CS 533: Natural Language Processing

Conditional Language Models, Machine Translation

Karl Stratos

Rutgers University

Review: Language Models (LMs)

- Defines the probability of a sentence as a product of word probabilities conditioning on history (chain rule):

$$\begin{aligned} p_\theta(x_1 \dots x_T) = & \; p_\theta(x_1 | \langle \text{bos} \rangle) \\ & \times p_\theta(x_2 | \langle \text{bos} \rangle, x_1) \\ & \times p_\theta(x_3 | \langle \text{bos} \rangle, x_1, x_2) \times \cdots \\ & \times p_\theta(x_T | \langle \text{bos} \rangle, x_1, x_2, \dots, x_{T-1}) \\ & \times p_\theta(\langle \text{eos} \rangle | \langle \text{bos} \rangle, x_1, x_2, \dots, x_{T-1}, x_T) \end{aligned}$$

- Unsupervised and data-efficient: Each unlabeled sentence $x \sim \mathbf{pop}$ provides $T+1$ labeled examples

$$(\text{inputs}, \text{targets}) = ((x_0, x_{0:1}, \dots, x_{0:T}), (x_1, x_2, \dots, x_{T+1}))$$

- Trained by minimizing the empirical cross-entropy loss

$$\widehat{J}(\theta) = -\frac{1}{N} \sum_{i=1}^{M} \sum_{t=1}^{T_i+1} \log p_\theta(x_t^{(i)} | x_{<t}^{(i)})$$

Review: n-Gram vs Recurrent LMs

- n-gram LM: $\mathbf{pop}(x_t|x_{<t}) \approx p_\theta(x_t|x_{t-K} \dots x_{t-1})$
  - **Markov assumption.** Next word only depends on past K words
  - Amounts to classifying an n-gram into a word: feedforward (Bengio et al., 2003), transformer (e.g., GPT)
- Recurrent LM
  - No Markov assumption, maintain a recurrent state h_{t-1} that encodes all history (e.g., LSTM hidden state, transformer output on entire history)
  - In practice trained by backpropagation through time
- **Perplexity**: $\exp(H(\mathbf{pop}(x_t|x_{<t}), p_\theta(x_t|x_{<t}))$
  - Branching factor/effective vocab size (V when random)
  - Most words predictable from short history, being recurrent may not yield visibly lower perplexity
  - But *truly* minimizing perplexity (against diminishing returns) will imply real language understanding

Masked Attention in Transformer LM

- Big advantage of self-attention over recurrent: computation over a sequence can be done in "one shot" during training
- Recurrent LM: 3 steps (cannot be parallelized)

$$\langle \text{bos} \rangle \; x_1 \; x_2 \; x_3 \quad \Rightarrow \quad \langle \text{bos} \rangle \; x_1 \; x_2 \; x_3 \quad \Rightarrow \quad \langle \text{bos} \rangle \; x_1 \; x_2 \; x_3$$

- Transformer LM: 1 step

$$\langle \text{bos} \rangle \; x_1 \; x_2 \; x_3$$

How? Simulate left-to-right during self-attention computation by masking future words!

$$\begin{bmatrix} l_{00} & l_{01} & l_{02} \\ l_{10} & l_{11} & l_{12} \\ l_{20} & l_{21} & l_{22} \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} l_{00} & -\infty & -\infty \\ l_{10} & l_{11} & -\infty \\ l_{20} & l_{21} & l_{22} \end{bmatrix}$$

- Masking only done for training, test time is actually left-to-right predictions.

Review: Search and Generation

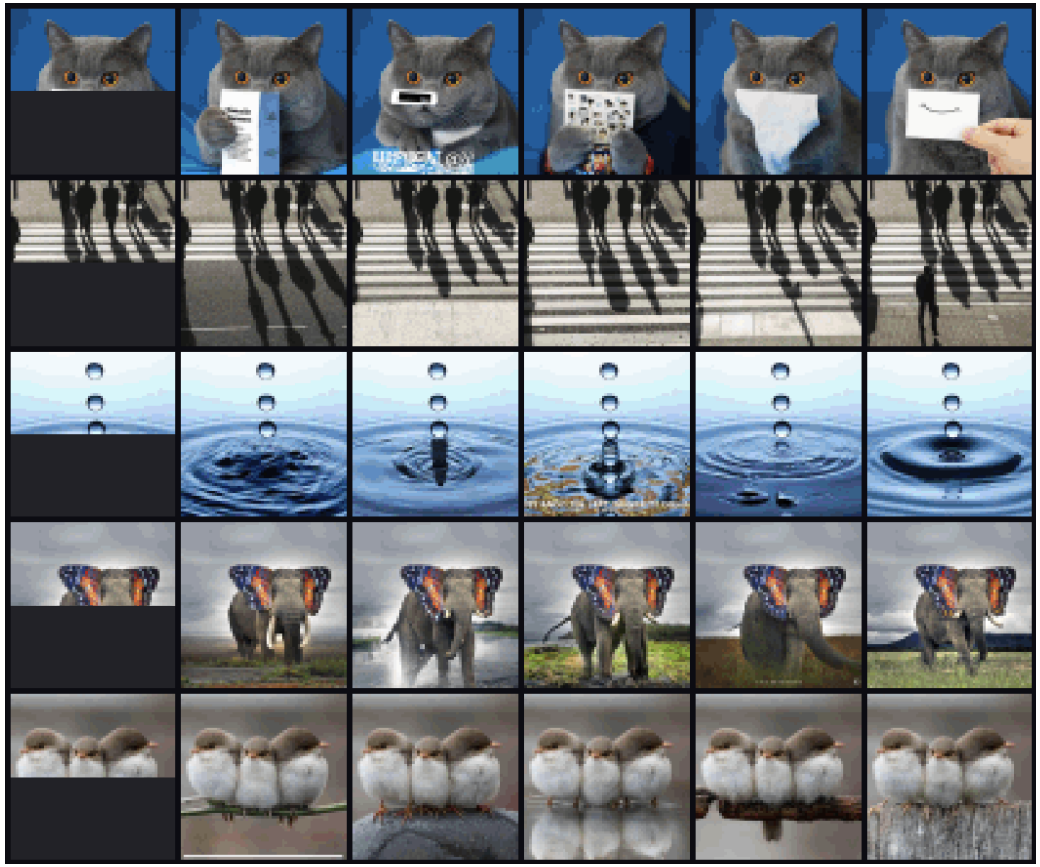
- Most likely sentence under the model

$$x^* = \underset{x_1 \dots x_T \in \mathcal{V}^T,\, T \in \{1 \dots T_{\max}\}}{\arg\max} \sum_{t=1}^{T+1} \log p_\theta(x_t | x_{<t})$$

