자연어 처리 DAY 3
Sequence to Sequence with Attention

Jaegul Choo
Associate Professor, Graduate School of AI, KAIST

boostcamp AI Tech

DAVIAN
Data and Visual Analytics Lab

1.

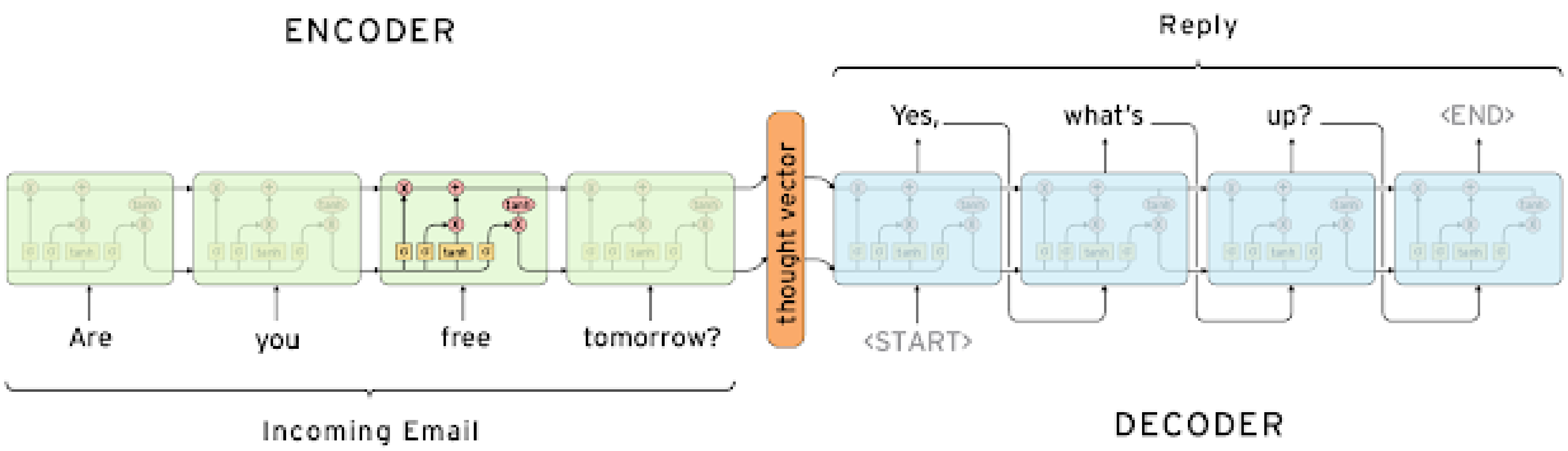
Seq2Seq with attention

Encoder-decoder architecture

Attention mechanism

Seq2Seq Model

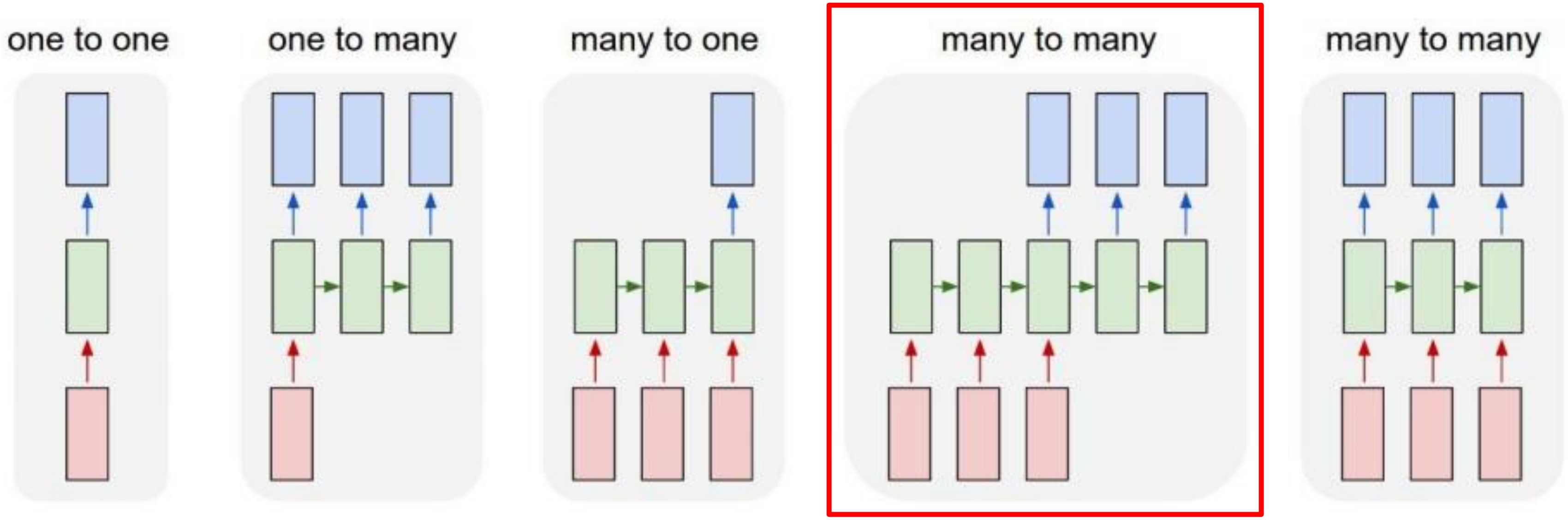
- It takes a sequence of words as input and gives a sequence of words as output
- It composed of an encoder and a decoder



Sequence to sequence learning with neural networks, ICML'14

Recall: Types of RNNs

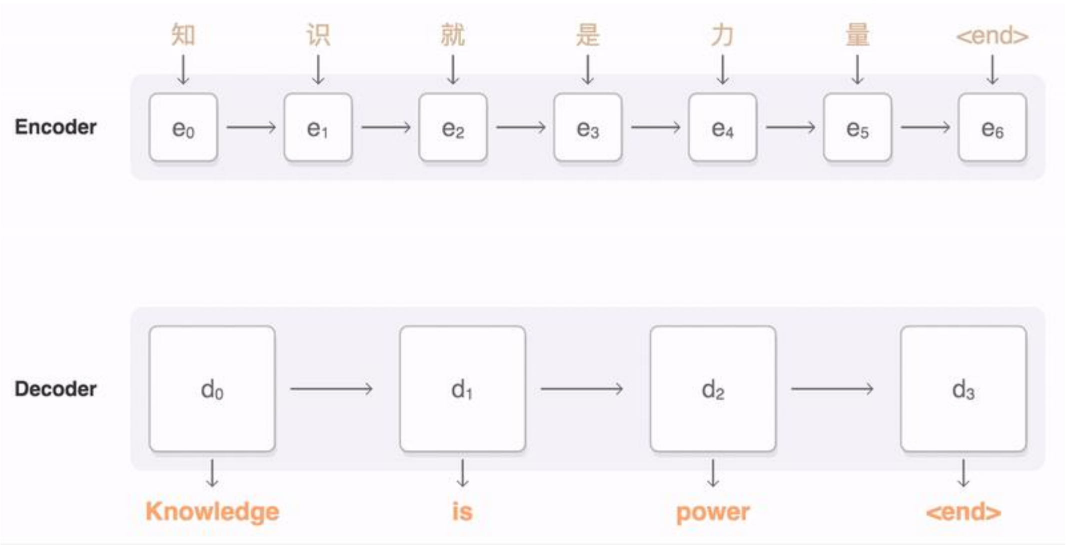
- **Sequence-to-sequence**
  - Machine Translation



