_h, B) + \sum_i g(k_i, p_i B)$$
$$= (J + 1)c + \lambda B[J + k_h + \sum_i p_i k_i]$$

  - By solving the optimization problem (for k_i and J), the model is 3-5x faster than the original softmax (in practice, $J = 5$ shows good computation/performance trade-off)

Professor Forcing [Lamb et al., 2016]

- Motivation:
  - Scheduled sampling (SS) is known to optimize wrong objective [Huszár et al., 2015]

  - Let P and Q be data and model distribution, respectively
  - Assume length 2 sequence $x_1 x_2$, and let ϵ be the ratio of real sample

  - Then the **objective** of scheduled sampling is

$$D_{SS}[P\|Q] = KL[P_{x_1}\|Q_{x_1}]$$

$$+(1-\epsilon)\mathbb{E}_{z\sim Q_{x_1}} KL[P_{x_2}\|Q_{x_2|x_1=z}] + \epsilon KL[P_{x_2|x_1}\|Q_{x_2|x_1}]$$

  - If $\epsilon = 1$, it is usual MLE objective, but as $\epsilon \to 0$, it pushes the conditional distribution $Q_{x_2|x_1}$ to the marginal distribution P_{x_2} instead of $P_{x_2|x_1}$
  - Hence, the factorized $Q^* = P_{x_1} P_{x_2}$ can minimize the objective

More Methods for Discrete GAN

- Gumbel-Softmax (a.k.a. concrete distribution):
  - Gradient estimator of REINFORCE has high variance
  - One can apply reparameterization trick… but how for **discrete** variables?

  - One can use **Gumbel-softmax trick** [Jang et al., 2017]; [Maddison et al., 2017] to achieve a *biased but low variance* gradient estimator

  - One can also get *unbiased* estimator using Gumbel-softmax estimator as a control variate for REINFORCE, called **REBAR** [Tucker et al., 2017]

- Discrete GAN is still an active research area
  - BSGAN [Hjelm et al., 2018], ARAE [Zhao et al., 2018], etc.
  - However, GAN is not popular for sequences (natural languages) as images yet