- Generally intractable: Must search through $O(V^{T_{\max}})$ possible sequences
- **Beam search.** Approximate x^* by keeping only β best hypotheses at each time step
  - Each step considers $V\beta$ branches, runtime $O(\text{top}_\beta(\beta V) T_{\max})$
  - $\beta = 1$ greedy decoding, $\beta = V^{T_{\max}}$ exact search
- Generation: Sample sentence $x \sim p_\theta$ stepwise, lots of tricks to calibrate randomness (softmax temperature, top-k/top-p sampling, beam search variants)

The Generality of "Language Models"

Generative Pretraining from Pixels (Chen et al., 2020)



Contextual generation to complete half-images

- "Vocabulary": pixel values discretized by k-means clustering

Conditional Language Models

- Conditional LM: Given a sequence $x = (x_1 \dots x_T)$, defines

$$p_\theta(y_1 \dots y_{T'} | x) = \prod_{t=1}^{T'+1} p_\theta(y_t | x, y_{<t})$$

- Everything is the same as LM except the conditioning part.
- Some context-specific jargons
  - x **source** sequence, $y = (y_1 \dots y_{T'})$ **target** sequence
  - **Encoder.** First, $\textbf{enc}_\theta : \mathcal{V}^T \to \mathbb{R}^D$ encodes all tokens in x into a differentiable representation (details later)
  - **Decoder.** In each step, $\textbf{dec}_\theta : \mathbb{R}^D \times \mathcal{V}^{t-1} \to \mathbb{R}^V$ predicts next word from the encoding and history by

$$p_\theta(y_t | x, y_{<t}) = \text{softmax}_{y_t}(\textbf{dec}_\theta(\textbf{enc}_\theta(x), y_{<t}))$$

- **Encoder-decoder** or **sequence-to-sequence** (seq2seq) framework: Training encoder & decoder jointly to optimize a function of $p_\theta(y|x)$

A Universal Framework for Supervised Learning

Machine translation

And the programme has been implemented ⇒ Le programme a été mis en application

Summarization

russian defense minister ivanov called sunday for the creation of a joint front for combating global terrorism ⇒ russia calls for joint front against terrorism

Data-to-text generation

TEAM	WIN	LOSS	PTS	FGPCT	RB	AS
Heat	11	12	103	49	47	27
Hawk	7	15	95	43	33	20

⇒ The Atlanta Hawks defeated the Miami Heat, 103-95, at Philips Arena on Wednesday. . .

Image captioning

⇒ the dog saw the cat

Dialogue/chatbot

If you could have dinner with one person, dead or alive, who would that be? That's a tough one. I think I'd have to go with Steve Jobs. I'd love to pick his brain. Ah, interesting choice. What do you know about Steve Jobs? He was one of the most influential people in the history of technology. He was a visionary. What questions would you ask him? ⇒ I'd want to know how he came up with the idea for the iPod. It was so revolutionary at the time.

LM vs Seq2Seq

- LM can also condition on the past (e.g., contextual text generation)
  - Sample from $p_\theta(\cdot|x_1 \dots x_J)$ where $x_1 \dots x_J$ is a prompt.
  - Amounts to running LM on $x_1 \dots x_J$ first and generating from final state.
- What benefits from having an explicit source encoder? More flexibility
  - In LM, conditioning object is treated the same as generation object and must be processed left-to-right (e.g., x_j cannot attend to $x_{>j}$)
  - In seq2seq, no longer have that constraint (e.g., can do full self-attention over $x_1 \dots x_J$)
  - More generally, conditioning object may not be a sequence (e.g., if image, use ResNet as encoder)
- Think of LM as a special case of seq2seq with constant source

Encoder Architecture: Single Embedding

- Source encoder: $\textbf{enc}_\theta : \mathcal{V}^T \rightarrow \mathbb{R}^D$
- Original seq2seq paper: Sequence to Sequence Learning with Neural Networks (Sutskever et al., 2014)
  - Encoded T source tokens into a vector

$$\textbf{enc}_\theta(x_1 \ldots x_T) = h_T \in \mathbb{R}^d$$

    where h_T is the last hidden state of a left-to-right LSTM (later studies found last hidden state of a right-to-left LSTM to work better)
- Kind of works, but not really well
  - Too much happening in an entire sequence to capture with a *single* vector

You can't cram the meaning of a whole sentence into a single vector!
–Ray Mooney

Encoder Architecture: Contextual Embeddings

- Instead of a single vector, encode T source tokens into T contextual embeddings

$$\textbf{enc}_\theta(x_1 \dots x_T) = (h_1 \dots h_T) \in \mathbb{R}^{T \times d}$$

  where $h_1 \dots h_T$ can be defined in various ways
  - Left-to-right/right-to-left/bidirectional RNN embeddings
  - Convolutional neural network embeddings
  - Transformer encoder embeddings
- Idea: Let the decoder work with individual tokens of the source via **attention**
  - This is the origin of the attention mechanism, proposed in: Neural Machine Translation by Jointly Learning to Align and Translate (Bahdanau et al., 2016)
  - We will assume contextual embeddings for source representation

Decoder Architecture

- Stepwise target decoder $\mathbf{dec}_\theta : \mathbb{R}^{T \times d} \times \mathcal{V}^{t-1} \rightarrow \mathbb{R}^V$
  - At step t, maps contextual embeddings of $x_1 \dots x_T$ and target history $y_1 \dots y_{t-1}$ to logits over $\mathcal{V}$
- Recall attention: Three types of vector
  - **Query vector**: $q \in \mathbb{R}^d$
  - **Key/value vectors**: $(k_1, v_1) \dots (k_T, v_T) \in \mathbb{R}^d \times \mathbb{R}^d$
- Key/value collectively called **memory bank** M
- Compute an embedding from M "attended" by q

$$(p_1 \dots p_T) = \text{softmax}(q^\top k_1, \dots, q^\top k_T)$$

$$\mathbf{Attn}(q, M) := \sum_{j=1}^{T} p_j v_j$$

- Here, q is an encoding of $y_1 \dots y_{t-1}$; (k_j, v_j) is an embedding of the j-th source token