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Seq2Seq Model with Attention

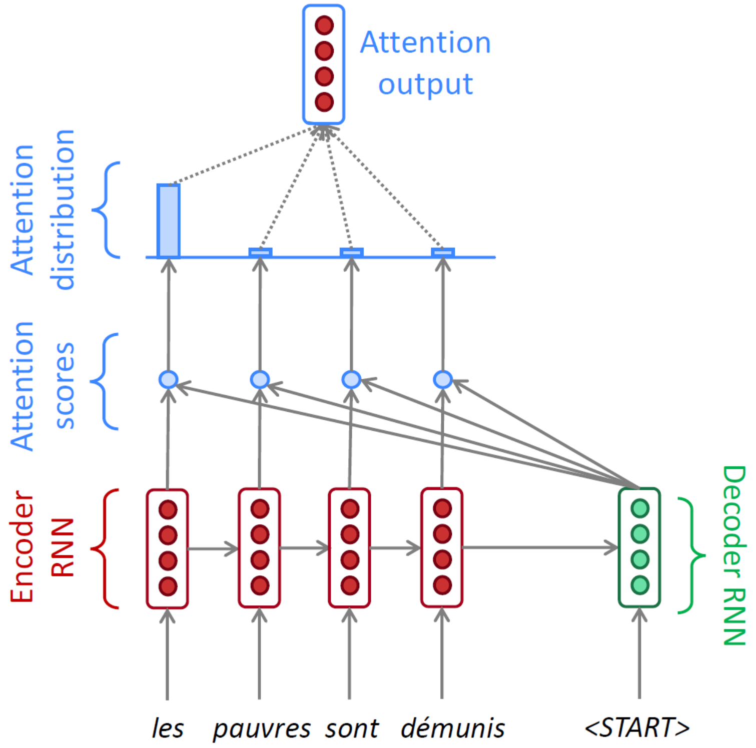
- Attention provides a solution to the bottleneck problem
- Core idea: At each time step of the decoder, focus on a particular part of the source sequence



