DeepHack.CISS

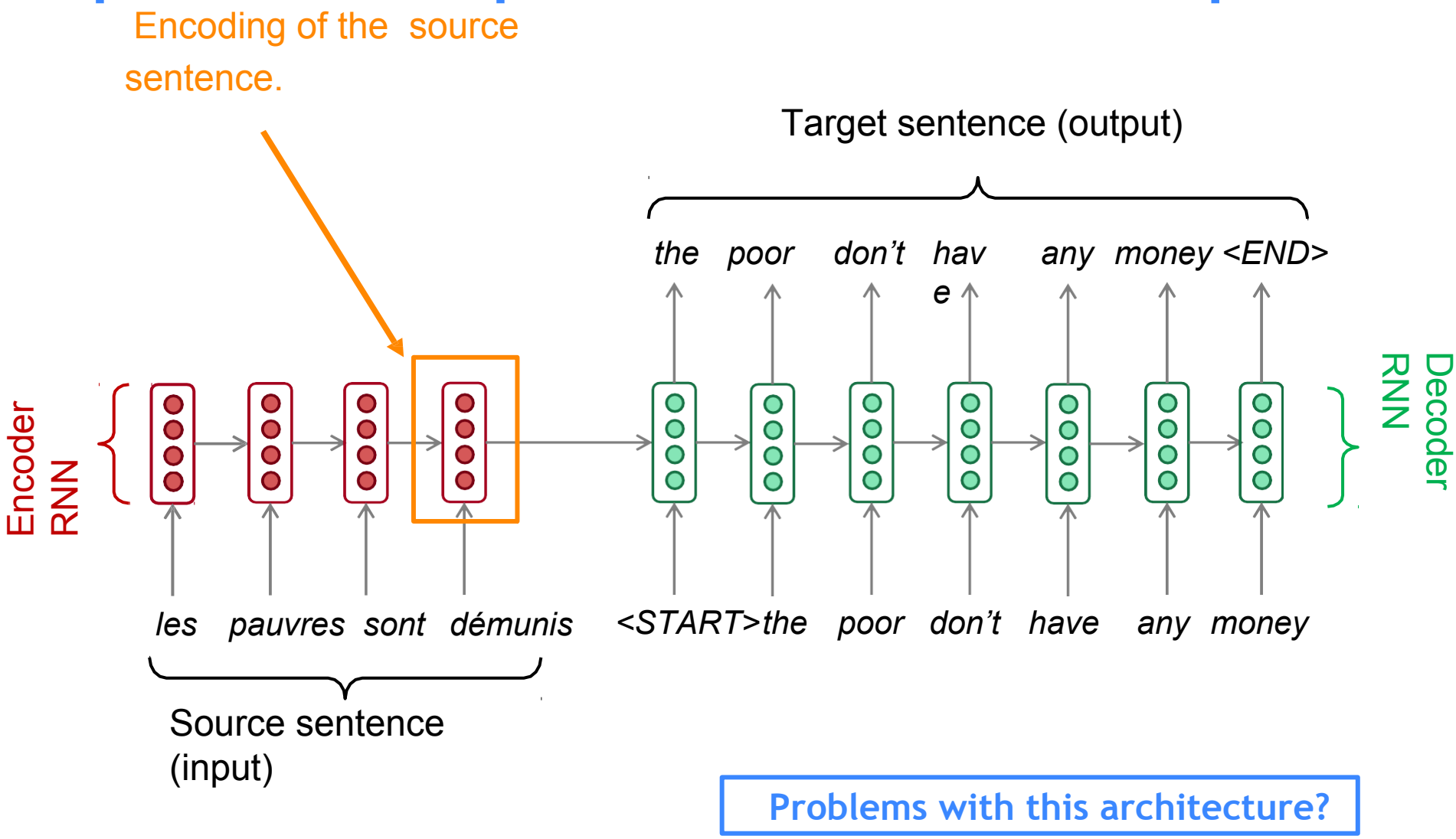
Attention

Valentin Malykh

Overview

- More types and uses of attention
  - Refresh and math
  - Pointer-sentinel model
  - Self-attention/intra-detention for summarization

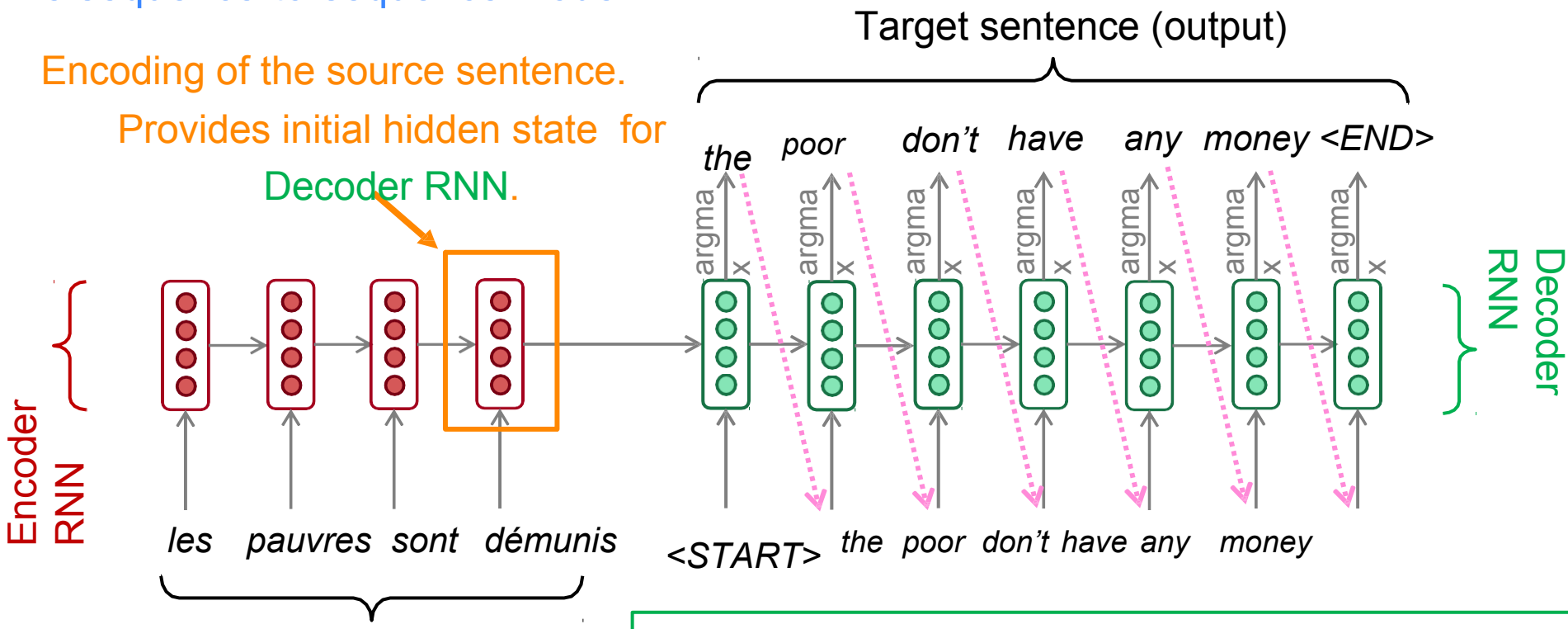
Sequence-to-sequence: the bottleneck problem



Problems with this architecture?

Neural Machine Translation (NMT)

The sequence-to-sequence model



Decoder RNN is a Language Model that generates target sentence conditioned on encoding.

Encoder RNN produces an encoding of the source sentence.

Note: This diagram shows **test time** behavior: decoder output is fed in as next step's input

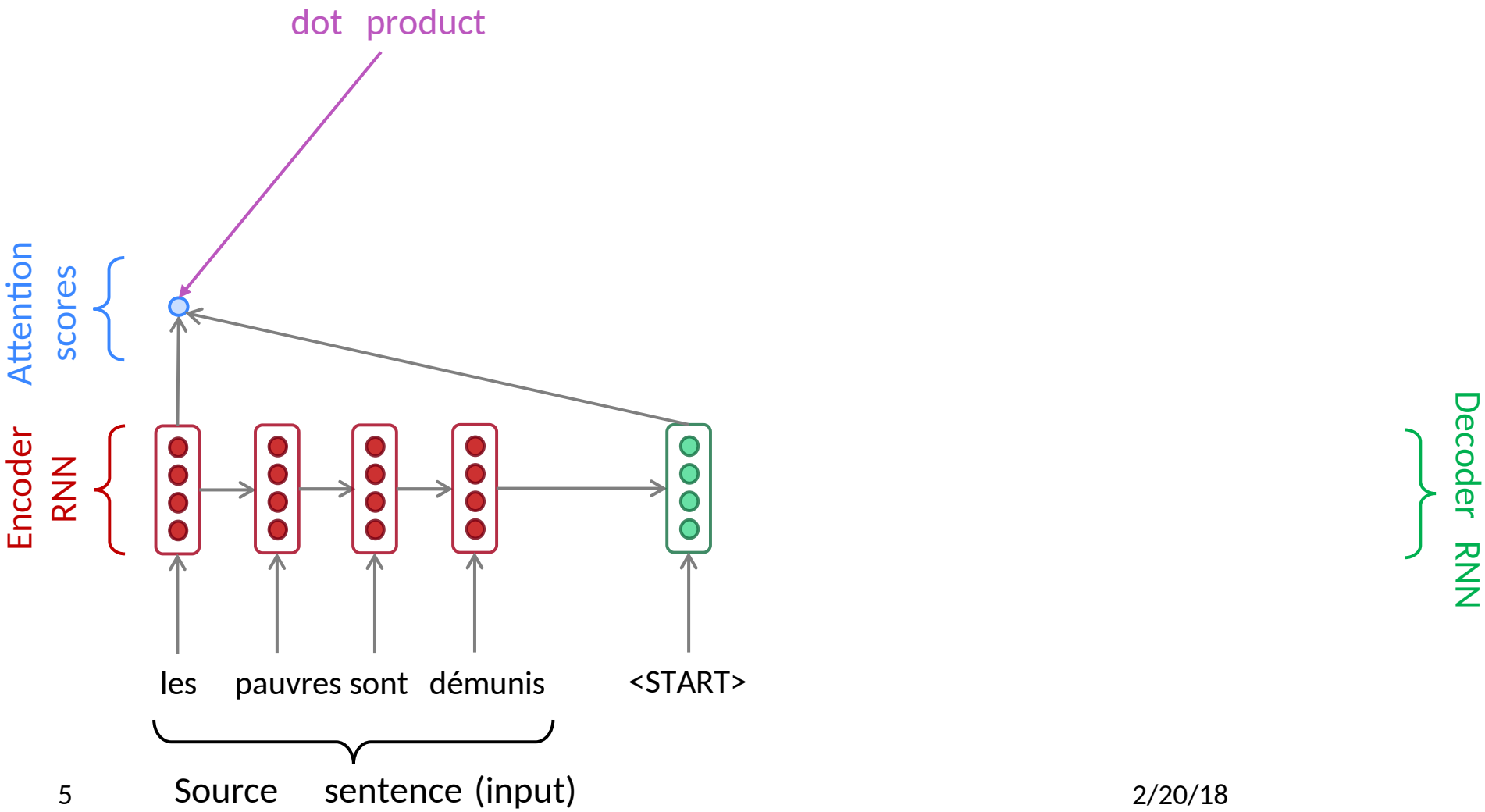
Attention

- **Attention** provides a solution to the bottleneck problem.

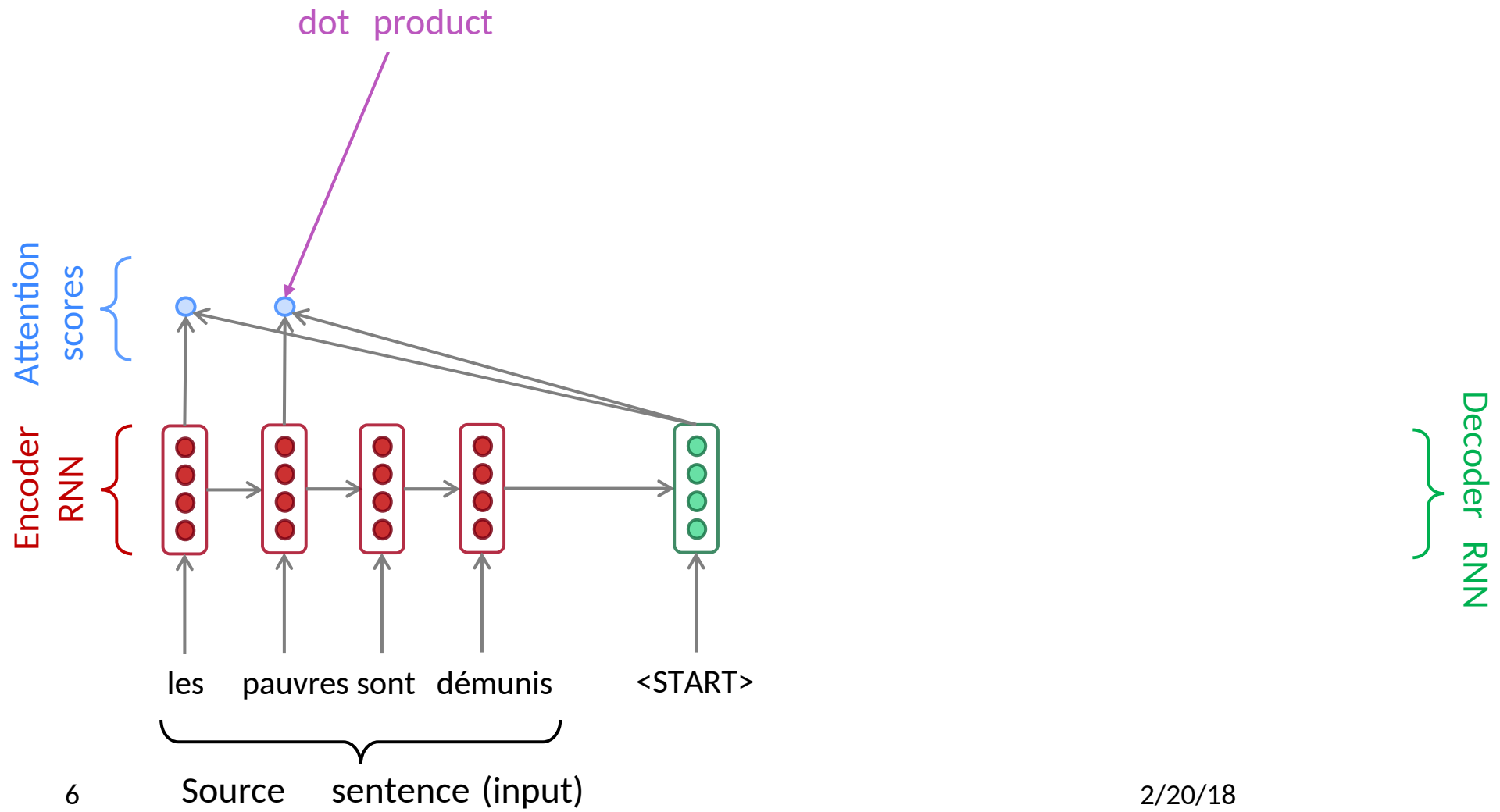
- Core idea: on each step of the decoder, *focus on a particular part* of the source sequence