Example: LSTM Decoder With Input Feeding (Luong et al., 2015)

- Source encodings $h = (h_1 \dots h_T) \in \mathbb{R}^{T \times d}$
- Memory bank: $M = \{(k_j, v_j)\}_{j=1}^{T}$ where $k_j = v_j = h_j$ (tying key and value vectors)
- Step $t = 1$: Assumes $y_0 = \langle \text{bos} \rangle$ given

$$q_0 = \textbf{LSTM}_\theta(0_{2d}, [E(y_0); 0_d]))$$
$$z_0 = \textbf{Attn}(q_0, M)$$
$$h'_0 = \tanh(W[z_0; q_0]) \qquad \textbf{dec}_\theta(h, y_0) = E^\top h'_0$$

Example: LSTM Decoder With Input Feeding (Luong et al., 2015)

- Source encodings $h = (h_1 \dots h_T) \in \mathbb{R}^{T \times d}$
- Memory bank: $M = \{(k_j, v_j)\}_{j=1}^{T}$ where $k_j = v_j = h_j$ (tying key and value vectors)
- Step $t = 1$: Assumes $y_0 = \langle \text{bos} \rangle$ given

$$q_0 = \textbf{LSTM}_\theta(0_{2d}, [E(y_0); 0_d]))$$
$$z_0 = \textbf{Attn}(q_0, M)$$
$$h'_0 = \tanh(W[z_0; q_0]) \qquad \textbf{dec}_\theta(h, y_0) = E^\top h'_0$$

- Step $t = 2$: Assumes y_0, y_1 given

$$q_1 = \textbf{LSTM}_\theta(q_0, [E(y_1); h'_0])$$
$$z_1 = \textbf{Attn}(q_1, M)$$
$$h'_1 = \tanh(W[z_1; q_1])) \qquad \textbf{dec}_\theta(h, (y_0,\ y_1)) = E^\top h'_1$$

Illustration: Recurrent Decoder

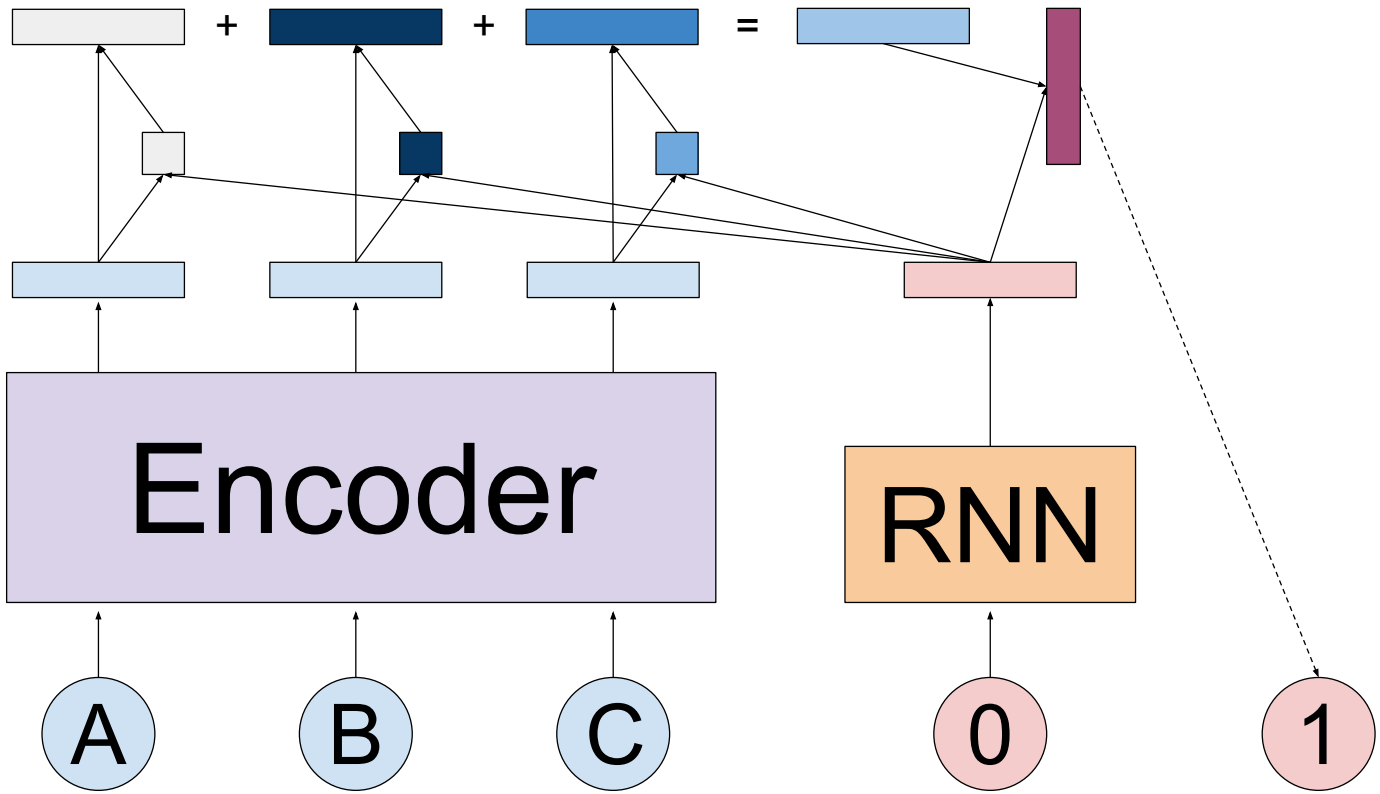
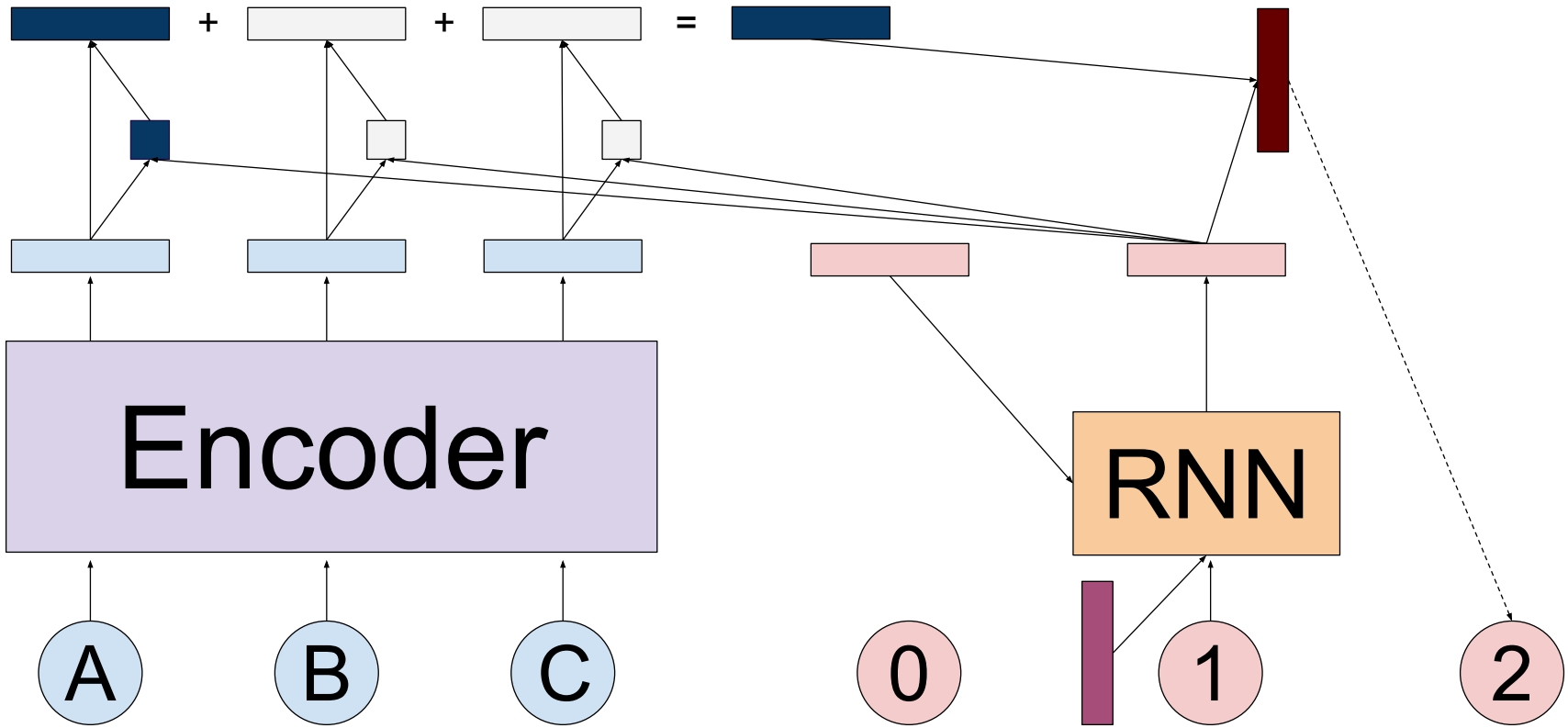