https://google.github.io/seq2seq/

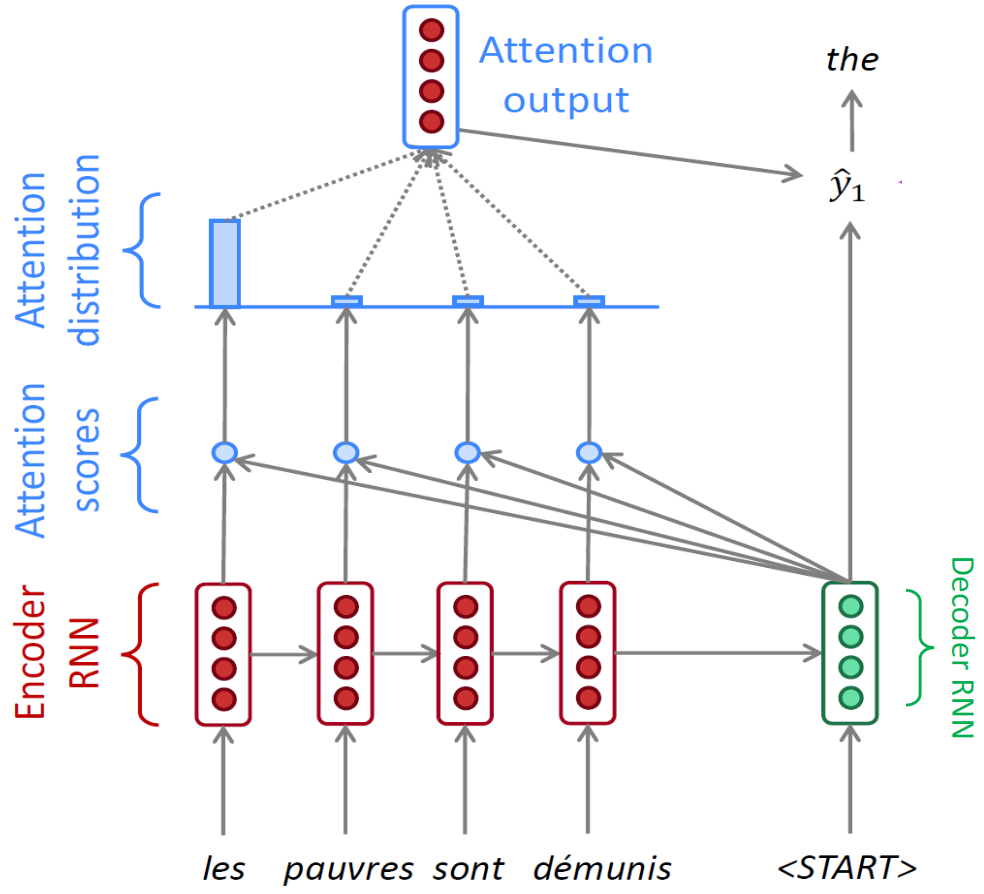
Seq2Seq Model with Attention

- 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

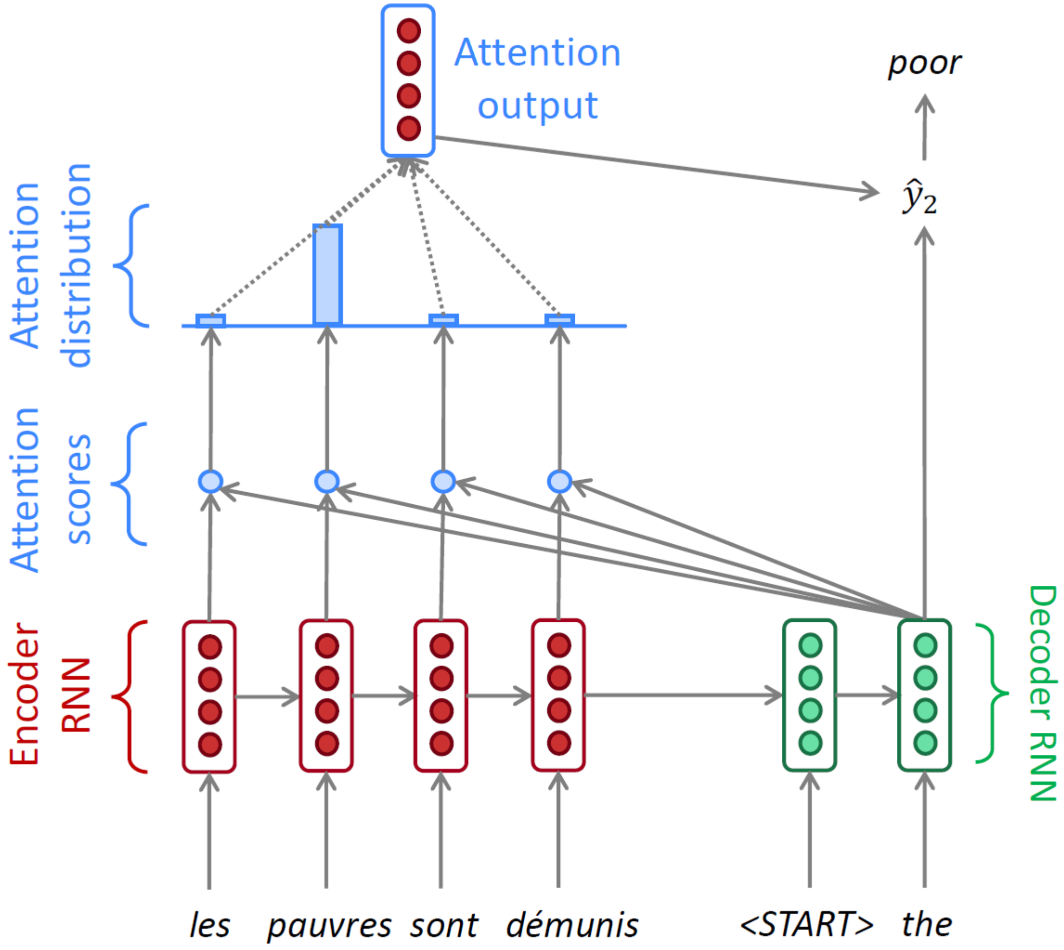


Seq2Seq Model with Attention

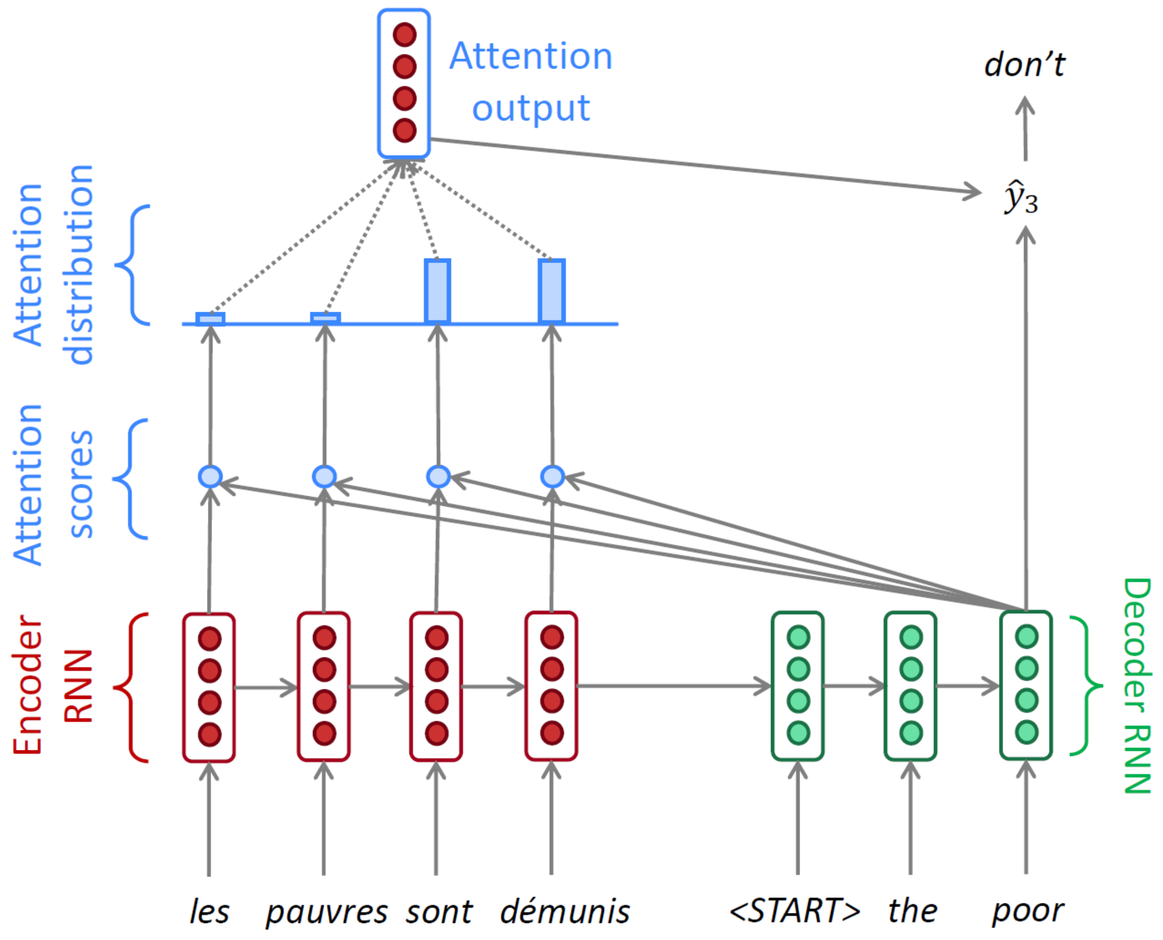
- Concatenate attention output with decoder hidden state, then use to compute $\widehat{y_1}$ as before