- First we will show via diagram (no equations), then we will show with equations

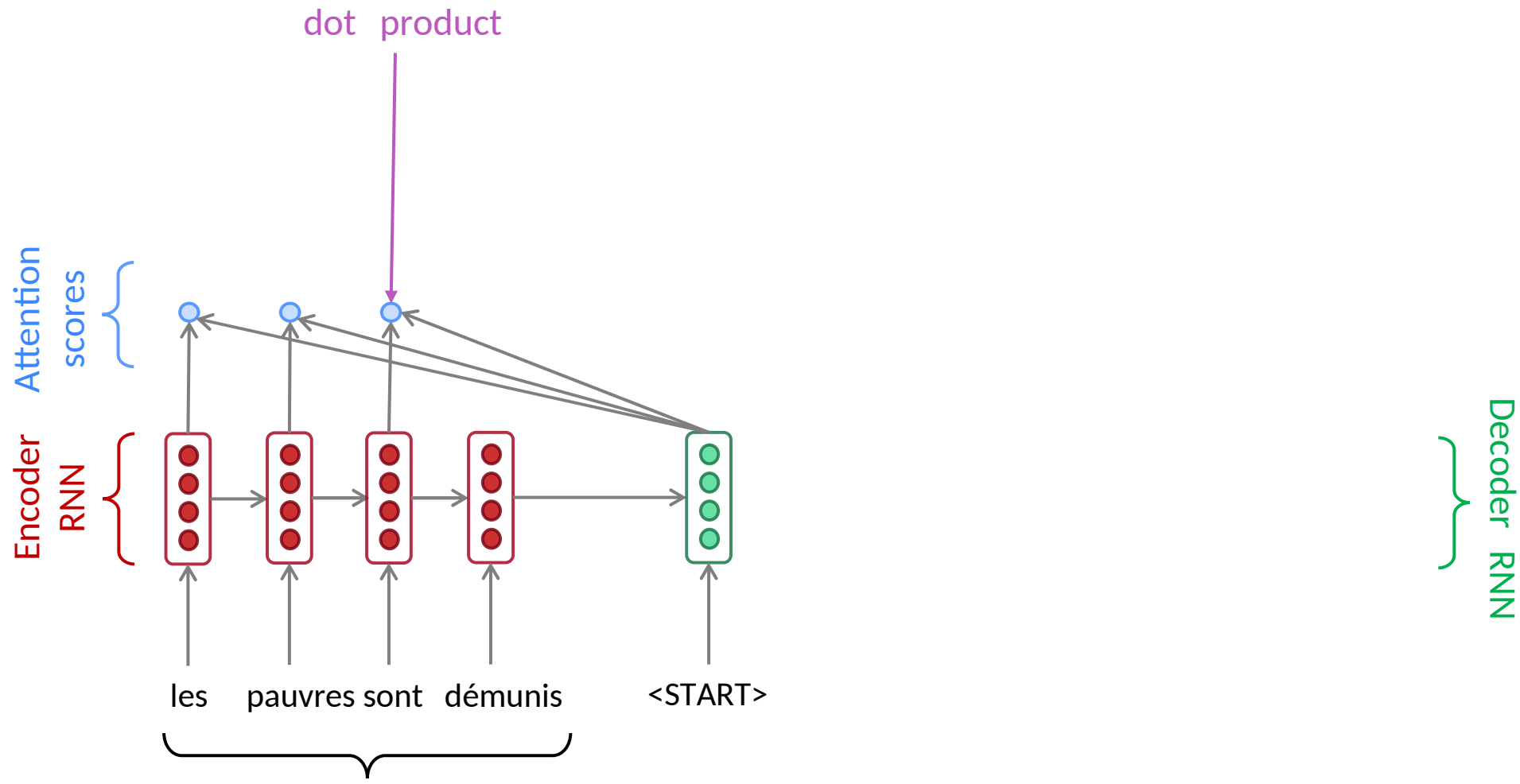
Recap: Sequence-to-sequence with attention



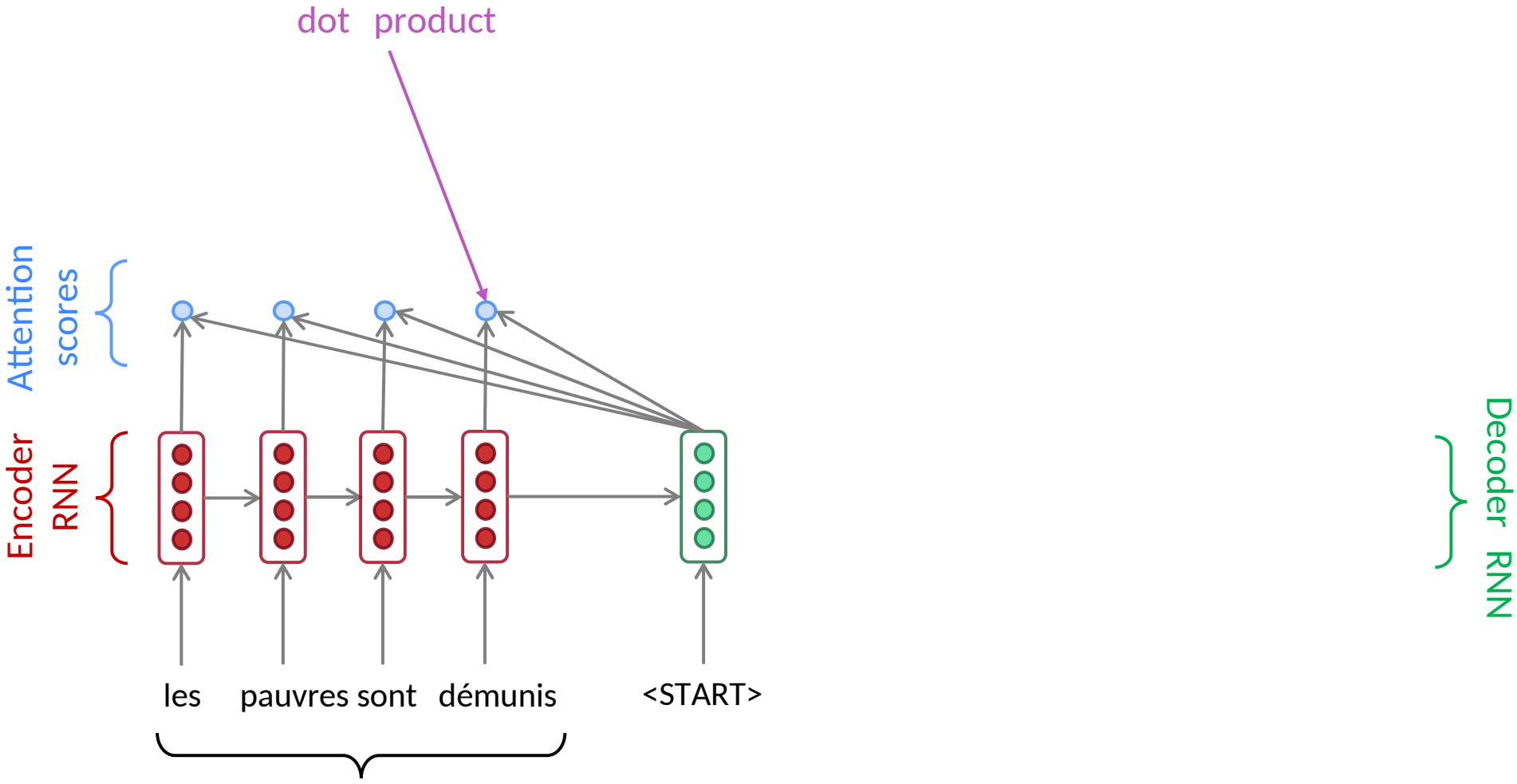
Recap: Sequence-to-sequence with attention



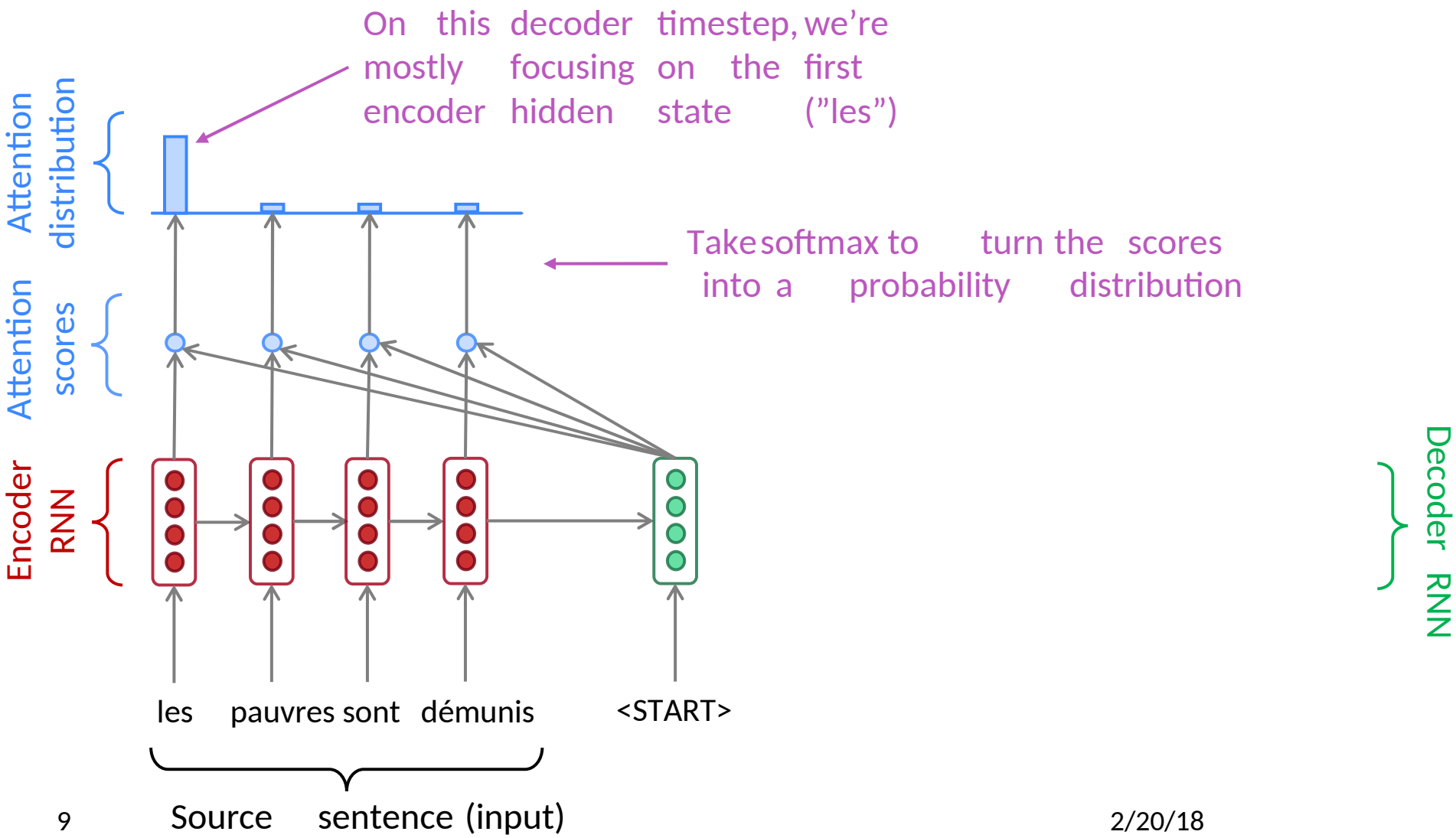
Recap: Sequence-to-sequence with attention