Illustration: Recurrent Decoder



Visualization of Learned Attention Weights

Rows: target tokens (French), columns: source tokens (English)

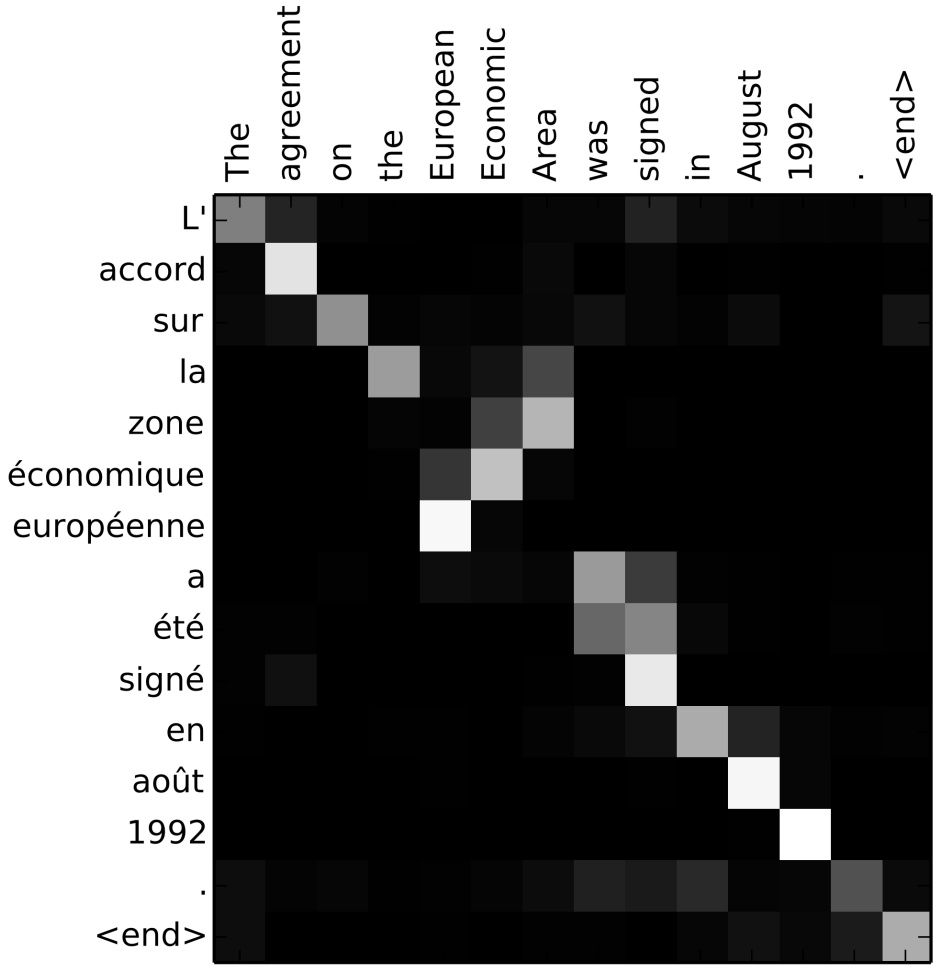
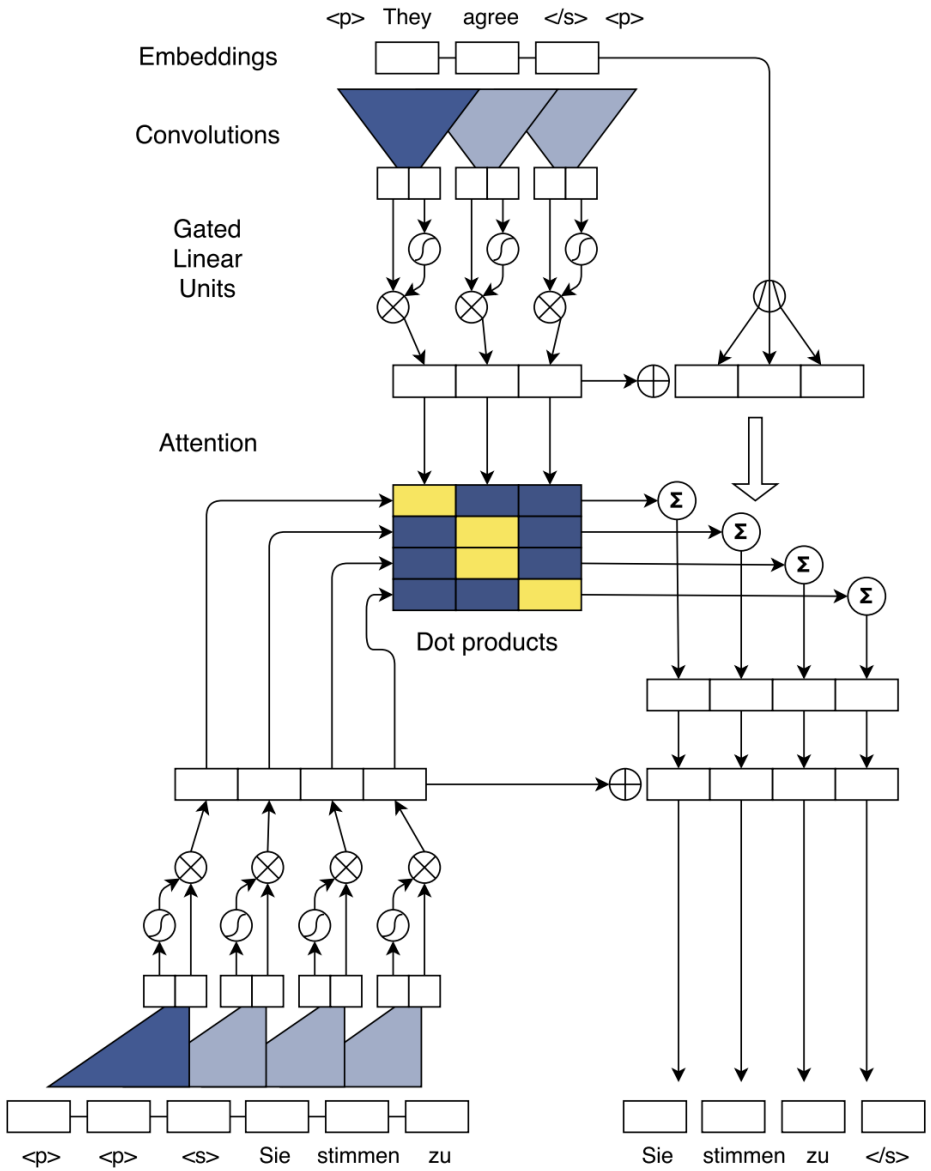