Seq2Seq Model with Attention

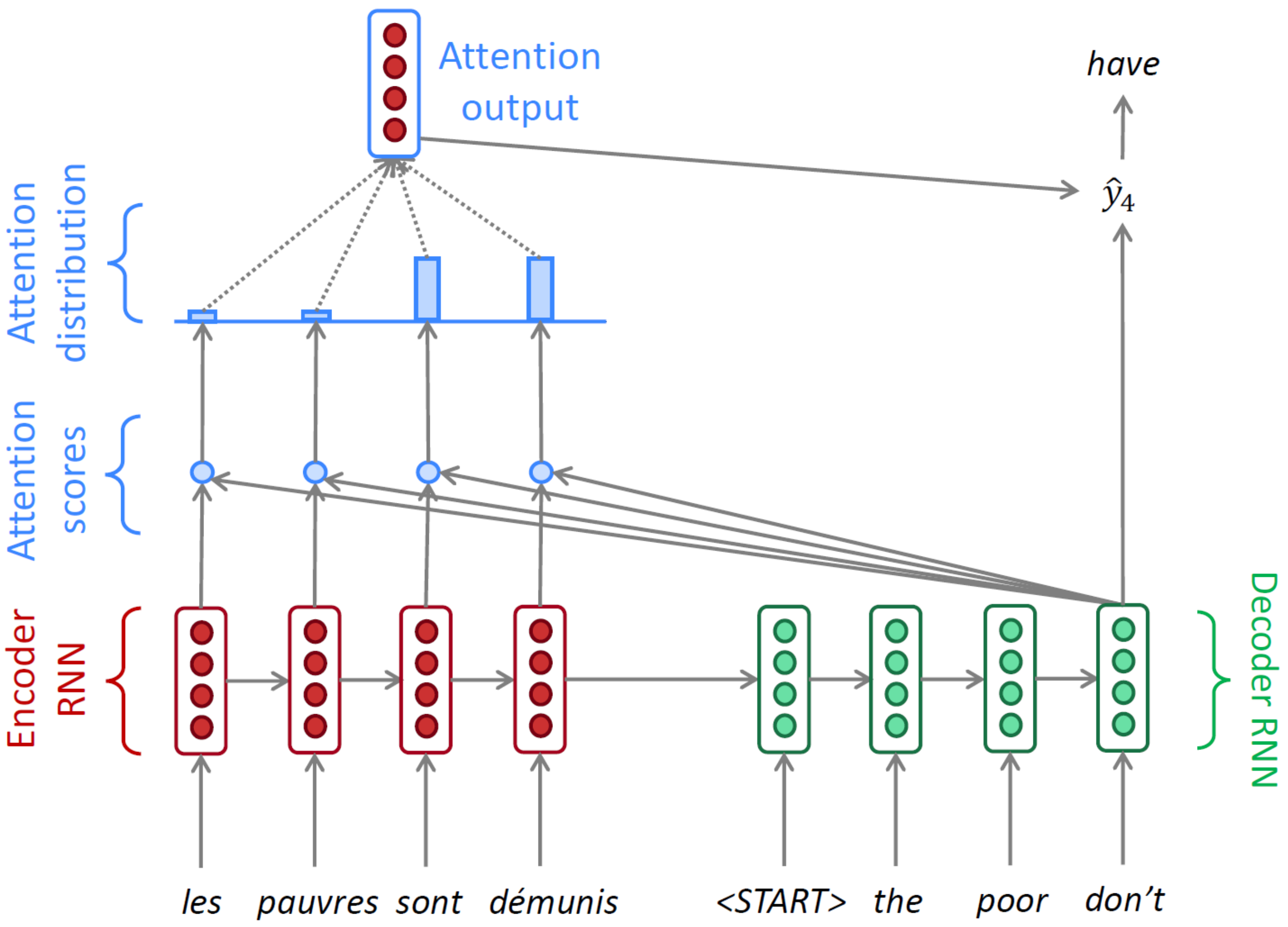


Seq2Seq Model with Attention



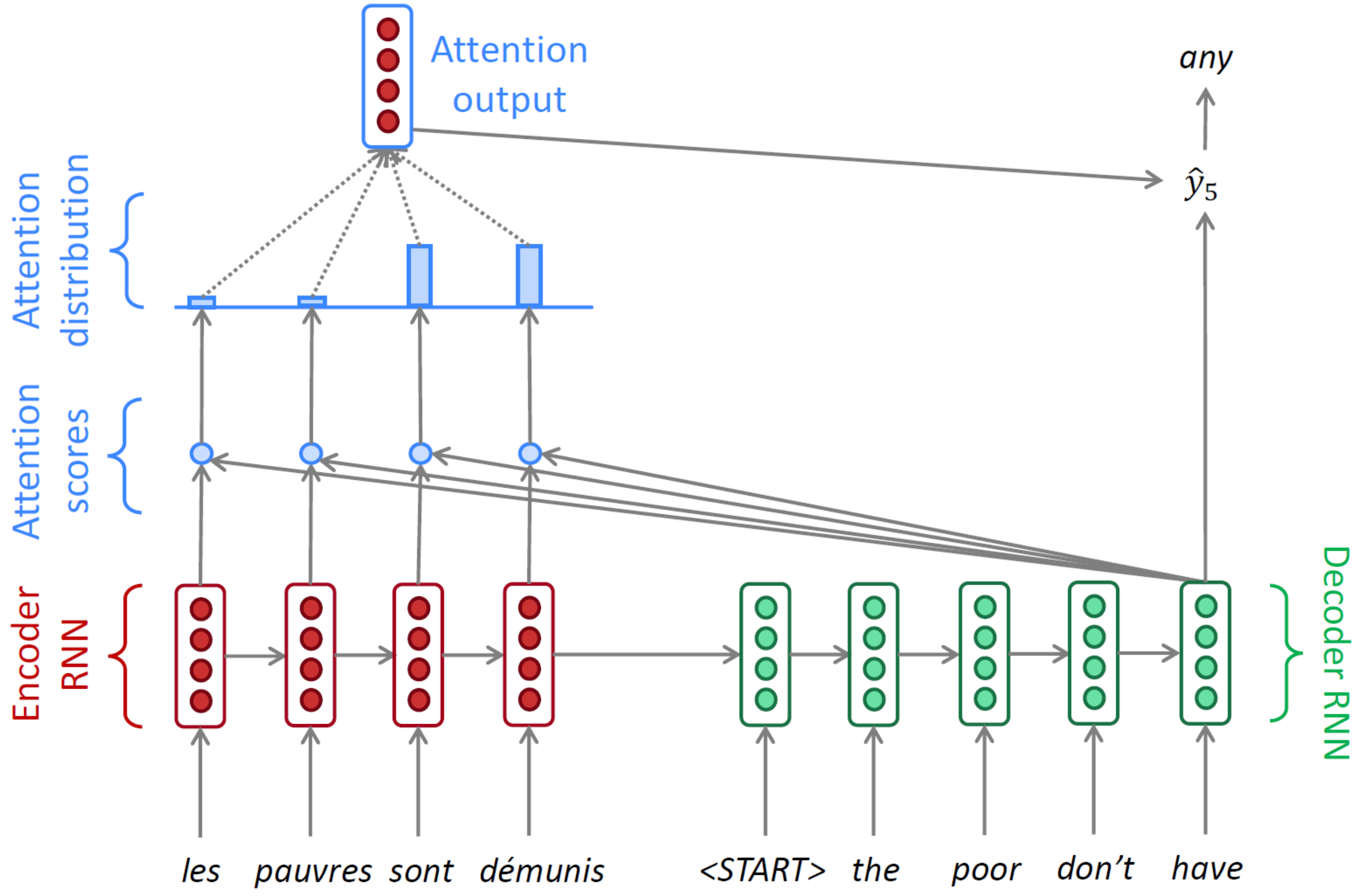
Seq2Seq Model with Attention

Encoder-decoder architecture, Attention mechanism

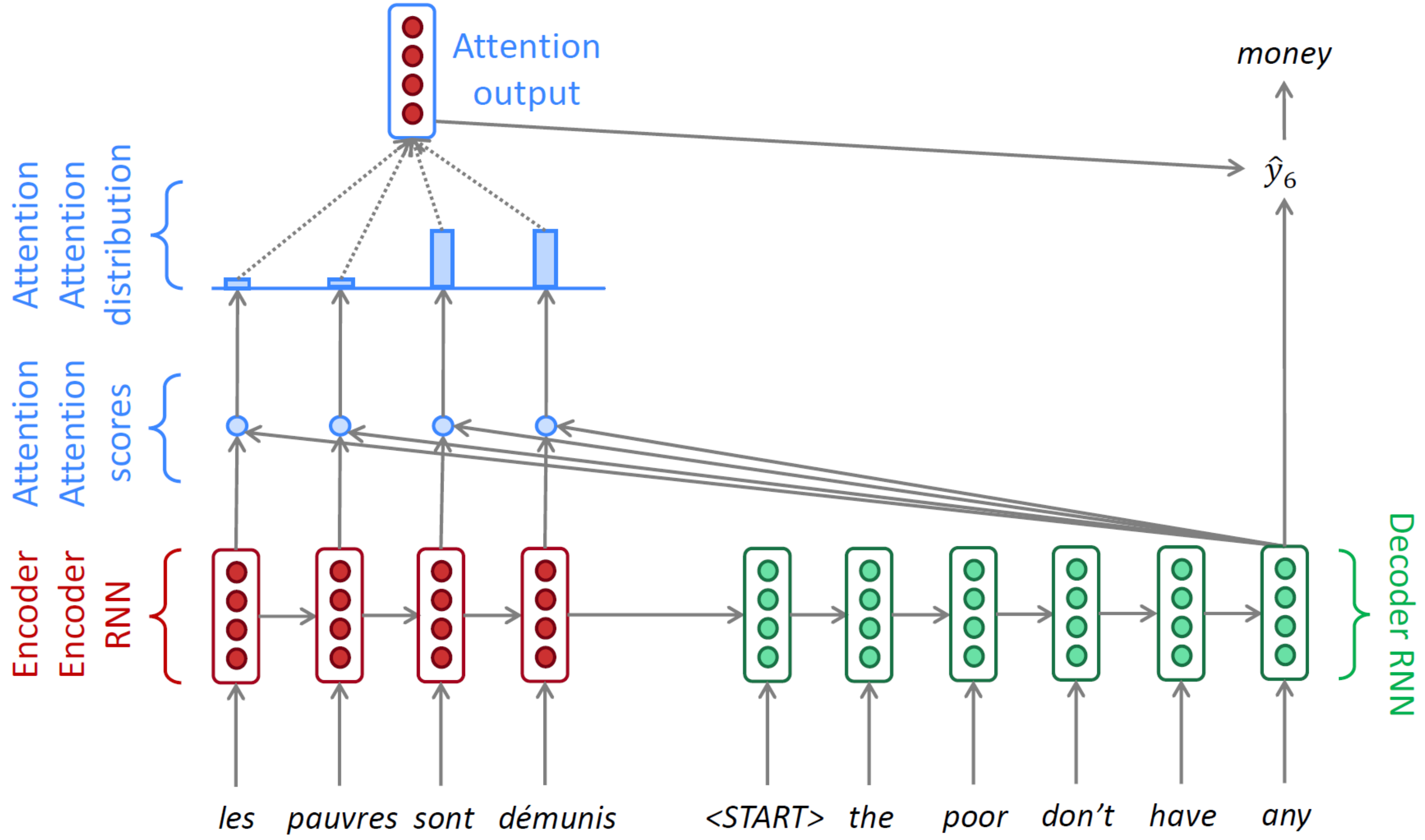


boostcamp AI Tech

Seq2Seq Model with Attention

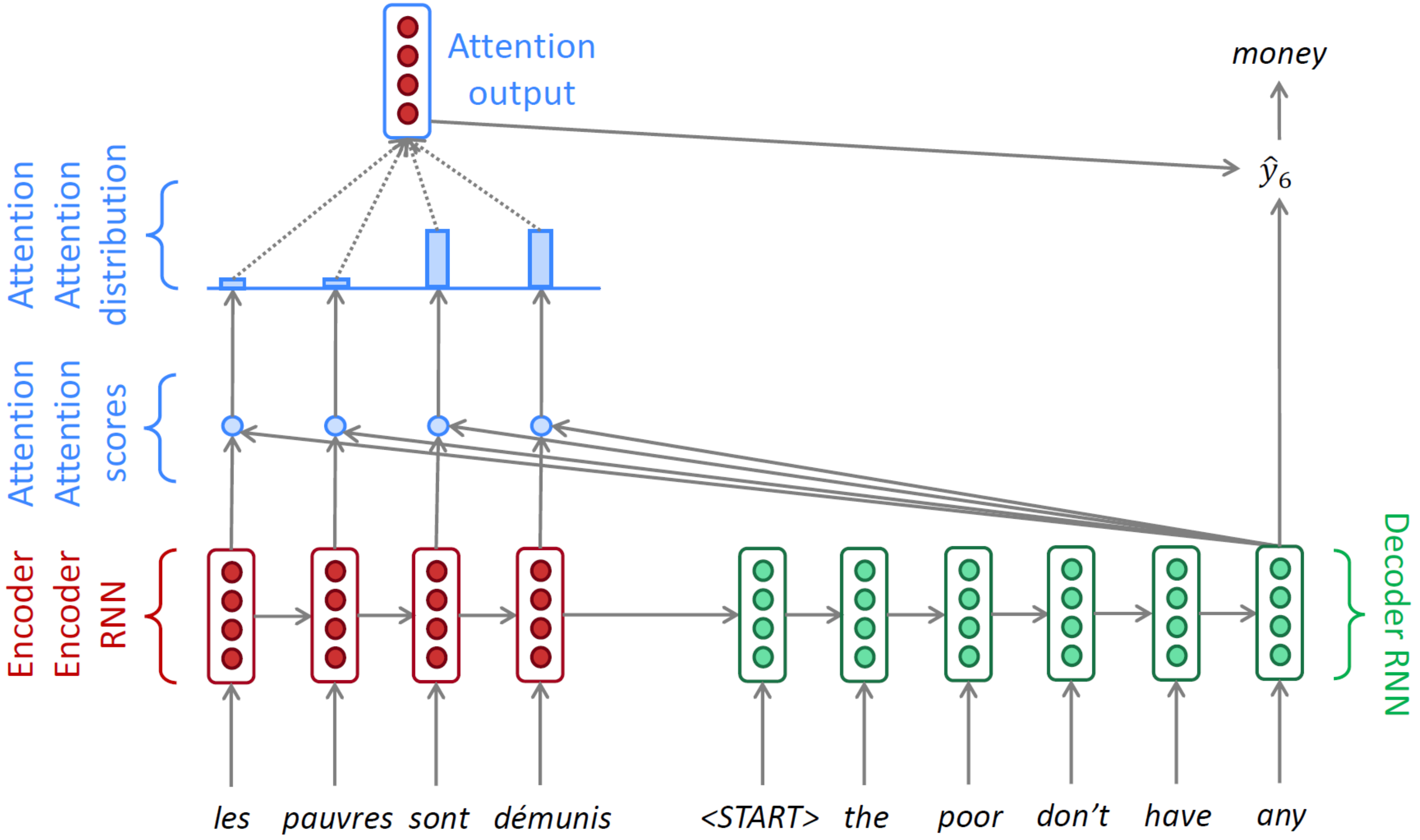


Seq2Seq Model with Attention



Seq2Seq Model with Attention

Encoder-decoder architecture, Attention mechanism



Different Attention Mechanisms

- **Luong** attention: they get the decoder hidden state at time t, then calculate attention scores, and from that get the context vector which will be concatenated with hidden state of the decoder and then predict the output.

$$\mathrm{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W_a} \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