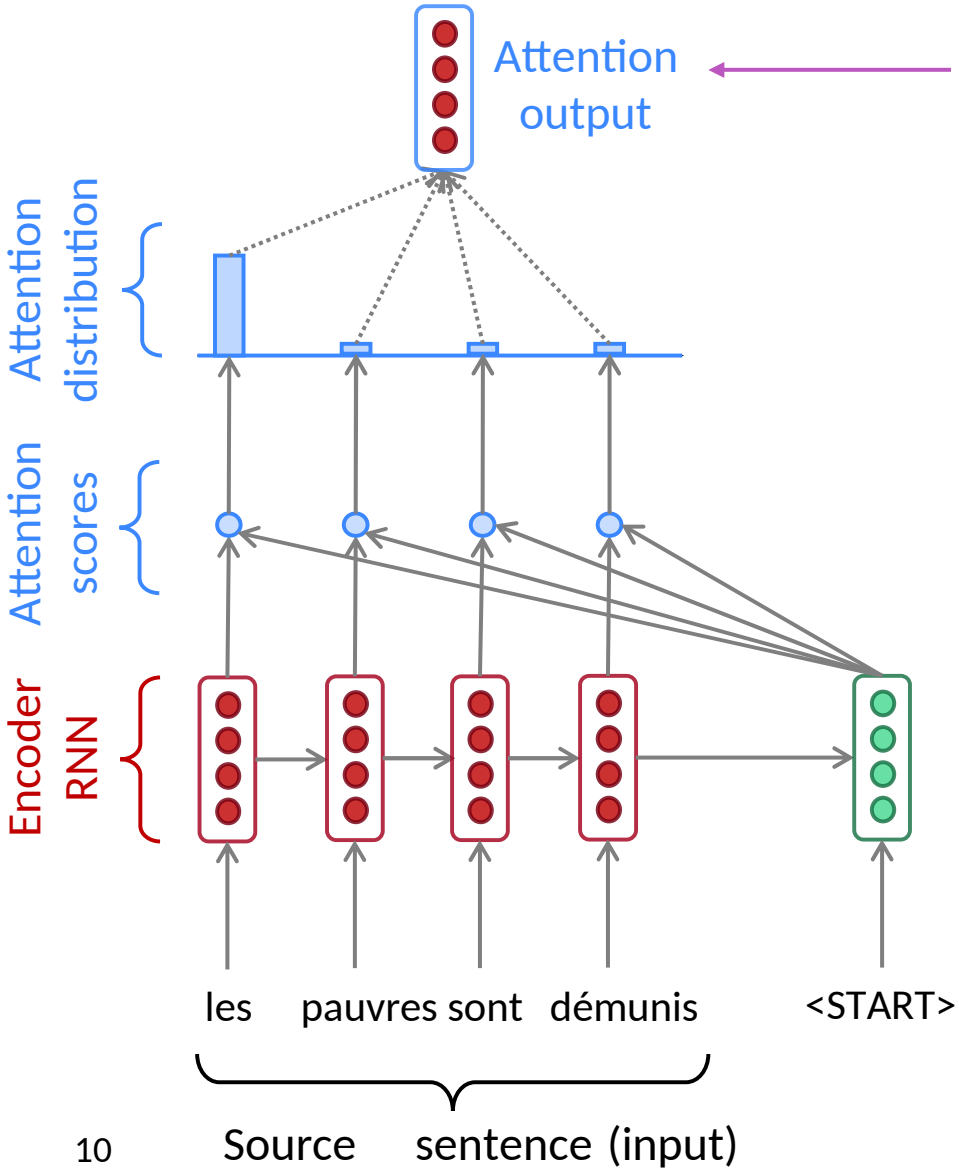
Recap: Sequence-to-sequence with attention



Recap: Sequence-to-sequence with attention



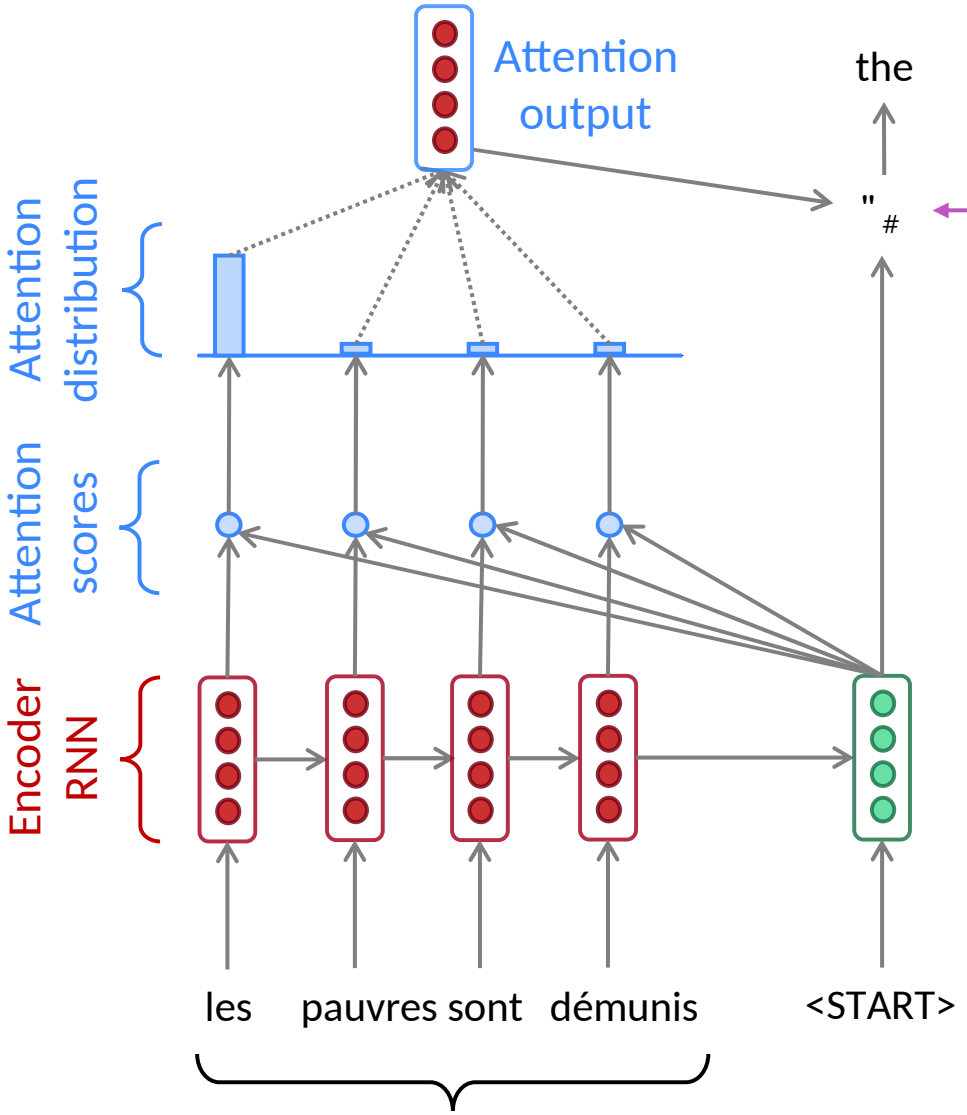
Recap: Sequence-to-sequence with attention



Use the attention distribution to take a **weighted sum** of the encoder hidden states.

The attention output mostly contains information the hidden states that received high attention.

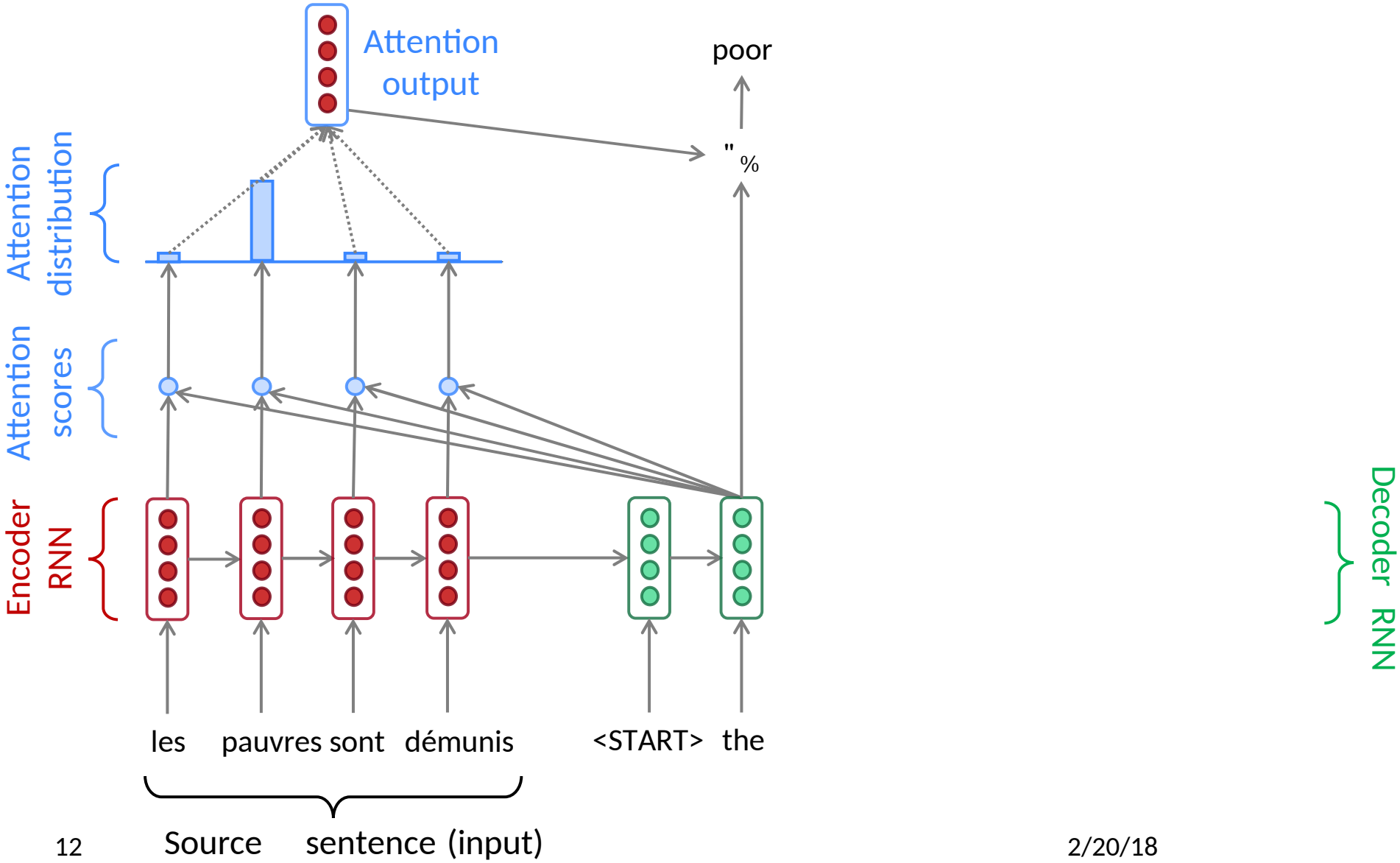
Recap: Sequence-to-sequence with attention



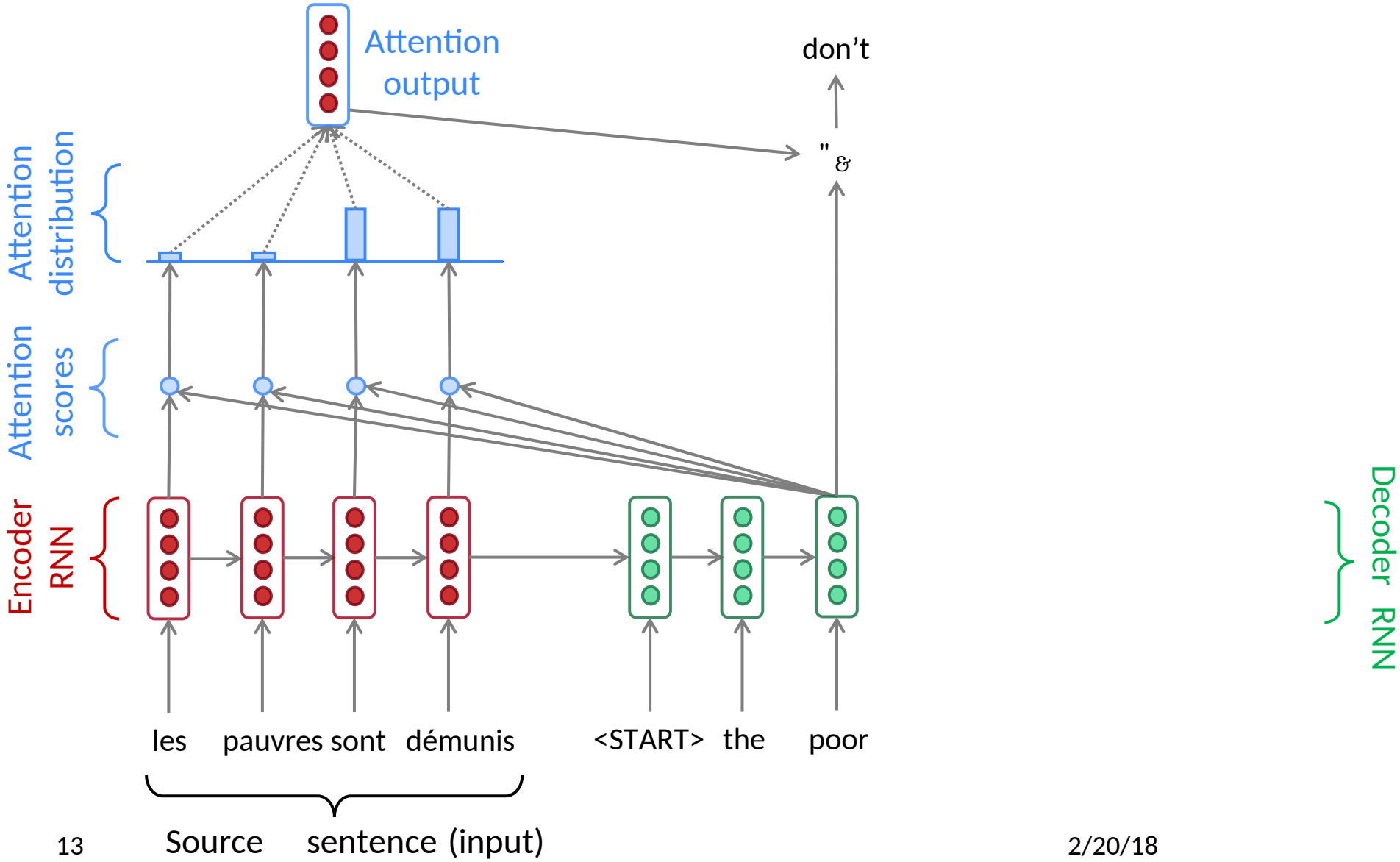
Concatenate attention output with decoder hidden state, then use to compute as before