Image credit: Bahdanau et al. (2016)

Example: Convolutional Seq2Seq (Gehring et al., 2017)

- Both encoder & decoder parameterized by convolutional blocks with filter size k
- Stacked to increase input field size
  - E.g., stacking 6 blocks with filter size $k = 5$ allows the model to condition on 25 tokens $(5 + 4 \times 5)$ at each prediction
- Motivation: Avoid recurrent computation, parallelize training
- Proposed lots of important ideas, used by transformer
  - We can compute all target losses in one shot
  - Repeated application of attention over multiple layers ("attention with multiple hops")
  - Position embeddings
  - Stabilization tricks for learning like normalization

Illustration: Convolutional Seq2Seq (Gehring et al., 2017)



Example: Transformer Decoder

- Recall multi-head attention layer: Given $Q \in \mathbb{R}^{T' \times d}$ and $K, V \in \mathbb{R}^{T \times d}$, compute

$$\textbf{Attn}_\theta^H(Q,K,V) = g_\theta\Big(\underbrace{\bigoplus_{i=1}^{H}}_{\text{concat}} \textbf{Attn}\big(\underbrace{f_\theta^{(q,i)}(Q)}_{d/H}, \underbrace{f_\theta^{(k,i)}(K)}_{d/H}, \underbrace{f_\theta^{(v,i)}(V)}_{d/H}\big)\Big)$$

- Given source encodings $h \in \mathbb{R}^{T \times d}$ and target history $y_{<t}$, set $Z = E(y_{<t}) \in \mathbb{R}^{(t-1) \times d}$ and (omitting non-attention details)

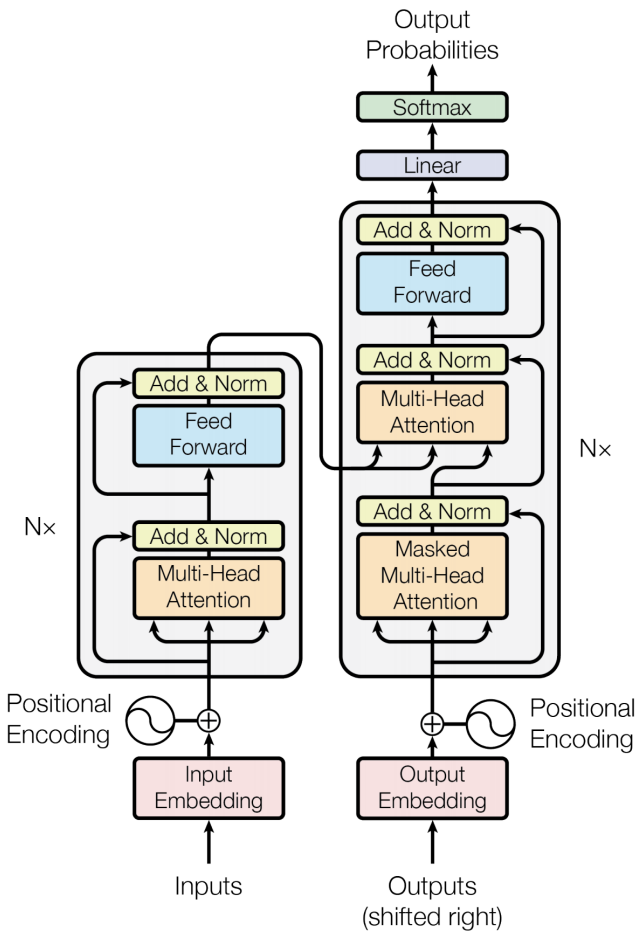
$$Q = \textbf{Attn}_\theta^H(Z,Z,Z) + Z$$
$$Z' = \textbf{Attn}_\theta^H(Q,h,h) + Q$$