- **Bahdanau** attention: At time t, we consider the hidden state of the decoder at time $t-1$. Then we calculate the alignment, context vectors as above. But then we concatenate this context with hidden state of the decoder at time $t-1$. So before the softmax, this concatenated vector goes inside a LSTM unit.
- **Luong** has different types of alignments. Bahdanau has only a concat-score alignment model.

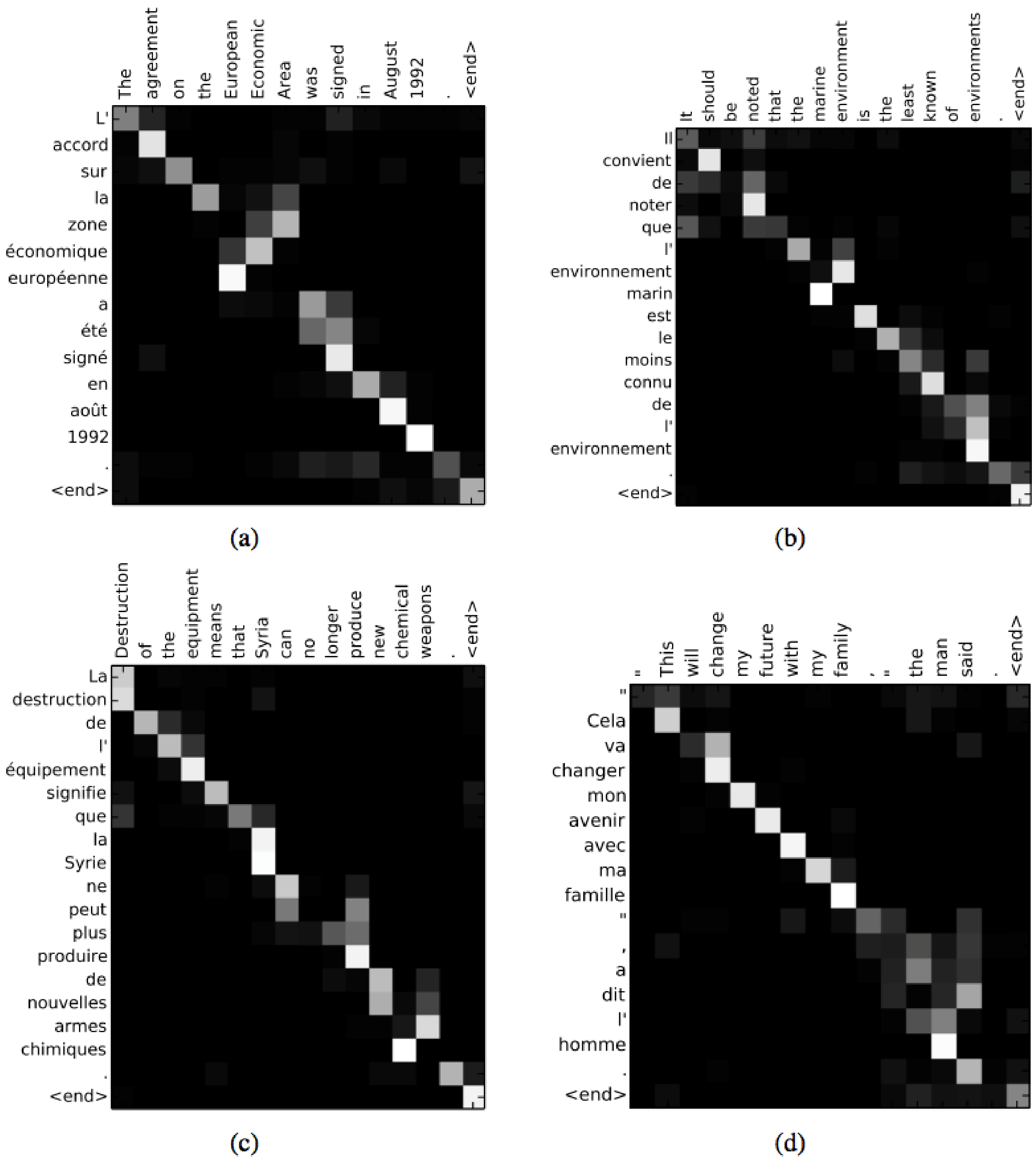
Effective Approaches to Attention-based Neural Machine Translation, EMNLP 2015

Attention is Great!

- Attention significantly improves NMT performance
  - It is useful to allow the decoder to focus on particular parts of the source
- Attention solves the bottleneck problem
  - Attention allows the decoder to look directly at source; bypass the bottleneck
- Attention helps with vanishing gradient problem
  - Provides a shortcut to far-away states
- Attention provides some interpretability
  - By inspecting attention distribution, we can see what the decoder was focusing on
  - The network just learned alignment by itself

Attention Examples in Machine Translation

- It properly learns grammatical orders of words
- It skips unnecessary words such as an article



2.

Beam search

Greedy decoding

- **Greedy decoding has no way to undo decisions!**
  - Input: il a m'entarté (he hit me with a pie)

→ *he* ___

→ *he hit* ___

→ *he hit a* ___ (whoops, no going back now…)

- How can we fix this?

Exhaustive search

- Ideally, we want to find a (length T) translation y that maximizes
  - $P(y|x) = P(y_1|x)P(y_2|y_1,x)P(y_3|y_2,y_1,x)\dots P(y_T|y_1,\dots,y_{T-1},x) = \prod_1^T P(y_t|y_1,\dots,y_{t-1},x)$
- We could try computing all possible sequences y
  - This means that on each step t of the decoder, we are tracking V^t possible partial translations, where V is the vocabulary size
  - This $O(V^t)$ complexity is far too expensive!