Decoder RNN

Recap: Sequence-to-sequence with attention



Recap: Sequence-to-sequence with attention



Basic Attention equations

- We have encoder hidden state $h_1, \ldots, h_N \in \mathbb{R}^h$
- On timestep t, we have decoder hidden state $s_t \in \mathbb{R}^h$
- We get the attention scores $\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$ for this step:

$$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$$

- We takesoftmax to get the attention distri $\alpha^t = \text{softmax}(\boldsymbol{e}^t) \in \mathbb{R}^N$ probability distribution and sums to 1)

$$\alpha^t = \text{softmax}(\boldsymbol{e}^t) \in \mathbb{R}^N$$

- We use it to take a weighted sum of the encoder hidden states to get the atention output

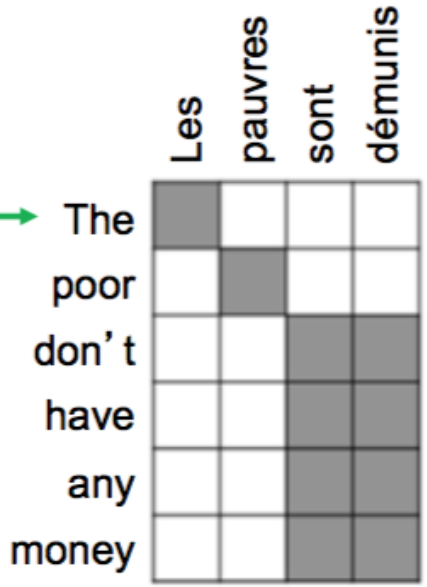
$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

- Finally we concatenate the attention output state $s_t \in \mathbb{R}^h$ and proceed as in the non-attention seq2seq model

$$[\boldsymbol{a}_t; \boldsymbol{s}_t] \in \mathbb{R}^{2h}$$

Recap: Attention is great

- Attention significantly improves NMT performance
  - It's very useful to allow decoder to focus on certain parts of the source
- Attention solves the bottleneck problem
  - Attention allows decoder to look directly at source; bypass bottleneck
- Attention helps with vanishing gradient problem
  - Provides shortcut to faraway states
- Attention provides some interpretability
  - By inspecting attention distribution, we can see what the decoder was focusing on
  - We get alignment for free!
  - This is cool because we never explicitly trained an alignment system
  - The network just learned alignment by itself



Attention is a general Deep Learning technique

- Last time: We saw that attention is a great way to sequence-to-sequence model for Machine Translation.
- However: Today we'll see attention is applied to many architectures (not just seq2seq) and many tasks (not just MT)

- **More general definition of attention**:
  - Given a set of vector values, and a vector query, **attention** is a technique to compute a weighted sum of the values, dependent on the query.
- We sometimes say that the query attends to the values.
- For example, in the seq2seq + attention model, each decoder hidden state attends to the encoder hidden states.

Attention is a general Deep Learning technique

More general definition of attention:

Given a set of vector values, and a vector query, **attention** is a technique to compute a weighted sum of the values, dependent on the query.

- **Intuition**:
  - The weighted sum is a selective summary of the information contained in the values, where the query determines which values to focus on.
  - Attention is a way to obtain a fixed-size representation of an arbitrary set of representations (the values), dependent on some other representation (the query).

There are several attention variants

- We have some values $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \in \mathbb{R}^{d_1}$ and a query $\boldsymbol{s} \in \mathbb{R}^{d_2}$
- $\boldsymbol{a} \in \mathbb{R}^{d_1}$ (sometimes called the context vector) from the attention scores $\boldsymbol{e} \in \mathbb{R}^N$ (or attention logits) like so:

$$\alpha = \text{softmax}(\boldsymbol{e}) \in \mathbb{R}^N \quad \text{(take softmax)}$$

$$\boldsymbol{a} = \sum_{i=1}^{N} \alpha_i \boldsymbol{h}_i \in \mathbb{R}^{d_1} \quad \text{(take weighted sum)}$$

- However, there are several ways you can compute $\boldsymbol{e} \in \mathbb{R}^N$

Attention variants

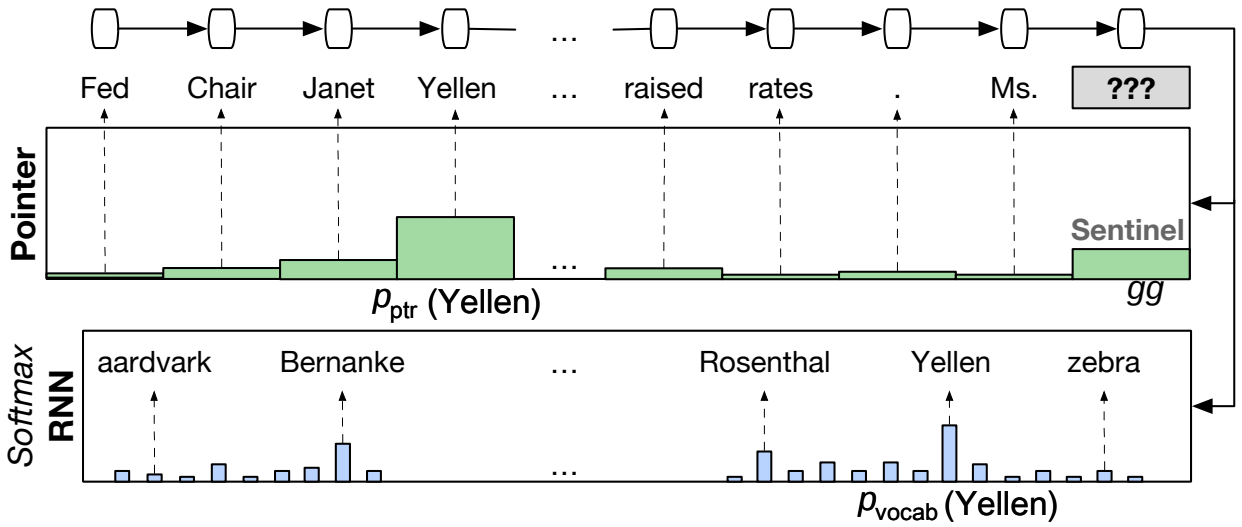
There are several ways you can compute $\boldsymbol{e} \in \mathbb{R}^N$ $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \in \mathbb{R}^{d_1}$ and $\boldsymbol{s} \in \mathbb{R}^{d_2}$:

- Basic dot-product attenti $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{h}_i \in \mathbb{R}$
  - Note: this assum $d_1 = d_2$
  - This is the version we saw earlier
- Multiplicative attentic $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{W} \boldsymbol{h}_i \in \mathbb{R}$
  - Where $\boldsymbol{W} \in \mathbb{R}^{d_2 \times d_1}$
- Additive attention: $\boldsymbol{e}_i = \boldsymbol{v}^T \tanh(\boldsymbol{W}_1 \boldsymbol{h}_i + \boldsymbol{W}_2 \boldsymbol{s}) \in \mathbb{R}$
  - Where $\boldsymbol{W}_1 \in \mathbb{R}^{d_3 \times d_1}, \boldsymbol{W}_2 \in \mathbb{R}^{d_3 \times d_2}$ $\boldsymbol{v} \in \mathbb{R}^{d_3}$ is a weight vector

More information: http://ruder.io/deep-learning-nlp-best-practices/index.html#attention

Attention application: Pointing to words for language modeling

- Idea: Mixture Model of softmax and pointers:

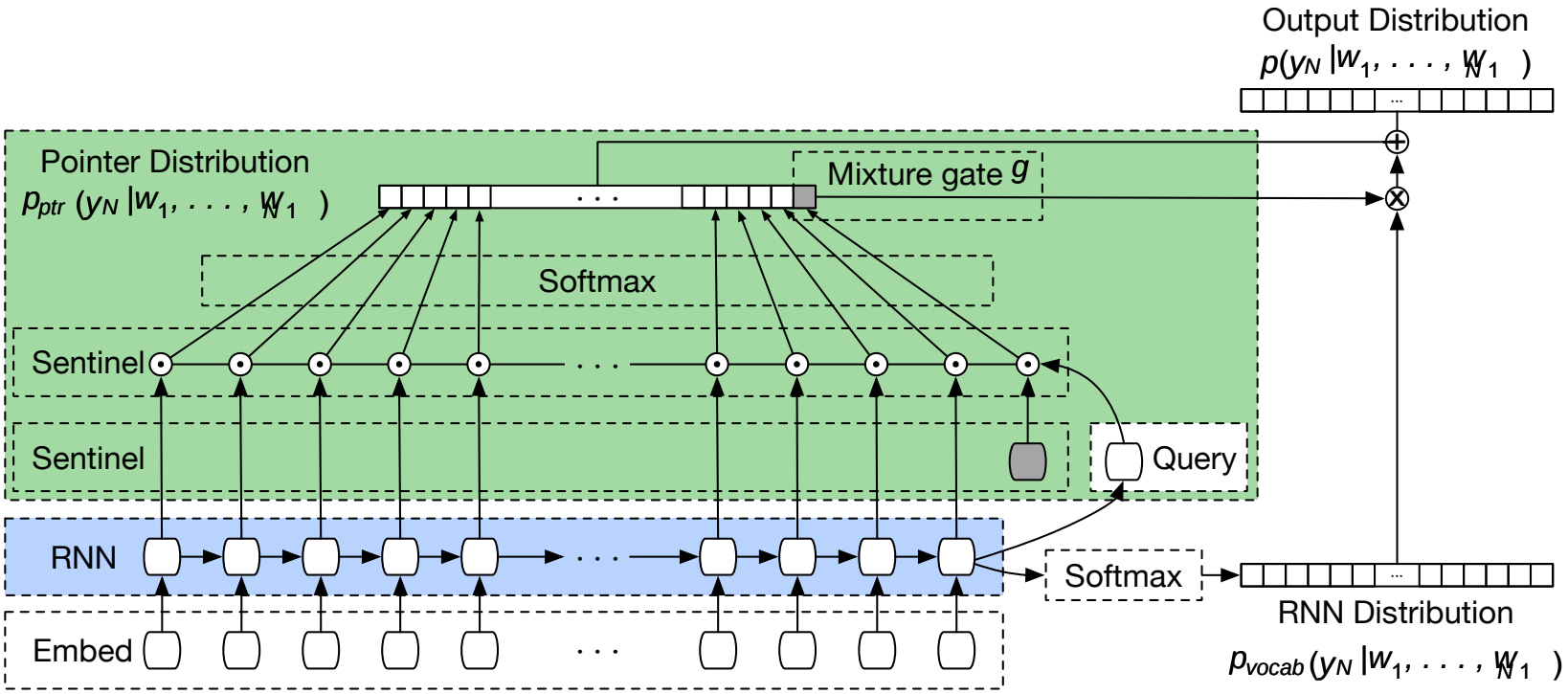


$p(\text{Yellen}) = g\, p_{\text{vocab}}(\text{Yellen}) + (1 \quad g)\, p_{\text{ptr}}(\text{Yellen})$

- PointerSentinel Mixture Models by Stephen Merity,Caiming Xiong, James Bradbury, Richard Socher

Pointer-Sentinel Model - Details

Pointer Sentinel Mixture Models



$$p(y_i|x_i) = g\, p_{\text{vocab}}(y_i|x_i) + (1-g)\, p_{\text{ptr}}(y_i|x_i)$$

$$z_i = q^T h_i, \qquad p_{\text{ptr}}(w) = \sum_{i \in I(w,x)} a_i,$$

$$a = \text{softmax}(z),$$

Pointer Sentinel for Language Modeling

Pointer Sentinel Mixture Models

Model	Parameters	Validation	Test
Mikolov & Zweig (2012) - KN-5	2M‡		141.2
Mikolov & Zweig (2012) - KN5 + cache	2M‡		125.7
Mikolov & Zweig (2012) - RNN	6M‡		124.7
Mikolov & Zweig (2012) - RNN-LDA	7M‡		113.7
Mikolov & Zweig (2012) - RNN-LDA + KN-5 + cache	9M‡		92.0
Pascanu et al. (2013a) - Deep RNN	6M		107.5
Cheng et al. (2014) - Sum-Prod Net	5M‡		100.0
Zaremba et al. (2014) - LSTM (medium)	20M	86.2	82.7
Zaremba et al. (2014) - LSTM (large)	66M	82.2	78.4
Gal (2015) - Variational LSTM (medium, untied)	20M	81.9 ± 0.2	79.7 ± 0.1
Gal (2015) - Variational LSTM (medium, untied, MC)	20M		78.6 ± 0.1
Gal (2015) - Variational LSTM (large, untied)	66M	77.9 ± 0.3	75.2 ± 0.2
Gal (2015) - Variational LSTM (large, untied, MC)	66M		73.4 ± 0.0
Kim et al. (2016) - CharCNN	19M		78.9
Zilly et al. (2016) - Variational RHN	32M	72.8	71.3
Zoneout + Variational LSTM (medium)	20M	84.4	80.6
Pointer Sentinel-LSTM (medium)	21M	72.4	**70.9**

Attention application: Intra-Decoder attention forSummarization

- Longer document summarization. Example:

Tony Blair has said he does not want to retire until he is 91 – as he unveiled pla
around the world. The defiant 61-year-old former Prime Minister said he had ‘decades’ still in him
he ever stepped down from his multitude of global roles. He told Newsweek magazine tha
government to go round the world to advise presidents and prime ministers on how to run
Newsweek Mr Blair said he did not want to retire until he was 91 years old Mr Blair
government to advise presidents and prime ministers on how to run their countries Mr Blair
president Bill Clinton when he took office in 1997. And he said he wanted to build up his
are ‘capable of changing global policy’. Last night, Tory MPs expressed horror at the prospect
30 years. Andrew Bridgen said: ‘We all know weak Ed Miliband’s called on Tony to give his
clearly gone to his head.’ (…)

- Summary:

  The former Prime Minister claimed he has 'decades' of work left in him. Joked he would 'turn to drink' if he ever stepped down from global roles. Wants to recruit former government heads to advise current leaders. He was 'mentored' by US president Bill Clinton when he started in 1997.