- Stack for many layers (feeding $Z = Z'$ as input), finally define

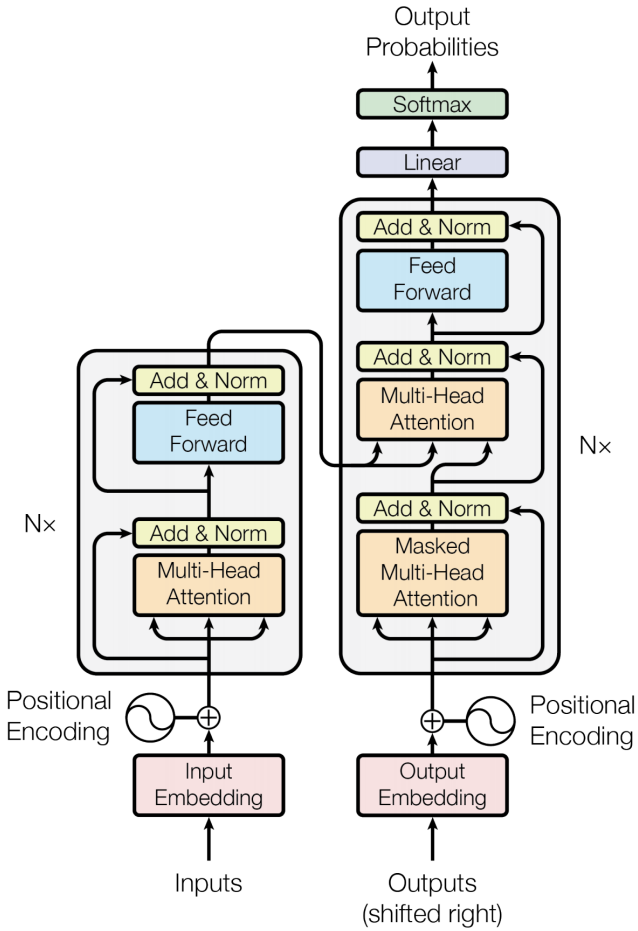
$$\textbf{dec}_\theta(h, y_{<t}) := \underbrace{W}_{V \times d} \underbrace{z'_{t-1}}_{d \times 1}$$

where z'_{t-1} is the last embedding of last layer's Z'

Transformer Encoder-Decoder

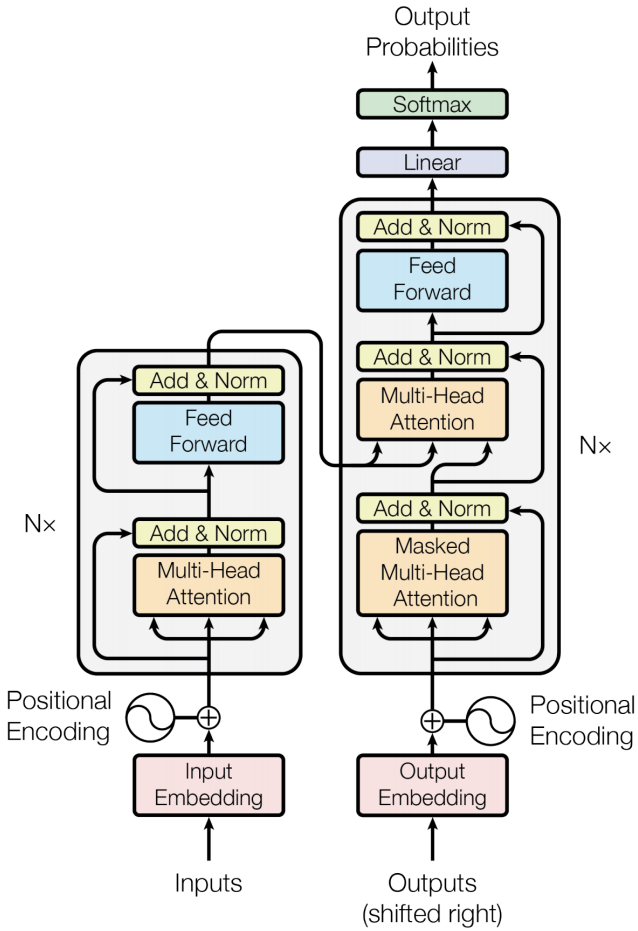


Transformer Encoder-Decoder



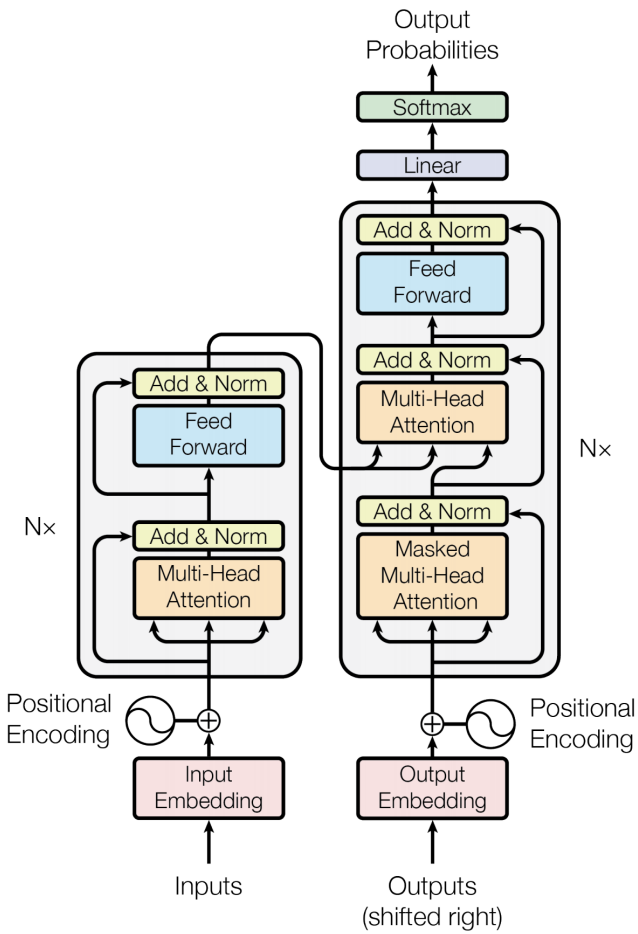
- Note $N\times$: Repeated attention (with multiple heads!)