Beam search

- Core idea: on each time step of the decoder, we keep track of the k most probable partial translations (which we call hypothese)
  - k is the beam size (in practice around 5 to 10)
- A hypothesis $y_1, \dots, y_t$ has a score of its log probability:

$$score(y_1, \dots, y_t) = \log P_{LM}(y_1, \dots, y_t | x) = \sum_{i=1}^{t} \log P_{LM}(y_i | y_1, \dots, y_{i-1}, x)$$

  – Scores are all negative, and a higher score is better

  – We search for high-scoring hypotheses, tracking the top k ones on each step

Beam search

- Beam search is not guaranteed to find a globally optimal solution.
- But it is much more efficient than exhaustive search!

Beam search: Example

- Beam size: k = 2

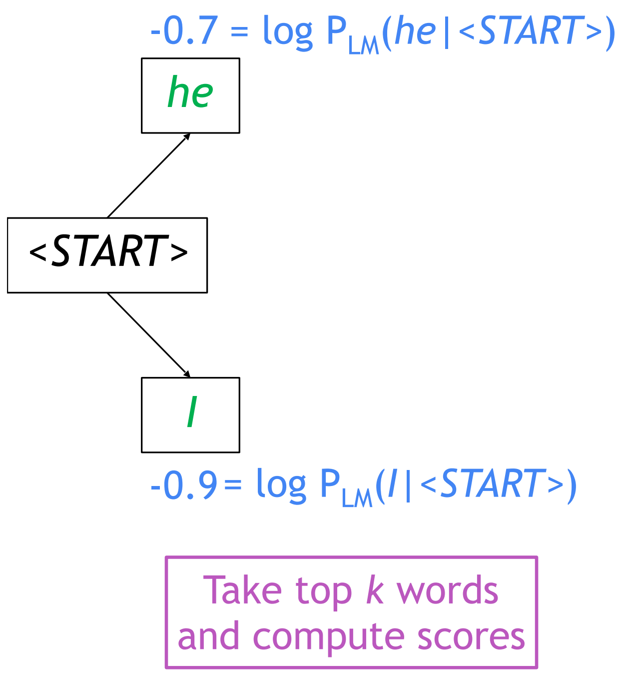
<START>

Calculate prob dist of next word

https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf