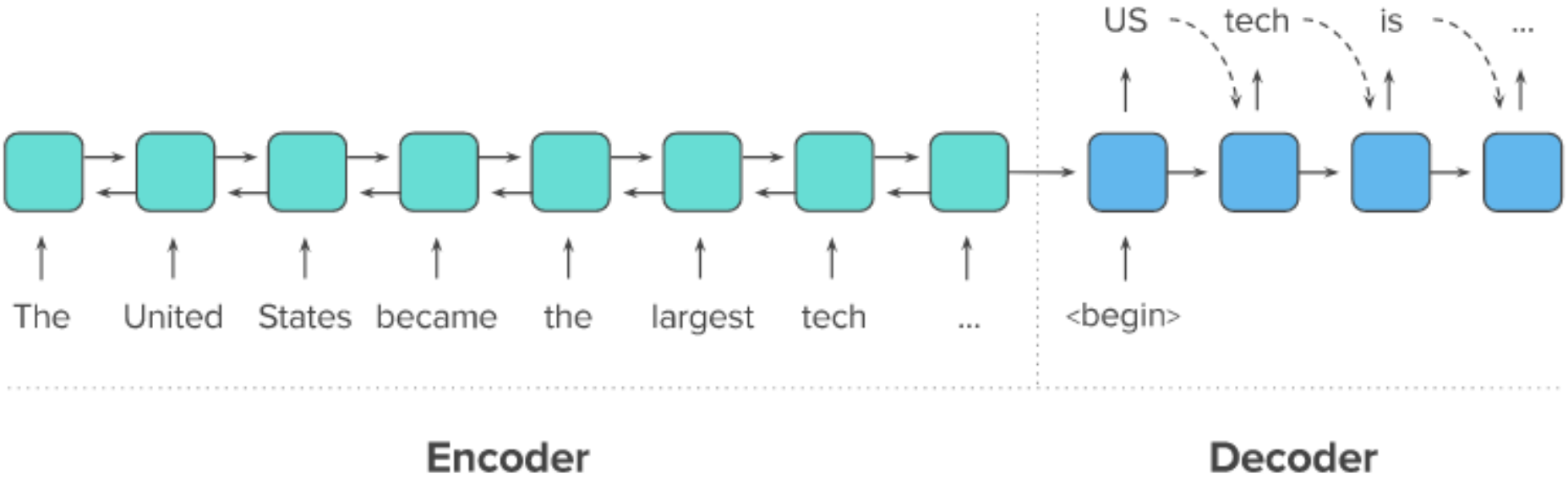
Attention application: Intra-Decoder attention for Summarization

- Based on paper:
- Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A Deep Reinforced Model for Abstractive Summarization
- But similar ideas appear elsewhere also

- Two necessary, new ingredients
  - Attention during generation
  - Reinforcement learning

Not in this lecture

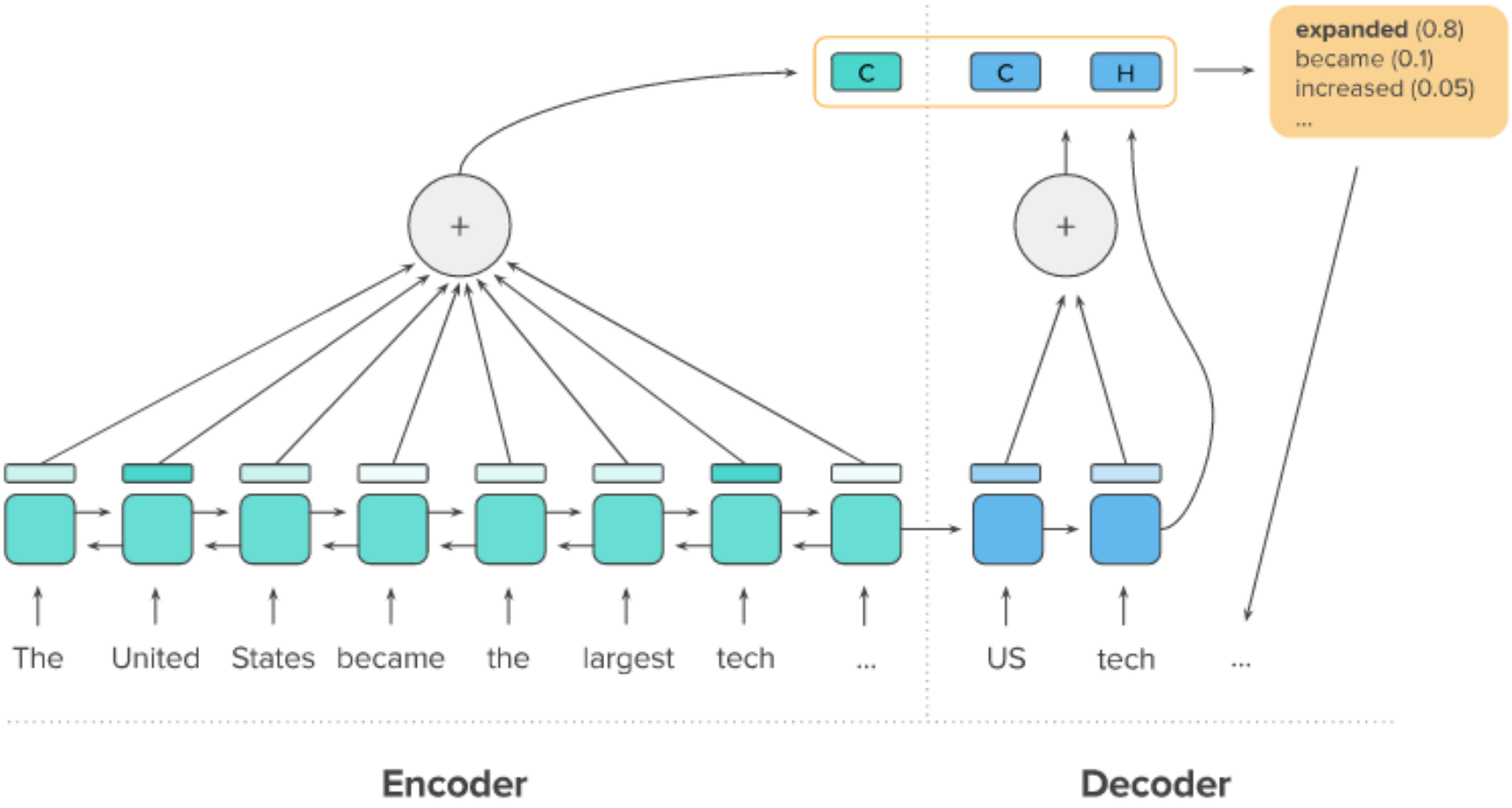
Attention application: Similar Seq2Seq Idea as in Translation



- Problems: For longer outputs (MT was just single sentences), the decoder starts to repeat itself

More advanced attention

1. More advanced encoder attention
2. Self-attention (= intra-decoder attention)



1. Details of this attention mechanism

- More advanced similarity function than simple inner product:

$$e_{ti} = f(h_t^d, h_i^e)$$

$$f(h_t^d, h_i^e) = {h_t^d}^T W_{\text{attn}}^e h_i^e$$

- Temporal attention function, penalizing input tokens that have obtained high attention scores in past decoding steps:

$$e'_{ti} = \begin{cases} exp(e_{ti}) & \text{if } t = 1 \\ \frac{exp(e_{ti})}{\sum_{j=1}^{t-1} \exp(e_{ji})} & \text{otherwise} \end{cases}$$

- Improves coverage and prevent repeated attention to same inputs

1. Details of this attention mechanism

- Combine softmax'ed weighted hidden states from encoder:

$$\alpha_{ti}^{e} = \frac{e'_{ti}}{\sum_{j=1}^{n} e'_{tj}}$$

$$c_t^e = \sum_{i=1}^{n} \alpha_{ti}^{e} h_i^e$$

- Remember softmax-normalized encoder for later!

2. Self-attention on decoder

- Self-attention: A general idea used in many RNN models (e.g. Language Models). The hidden states "attend to themselves", i.e. each hidden state attends to the previous hidden states of the same RNN.

- On step t, attends to previous **decoder** hidden states

$$e^d_{tt'} = h^{dT}_t W^d_{\text{attn}} h^d_{t'}$$

- Apply softmax to get attention distribution over previous hidden states $e^d_{tt'} = h^{dT}_t W^d_{\text{attn}} h^d_{t'}$ for t' = 1,...,t-1:

$$\alpha^d_{tt'} = \frac{exp(e^d_{tt'})}{\sum_{j=1}^{t-1} exp(e^d_{tj})}$$

- Compute decoder attention output: $c^d_t = \sum_{j=1}^{t-1} \alpha^d_{tj} h^d_j$

2. Combine softmax and pointers using both attention computations

- Compute probability of copying/pointing to word from input:

$$p(u_t = 1) = \sigma(W_u[h_t^d \| c_t^e \| c_t^d] + b_u)$$

- If not copying/pointing, use standard softmax:

$$p(y_t | u_t = 0) = \text{softmax}(W_{\text{out}}[h_t^d \| c_t^e \| c_t^d] + b_{\text{out}})$$

- If pointing, use encoder attention weights (from 2 slides ago)

$$p(y_t = x_i | u_t = 1) = \alpha_{ti}^e$$

- Combine everything:

$$p(y_t) = p(u_t = 1)p(y_t | u_t = 1) + p(u_t = 0)p(y_t | u_t = 0)$$

Summarization Results

cia documents reveal iot-specific televisions can be used to secretly record conversations .
cybercriminals who initiated the attack managed to commandeer a large number of internet-connected devices in current use .
cia documents revealed that microwave ovens can spy on you - maybe if you personally don't suffer the consequences of the sub-par security of the iot .