Transformer Encoder-Decoder



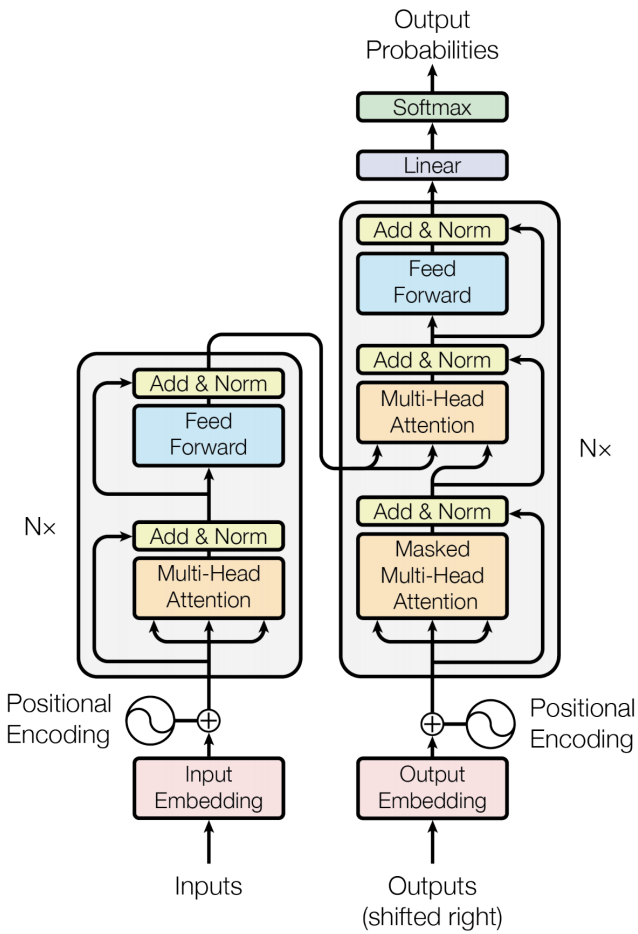
- Note $N\times$: Repeated attention (with multiple heads!)
- Decoder has two types of attention
  - Self-attention: Within targets
  - Cross-attention: Between targets and source embeddings

Transformer Encoder-Decoder



- Note $N\times$: Repeated attention (with multiple heads!)
- Decoder has two types of attention
  - Self-attention: Within targets
  - Cross-attention: Between targets and source embeddings
- Efficient parallelizable training
  1. Encode source (one-time)
  2. Decode over entire target sequence at once. Every layer/position computes masked self-attention (and unmasked cross-attention)

Transformer Encoder-Decoder



- Note $N\times$: Repeated attention (with multiple heads!)
- Decoder has two types of attention
  - Self-attention: Within targets
  - Cross-attention: Between targets and source embeddings
- Efficient parallelizable training
  1. Encode source (one-time)
  2. Decode over entire target sequence at once. Every layer/position computes masked self-attention (and unmasked cross-attention)

Transformer was the first to rely entirely on self-attention for encoding, without use of recurrent or convolutional components

- Attention Is All You Need (Vaswani et al., 2017)

Computational Efficiency (Per-Layer)

Layer Type	Complexity	# Sequential Ops	Max Path Length
Self-Attention	$O(T^2 d)$	$O(1)$	$O(1)$
Recurrent	$O(Td)$	$O(T)$	$O(T)$
Convolutional	$O(kfTd)$	$O(1)$	$O(\log_k(T))$
Self-Attention$_r$	$O(rTd)$	$O(1)$	$O(T/r)$

- Sequence length T, dimension d, filter width k, number of filters f
- Self-Attention$_r$: attention limited to $r \leq T$ tokens
- Computation bottleneck of transformer: quadratic dependence on length
  - Very active research on alleviating the bottleneck (Reformer, Linformer, Performer, Longformer, *inter alia*)

Machine Translation (MT)

- Goal: Translate text from one language to another. One of the oldest problems in artificial intelligence (e.g., Georgetown-IBM experiment, 1954)
- Early '90s to early 2010: Statistical MT (SMT), huge engineering pipeline
  - Infer word/phrase alignments with latent-variable models ("IBM models").
  - Run syntactic analyzers (e.g., parser) to extract features and manipulate text (e.g., phrase re-ordering).
  - Use a separately trained language model to enforce fluency, etc.
- From 2014 onward: Neural MT (NMT)
  - Invention of seq2seq and attention for MT: Single model trained end-to-end
  - This really turned the table in NLP, deep learning became dominant

Training Data for MT

- **Parallel corpus**: Pairs (x, y) where x is some text in language A and y is text of "same meaning" in language B
  - Single parallel corpus gives 2 directions in supervision: $A \mapsto B$ and $B \mapsto A$
- Many natural annotations to exploit! Examples
  - **Europarl** (Koehn, 2005): Proceedings of European Parliament in 21 European languages
  - **UN Corpus** (Ziemski et al., 2016): Public UN documents in 6 languages
  - **IWSLT17** (Mauro et al., 2017): Translated transcripts of TED talks and university lectures
- Data not infinite (as in language modeling), but typically assumes a great deal of supervision
  - E.g., WMT14 contains 4.5 million English-German sentence pairs for training
  - Still data can be very limited for languages without many speakers (low-resource languages)

Automatic Evaluation of Translation Output

- Likelihood/perplexity of reference text?
- Problem: Same text can have multiple valid translations with varying qualities

  Translation output: It is a guide to action which ensures that the military always obeys the commands of the party

  Reference 1: It is a guide to action that ensures that the military will forever heed Party commands

  Reference 2: It is the guiding principle which guarantees the military forces always being under the command of the Party

  Reference 3: It is the practical guide for the army always to heed the directions of the party

Automatic Evaluation of Translation Output

- Likelihood/perplexity of reference text?
- Problem: Same text can have multiple valid translations with varying qualities

  Translation output: It is a guide to action which ensures that the military always obeys the commands of the party

  Reference 1: It is a guide to action that ensures that the military will forever heed Party commands

  Reference 2: It is the guiding principle which guarantees the military forces always being under the command of the Party

  Reference 3: It is the practical guide for the army always to heed the directions of the party