Internet of Things (IoT) security breaches have been dominating the headlines lately . WikiLeaks's trove of CIA documents revealed that internet-connected televisions can be used to secretly record conversations . Trump's advisor Kellyanne Conway believes that microwave ovens can spy on you - maybe she was referring to microwave cameras which indeed can be used for surveillance . And don't delude yourself that you are immune to IoT attacks , with 96 % of security professionals responding to a new survey expecting an increase in IoT breaches this year . Even if you personally don't suffer the consequences of the sub-par security of the IoT , your connected gadgets may well be unwittingly cooperating with criminals . Last October , Internet service provider Dyn came under an attack that disrupted access to popular websites . The cybercriminals who initiated the attack managed to commandeer a large number of internet-connected devices (mostly DVRs and cameras) to serve as their helpers . As a result , cybersecurity expert Bruce Schneier has called for government regulation of the IoT , concluding that both IoT manufacturers and their customers don't care about the security of the 8.4 billion internet-connected devices in current use . Whether because of government regulation or good old-fashioned self-interest , we can expect increased investment in IoT security technologies . In its recently-released TechRadar report for security and risk professionals , Forrester Research discusses the outlook for the 13 most relevant and important IoT security technologies , warning that " there is no single , magic security bullet that can easily fix all IoT security issues . " Based on Forrester's analysis , here's my list of the 6 hottest technologies for IoT security : IoT network security : Protecting and securing the network connecting IoT devices to back-end systems on the internet . IoT network security is a bit more challenging than traditional network security because there is a wider range of communication protocols , standards , and device capabilities , all of which pose significant issues and increased complexity . Key capabilities include traditional endpoint security features such as antivirus and antimalware as well as other features such as firewalls and intrusion prevention and detection systems . Sample vendors : Bayshore Networks , Cisco , Darktrace , and Senrio . IoT authentication : Providing the ability for users to authenticate an IoT device , including managing multiple users of a single device (such as a connected car) , ranging from simple static password/pins to more robust authentication mechanisms such as two-factor

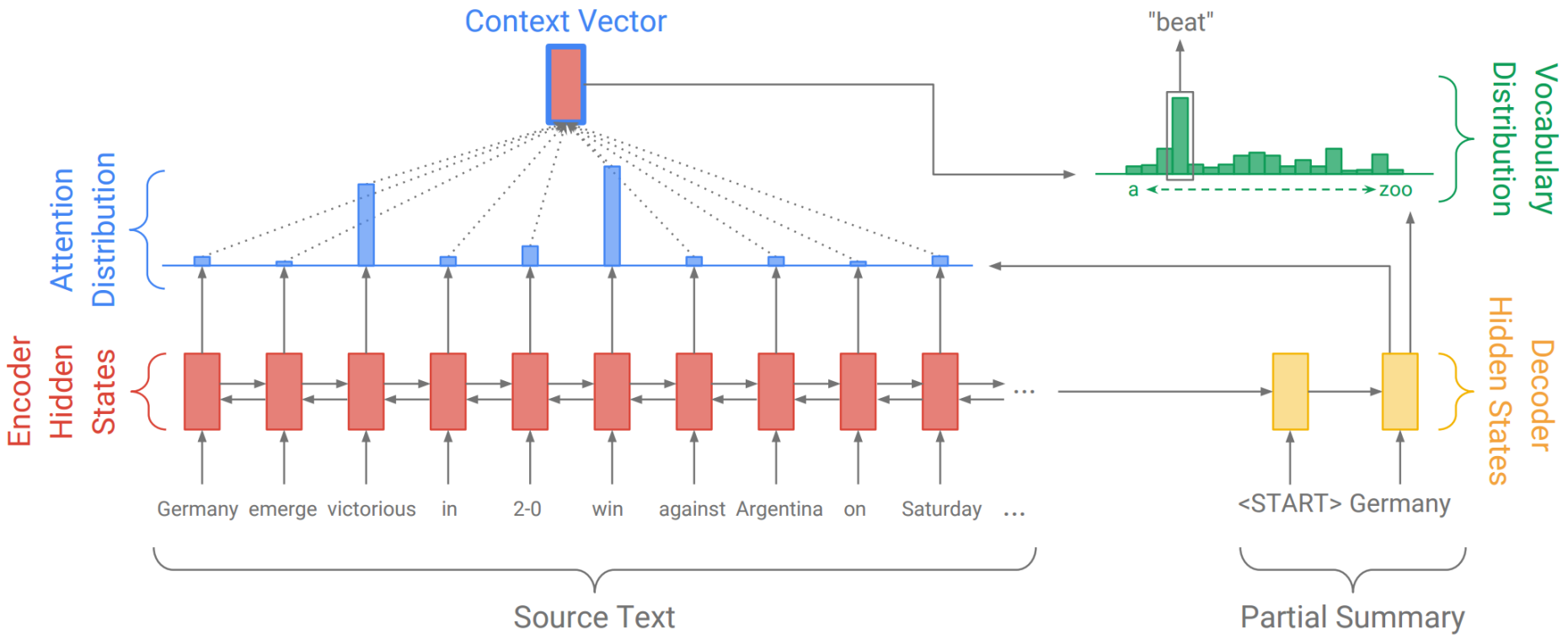
Summarization Results

The bottleneck is no longer access to information; now it's our ability to keep up.
AI can be trained on a variety of different types of texts and summary lengths.
A model that can generate long, coherent, and meaningful summaries remains an open research problem.

The last few decades have witnessed a fundamental change in the challenge of taking in new information. The bottleneck is no longer access to information; now it's our ability to keep up. We all have to read more and more to keep up-to-date with our jobs, the news, and social media. We've looked at how AI can improve people's work by helping with this information deluge and one potential answer is to have algorithms automatically summarize longer texts. Training a model that can generate long, coherent, and meaningful summaries remains an open research problem. In fact, generating any kind of longer text is hard for even the most advanced deep learning algorithms. In order to make summarization successful, we introduce two separate improvements: a more contextual word generation model and a new way of training summarization models via reinforcement learning (RL). The combination of the two training methods enables the system to create relevant and highly readable multi-sentence summaries of long text, such as news articles, significantly improving on previous results. Our algorithm can be trained on a variety of different types of texts and summary lengths. In this blog post, we present the main contributions of our model and an overview of the natural language challenges specific to text summarization.

Similar ideas explored simultaneously by Abi et al.

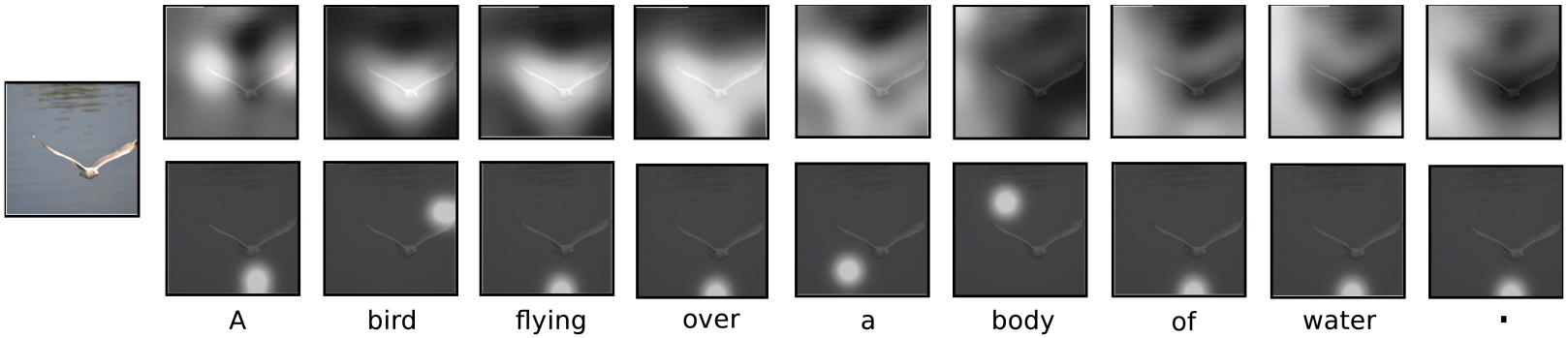
- GetTo The Point: Summarization with Pointer-Generator Networks, Abigail See, Peter J. Liu, Christopher Manning, 2



Blog post: http://www.abigailsee.com/2017/04/16/taming-rnns-for-better-summarization.html

Using attention for coverage

- Caption generation



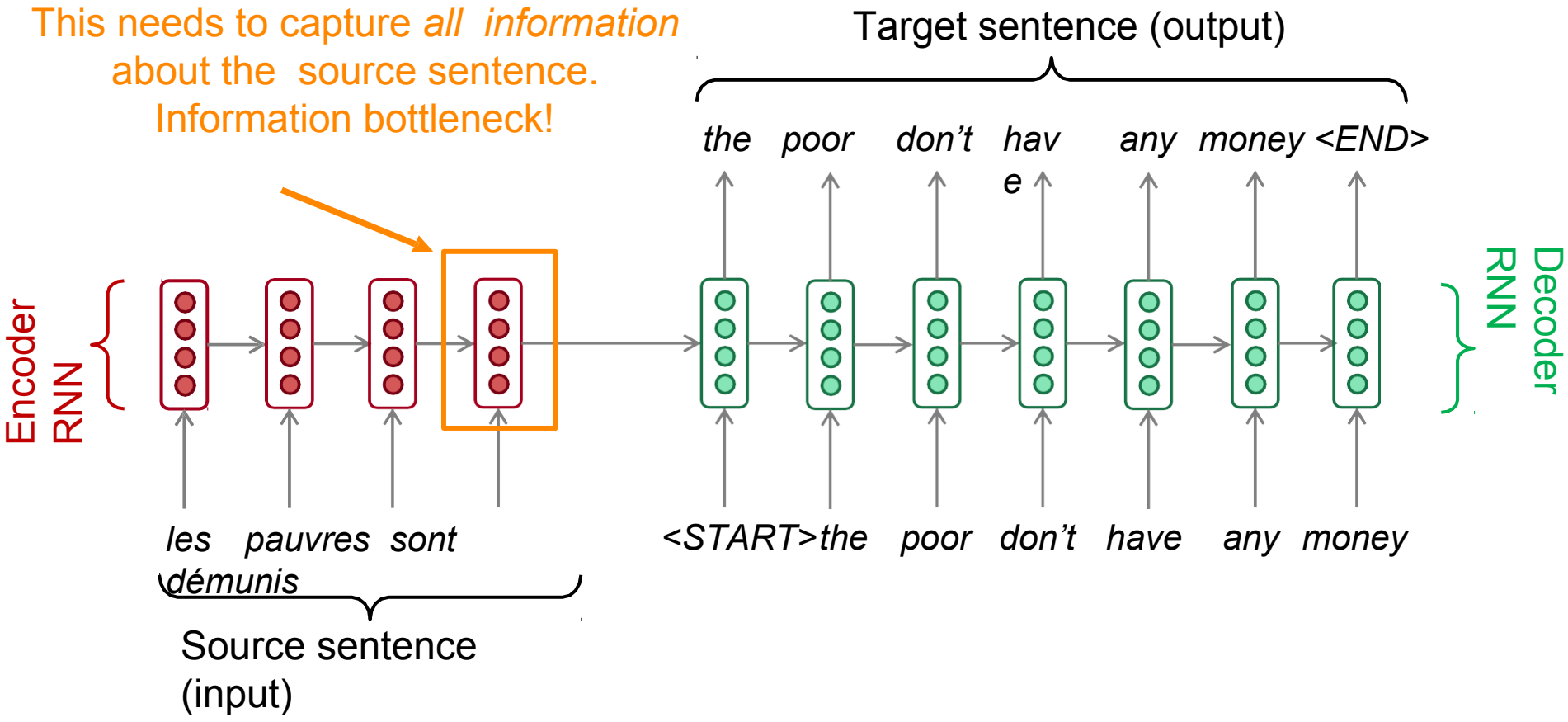
How to not miss an important image patch?

Xu, Ba, Kiros, Cho, Courville, Salakhutdinov, Zemel, Bengio. **Show Tell:Neural Image Caption Generation with Visual Attention**. ICM

Sequence-to-sequence: the bottleneck problem

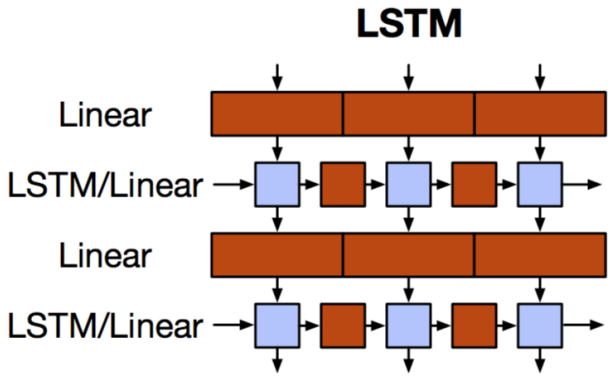
Encoding of the source sentence.

This needs to capture *all information* about the source sentence.
Information bottleneck!



Problems with RNNs = Motivation for Transformers

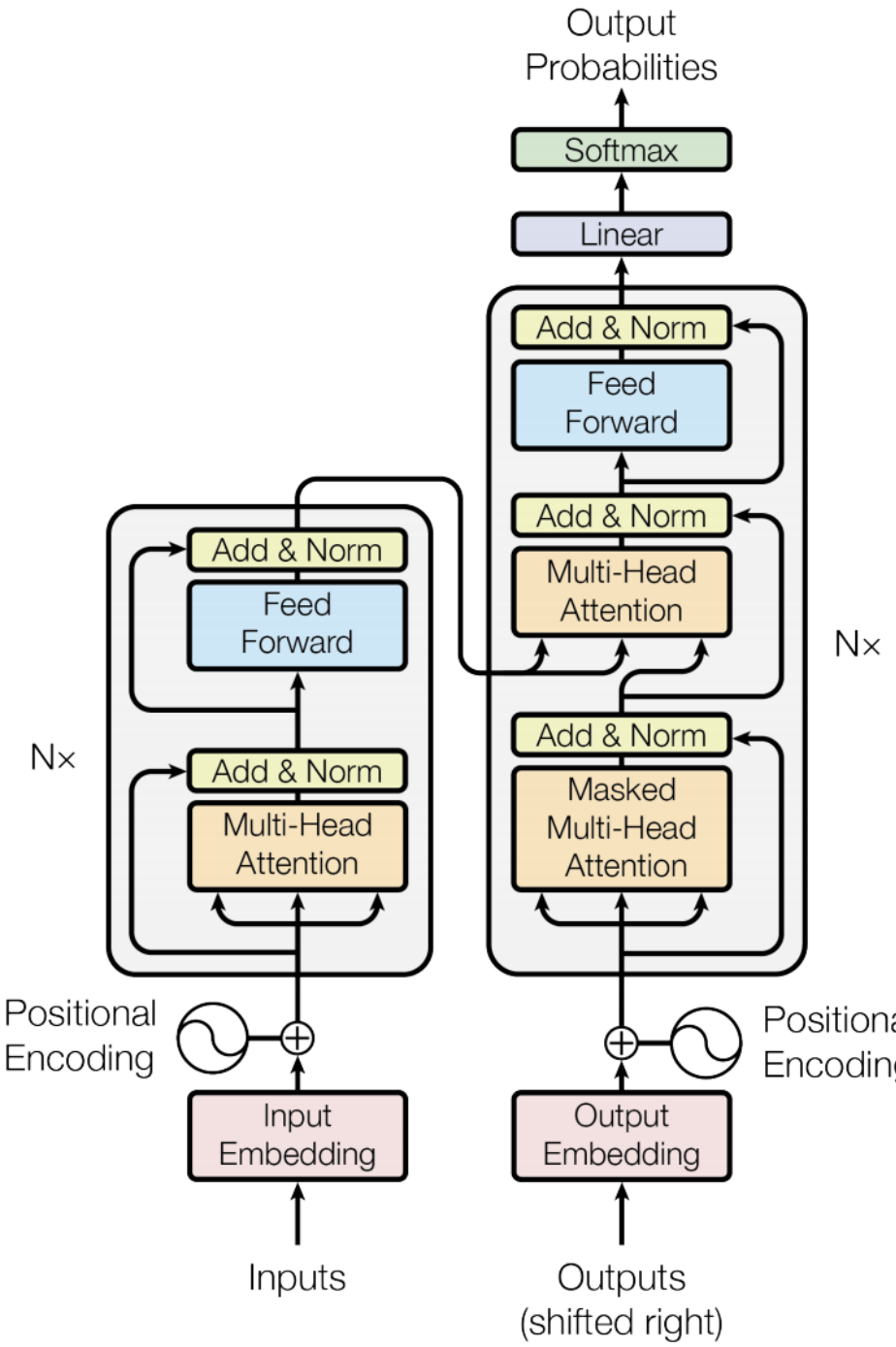
- Sequential computation prevents parallelization



- Despite GRUs and LSTMs, RNNs still need attention mech to deal with long range dependencies – path length for c dependent computation between states grows with sequence
- But if attention gives us access to any state… maybe we don't need the RNN? 🤔

Transformer Overview

- Sequence-to-sequence
- Encoder-Decoder
- Task: machine translation with parallel corpus
- Predict each translated word
- Final cost/error function is standard cross-entropy error on top of a softmax classifier



This and related figures frompaper: https://arxiv.org/pdf/1706.03762.pdf

Transformer Paper

- Attention Is All You Need [2017]
- by Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, LukaszKaiser, Illia Polosukhin
- Equal contribution. Listing order is random. Jakob proposed replacing RN attention and started the effort to evaluate this idea.Ashish, with Illia, designe implemented the first Transformer models and has been crucially involved in aspect of this work. Noam proposed scaled dot-product attention, m attention and the parameter-free position representation and became the ot person involved in nearly every detail. Niki designed, implemented, tuned evaluated countless model variants in our original codebase and tensor2tens also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively ac our research.

Transformer Basics

- Let's define the basic building blocks of transformer networks first: new attention layers!

Dot-Product Attention (Extending our previous def

- Inputs: a query q and a set of key-value (k-v) pairs to
- Query, keys, values, and output are all vectors

- Output is weighted sum of values, where
- Weight of each value is computed by an inner product o and corresponding key
- Queries and keys have same dimensionality d

$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i$$

Dot-Product Attention – Matrix notation

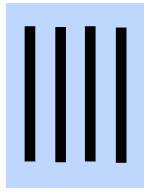
- When we have multiple queries q, we stack them in a matrix Q:

$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i$$

- Becomes: $A(Q, K, V) = softmax(QK^T)V$

$[|Q| \quad x_k] \quad d \quad _k x \quad |K|][d \quad x \quad _v] \quad [|K| \quad x \quad d$

softmax row-wise



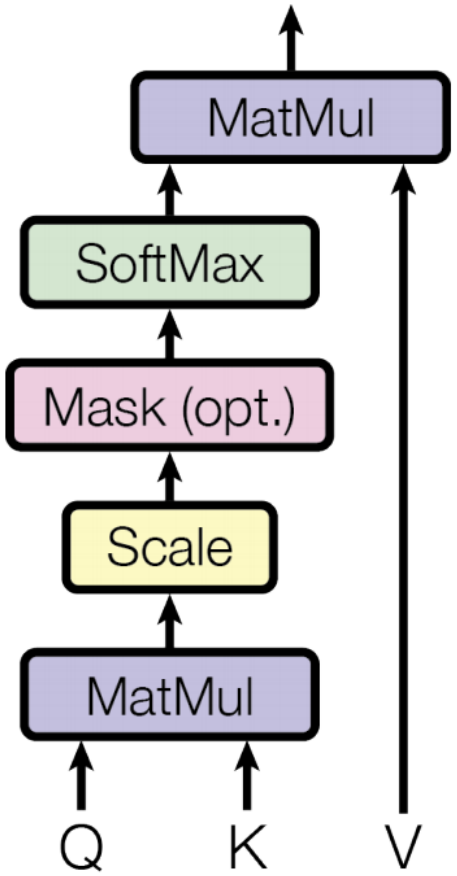
$= \quad [|Q| \quad _v]x \quad d$

Scaled Dot-Product Attention

- Problem: As d gets large, the variance of q k increases values inside the softmax get large the softmax gets very peaked--> hence its gradient gets smaller.

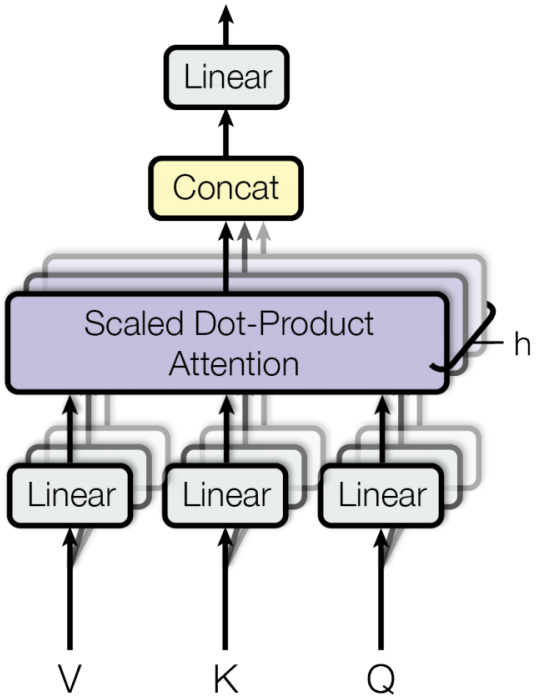
- Solution: Scale by length of query/key vectors:

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



Self-attention and Multi-head attention

- The input word vectorscould be the queries, keys and
- In other words: the word vectors themselves select each other
- Word vector stack = Q = K = V
- Problem: Only one way for words to interact with one-another
- Solution: Multi-head attention
- First map Q, K, V into h many dimensional spaces via W matrices
- Then apply attention, then concaten Outputs and pipe through linear layer

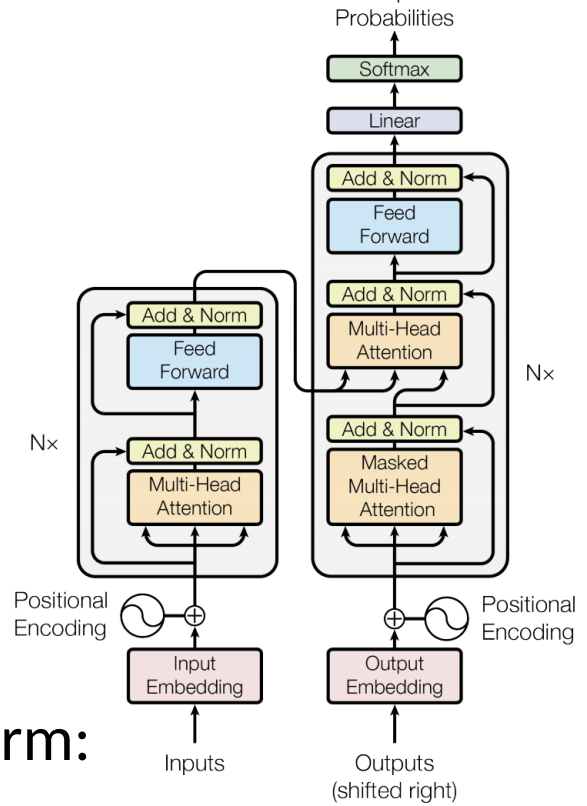


$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Complete transformer block

- Each block has two "sublayers"

1. Multihead attention
2. 2 layer feed-forward Nnet (with relu)



Each of these two steps also has:

Residual (short-circuit) connection and LayerNorm:

LayerNorm(x + Sublayer(x))

Layernorm changes input to have mean 0 and variance 1, per layer and per training point (and adds two more parameters)

$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \qquad \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^l - \mu^l\right)^2}$$

$$h_i = f\left(\frac{g_i}{\sigma_i}\left(a_i - \mu_i\right) + b_i\right)$$

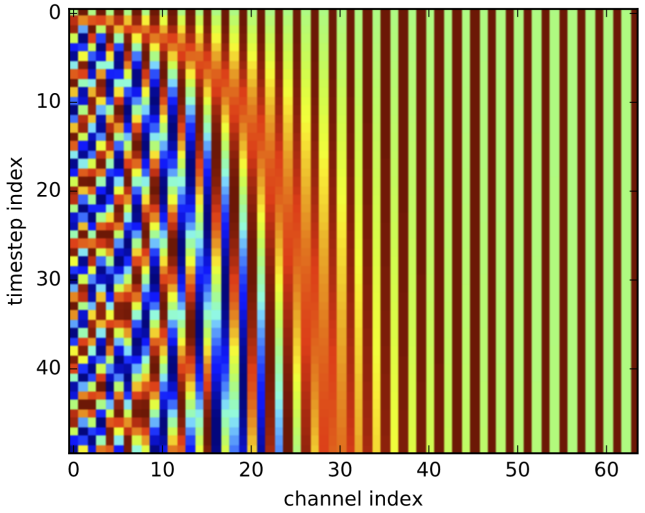
Layer Normalization by Ba, Kiros and Hinton, https://arxiv.org/pdf/1607.06450.pdf

Encoder Input

- Actual word representations are byte-pair encodings

- Also added is a positional encoding so same words at different locations have different overall representations:

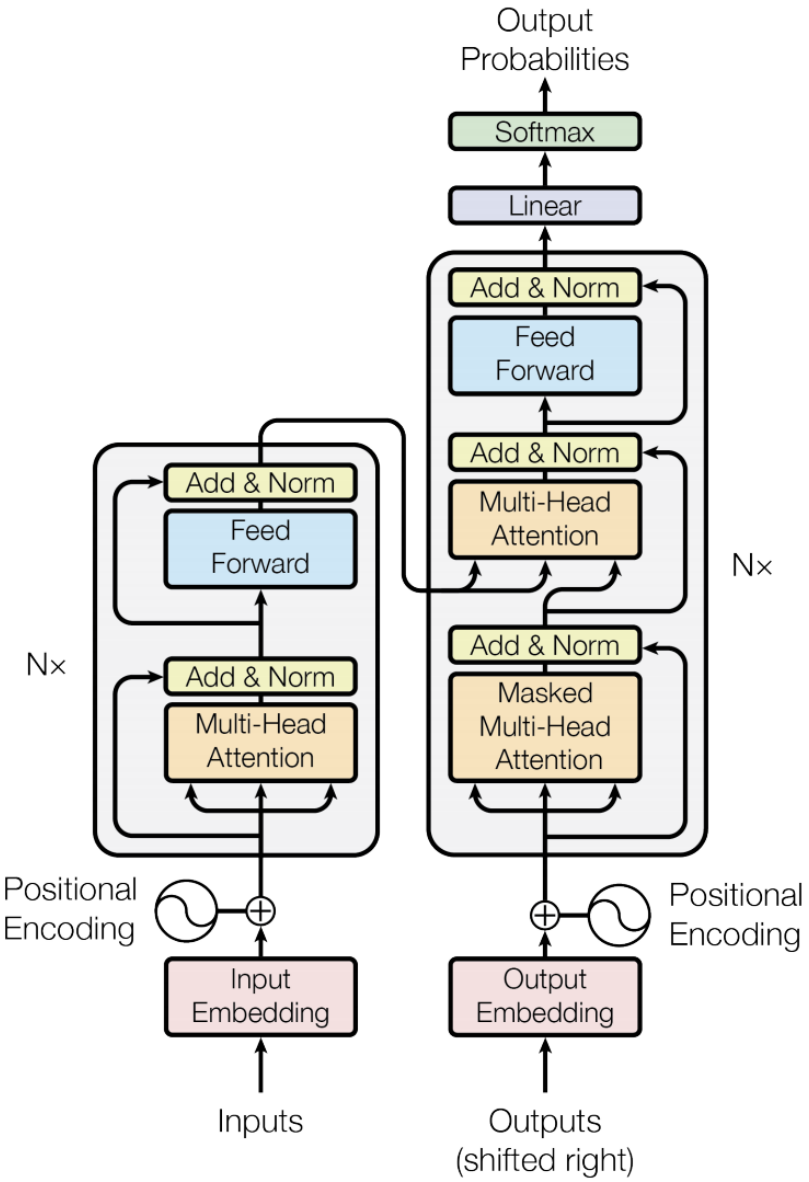
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$