Automatic Evaluation of Translation Output

- Likelihood/perplexity of reference text?
- Problem: Same text can have multiple valid translations with varying qualities

  Translation output: It is a guide to action which ensures that the military always obeys the commands of the party

  Reference 1: It is a guide to action that ensures that the military will forever heed Party commands

  Reference 2: It is the guiding principle which guarantees the military forces always being under the command of the Party

  Reference 3: It is the practical guide for the army always to heed the directions of the party

- One idea: Report the **precision** of predicted unigram types in references r

$$p_1 = \frac{\sum_r \#\text{ predicted unigram types present in } r}{\#\text{ predicted unigram types}} = 0.94$$

Similarly can compute bigram precision $p_2 = 0.59$

BLEU

- Report the harmonic mean of n-gram precisions for small values of n (e.g., $n = 1, 2, 3, 4$), calibrated by a brevity penalty

$$\text{BLEU} = \min\left(1, \underbrace{\frac{\sum_i \widehat{T}'_i}{\sum_i T'_i}}_{< 1 \text{ if outputs shorter than refs}}\right) \times \underbrace{\left(\prod_{n=1}^{4} p_n\right)^{\frac{1}{4}}}_{\text{avg. } n\text{-gram precision}}$$

- $\widehat{T}'_i$: Ouput translation length in example i
- T'_i: Reference translation length in example i, if multiple references can choose shortest or closest in length wrt. output
- This is a **corpus-level** statistic: bigger corpus, more reliable
- Flawed but popular and convenient, correlates with human judgment of translation quality
  - BLEU: a Method for Automatic Evaluation of Machine Translation (Papineni et al., 2002)

Empirical Performance

- Attention vs no attention: dramatic difference
  - Attending over source encodings > 10 higher BLEU compared to single-vector encoding (Bahdanau et al., 2016)
- Reported results on WMT14 English→German (4.5 million sentence pairs, 110 million target words for training)
  - Winner of WMT14 (SMT): 20.7
  - Input-feeding LSTM (ensemble): 23.0
  - Convolutional Seq2Seq (ensemble): 26.4
  - Transformer: 28.4
  - T5*: 32.1

  $*$: Trained on external data. T5 (Raffel et al., 2019) is a transformer encoder-decoder model trained on a large collection of NLP tasks (including MT) framed as seq2seq problems

Direct Optimization of BLEU

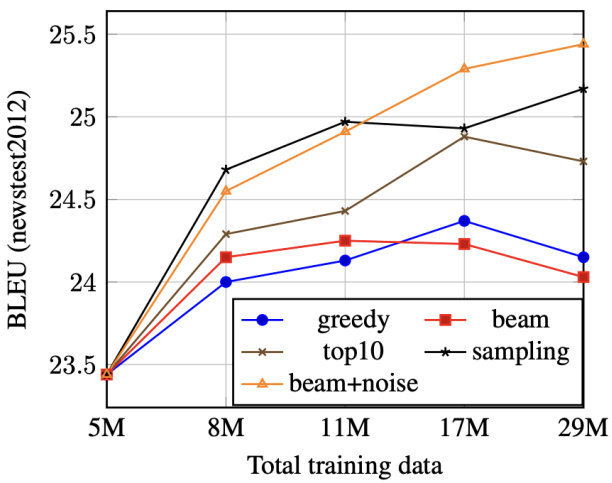
- Training: Per-token cross-entropy loss $\widehat{J}(\theta)$
- Can follow up to directly optimize BLEU score

$$\widehat{J}_{\mathrm{RL}}(\theta) = -\sum_{i=1}^{N}\sum_{j=1}^{M} p_\theta(y_j^{(i)}|x^{(i)})\mathrm{BLEU}(y^{(i)}, y_j^{(i)})$$

  Drawing M iid samples $y_1^{(i)} \dots y_M^{(i)} \sim p_\theta(\cdot|x^{(i)})$
- Approximation of expected BLEU score by sampling
- Workflow: First fully optimize model by cross entropy. Then optimize $\alpha\widehat{J}(\theta) + \widehat{J}_{\mathrm{RL}}(\theta)$ where α is some small value (e.g., 0.01).
- Can give a slight improvement in BLEU, but found to not improve translation quality by human judgment (Wu et al., 2016)
- Can use more refined methods, e.g., minimimum risk training (Shen et al., 2016), and optimize better non-differentiable score, e.g., SIMILE (Wieting et al., 2019)

Back-Translation

- Data augmentation method for semi-supervised MT
  1. Train an initial translation system on (limited) parallel corpus.
  2. Apply the system on a large quantity of target monolingual text to create a new parallel corpus (synthetic source).
  3. Train a final translation system on the union of all data.
- Effective, especially for low-resource but even for high-resource setting



Performance of different decoding strategies for back-translation

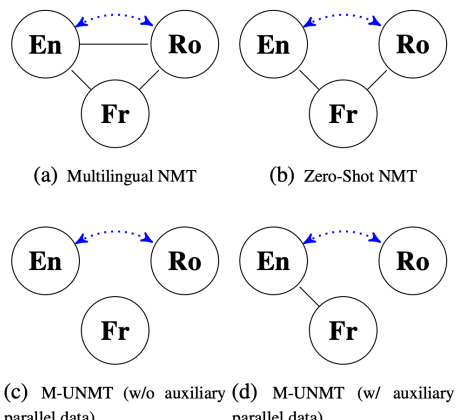
(Edunov et al., 2018)

Multilingual, Zero-Shot, Unsupervised Translation

- Learn a *single* model to translate between $k > 1$ languages
- Basic approach: Data preprocessing (Johnson et al., 2017)

$$(\langle \text{bos} \rangle, \text{Hello}, \langle \text{2es} \rangle, \langle \text{eos} \rangle) \quad \Rightarrow \quad (\langle \text{bos} \rangle, \text{¿Hola}, \langle \text{eos} \rangle)$$

- Emergence of **zero-shot translation**
  - Training data: German-English and English-French
  - Multilingual model, trained with $\langle \text{2en} \rangle, \langle \text{2de} \rangle, \langle \text{2fr} \rangle$
  - Test time: Give German text with $\langle \text{2fr} \rangle$!
  - Even though no direct German-French supervision, the model outputs nontrivial translation (but still bad compared to supervised)
- Can also consider a setting with zero parallel corpus



Various multilingual NMT setups (Garcia et al., 2020)