Complete Encoder

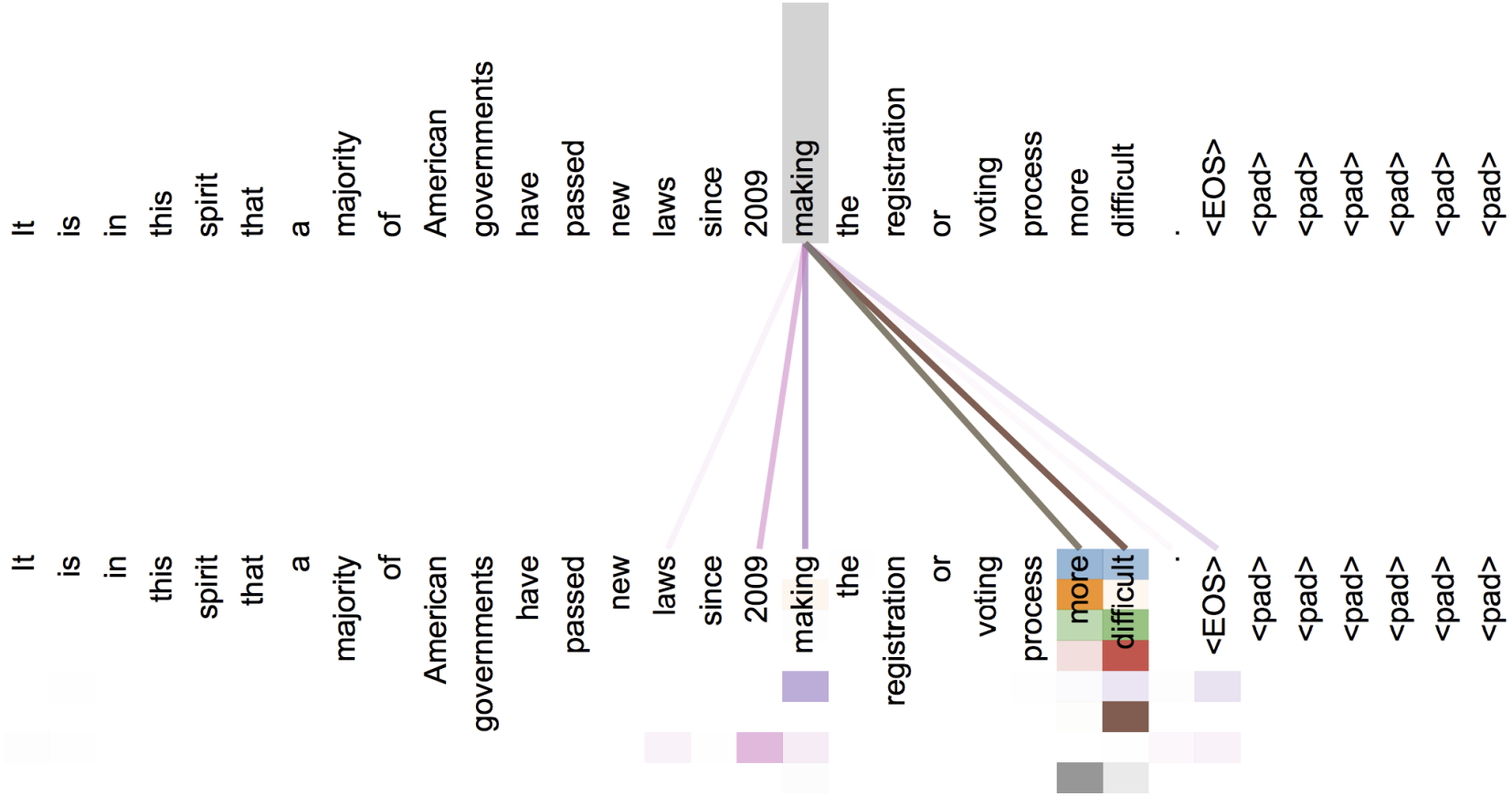
- For encoder, at each block, we use the same Q, K and V from the previous layer

- Blocks are repeated 6 times

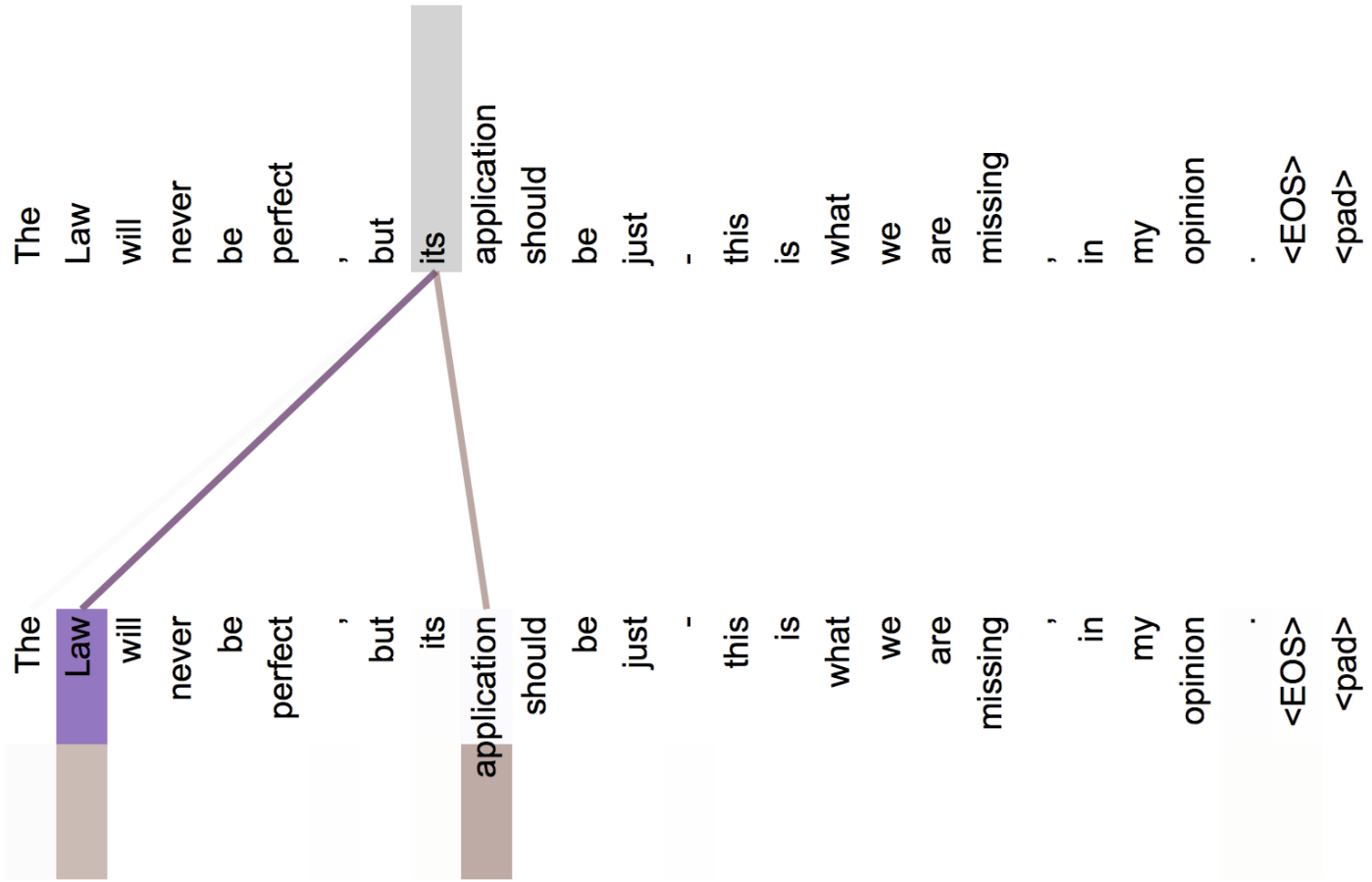


Attention visualization in layer 5

- Words start to payattention to other words in sensible ways



Attention visualization: Implicit anaphora resolution

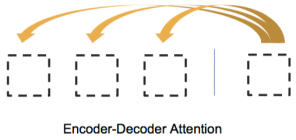


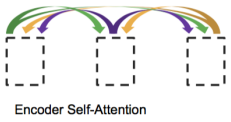
In 5th Layer. Isolated attentions from just the word 'its' for attention heads 5 and 6. Note that the attentions are very sharp for this word.

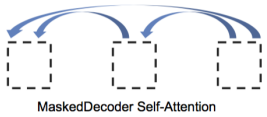
Transformer Decoder

- 2 sublayer changes in decoder
- Masked decoder self-attention on previously generated outputs:

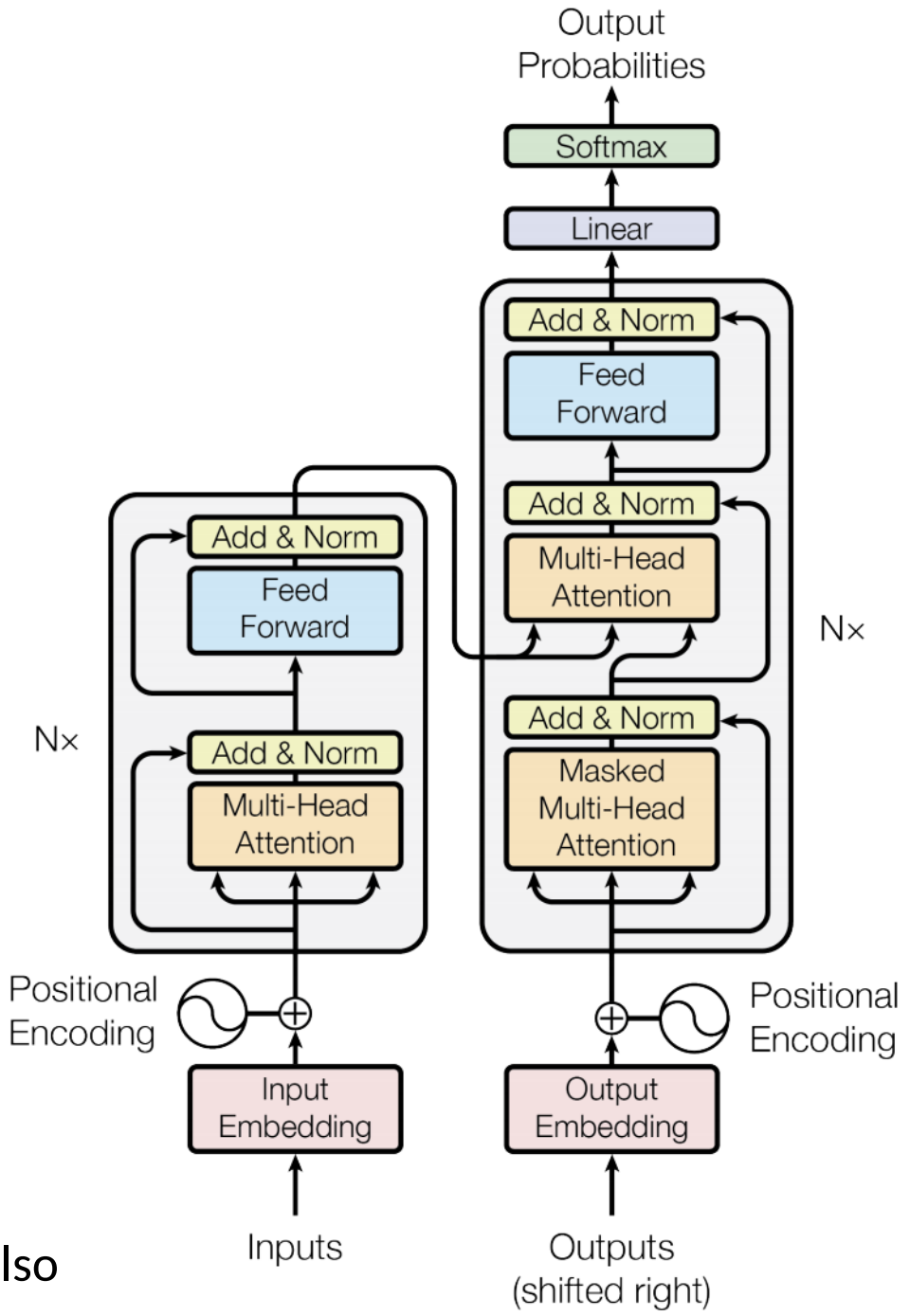
- Encoder-Decoder Attention, where queries come from previous decoder layer and keys and values come from output of encoder







- Blocks repeated 6 times also



Tips and tricks of the Transformer

Details in paper and later lectures:

- Byte-pair encodings
- Checkpoint averaging
- ADAM optimizer with learning rate changes
- Dropout during training at every layer justbefore adding residual
- Label smoothing
- Auto-regressive decoding with beam search and length penalties

- Overall, they are hard to optimize and unlike LSTMs don't usually work out of the box and don't play well yet with other building blocks on tasks.

Experimental Results for MT

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**	$2.3 \cdot 10^{19}$	

Experimental Results for Parsing

Parser	Training	WSJ 23 F1
Vinyals & Kaiser el al. (2014) [37]	WSJ only, discriminative	88.3
Petrov et al. (2006) [29]	WSJ only, discriminative	90.4
Zhu et al. (2013) [40]	WSJ only, discriminative	90.4
Dyer et al. (2016) [8]	WSJ only, discriminative	91.7
Transformer (4 layers)	WSJ only, discriminative	91.3
Zhu et al. (2013) [40]	semi-supervised	91.3
Huang & Harper (2009) [14]	semi-supervised	91.3
McClosky et al. (2006) [26]	semi-supervised	92.1
Vinyals & Kaiser el al. (2014) [37]	semi-supervised	92.1
Transformer (4 layers)	semi-supervised	92.7
Luong et al. (2015) [23]	multi-task	93.0
Dyer et al. (2016) [8]	generative	93.3