Beam search: Example

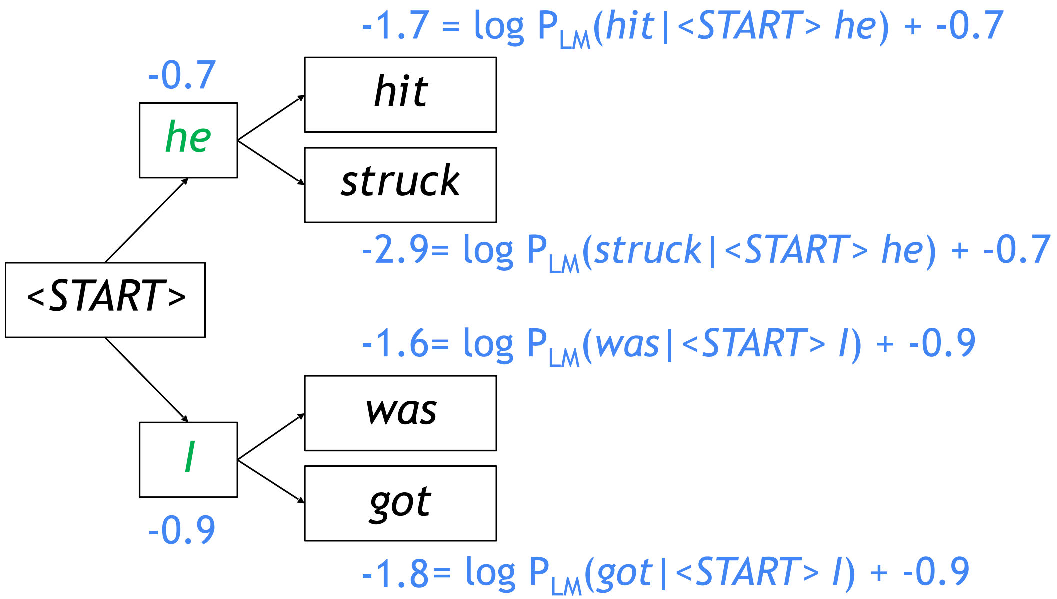
- Beam size: k = 2



https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf

Beam search: Example

- Beam size: k = 2

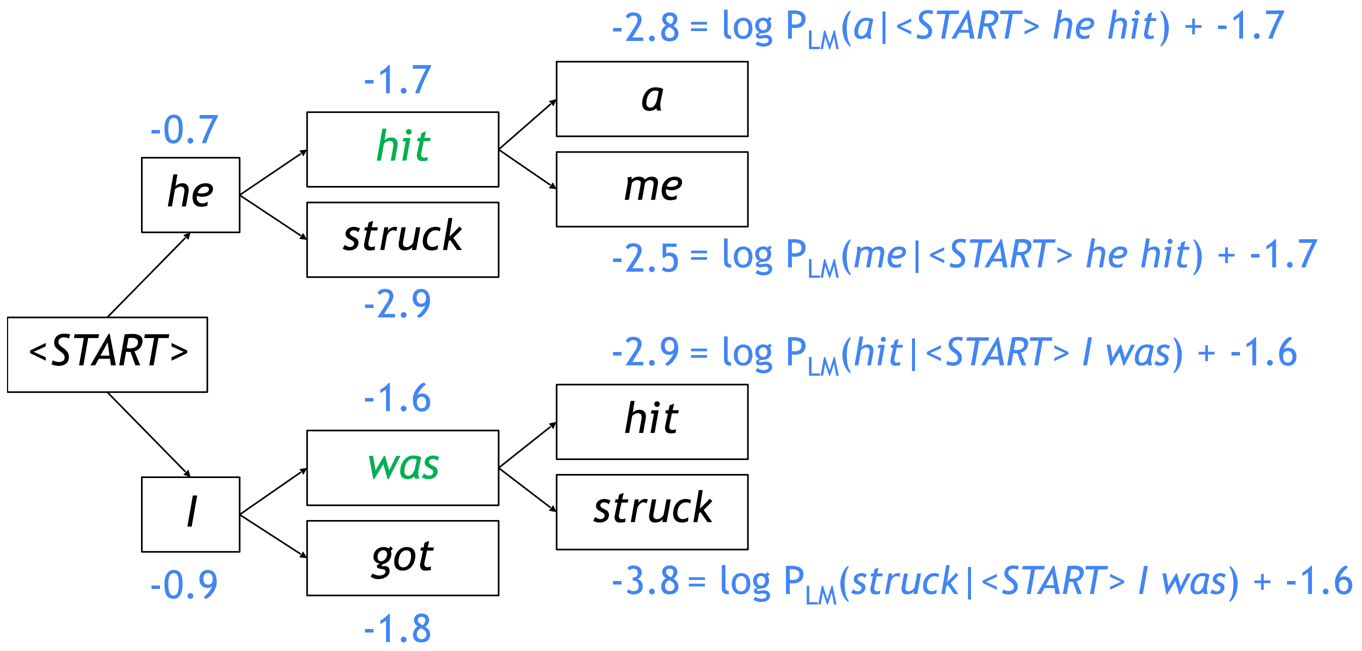


For each of the *k* hypotheses, find top *k* next words and calculate scores

https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf

Beam search: Example

- Beam size: k = 2

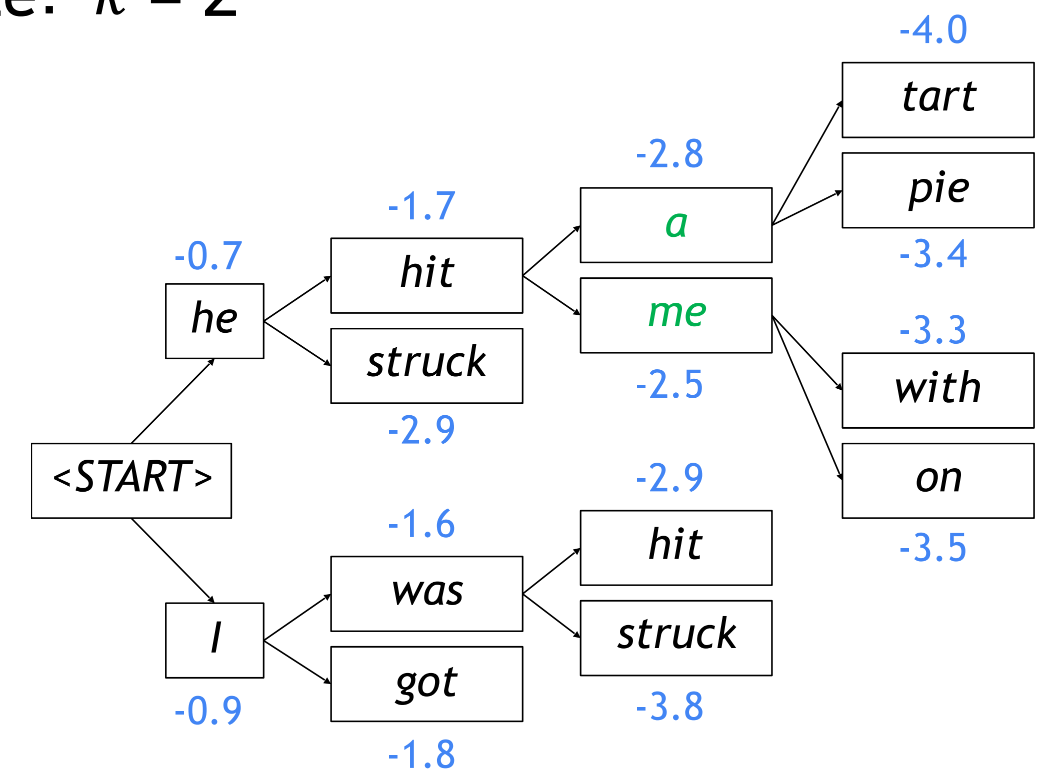


For each of the k hypotheses, find top k next words and calculate scores

https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf

Beam search: Example

- Beam size: k = 2

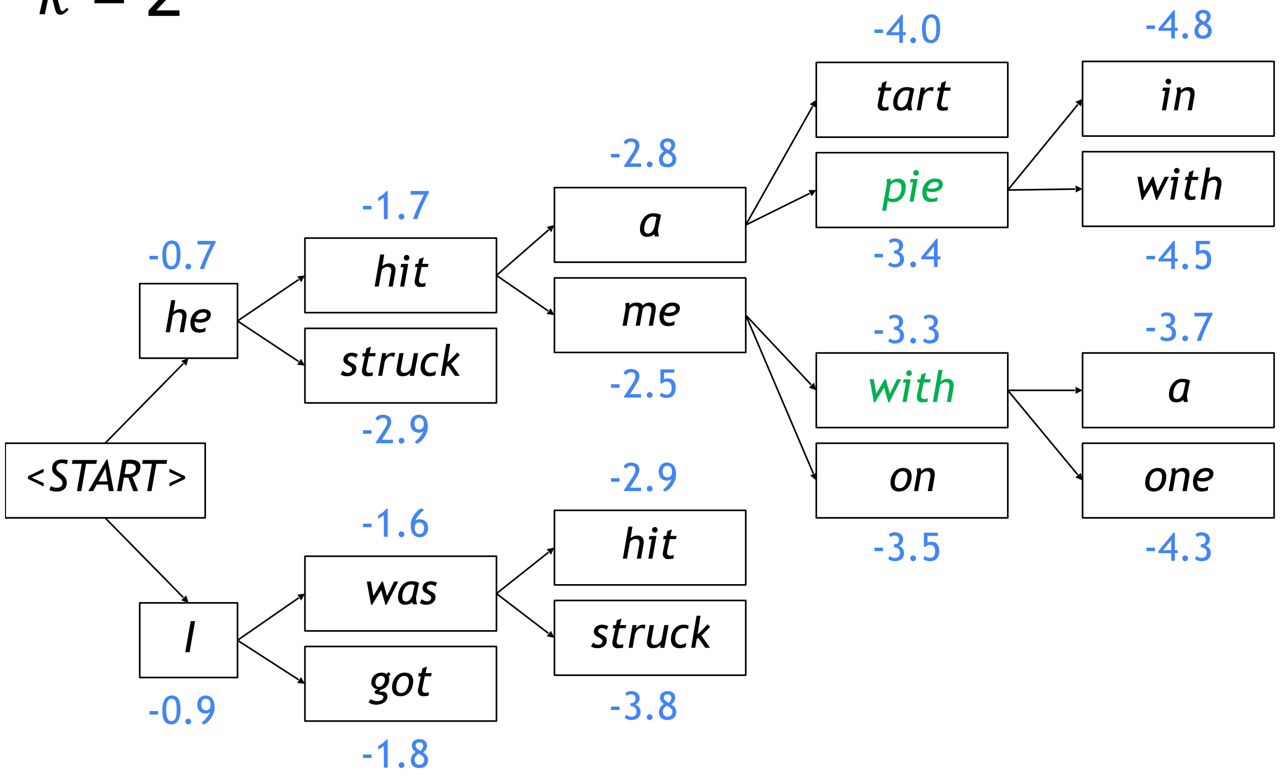


For each of the k hypotheses, find top k next words and calculate scores

https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf

Beam search: Example

- Beam size: k = 2

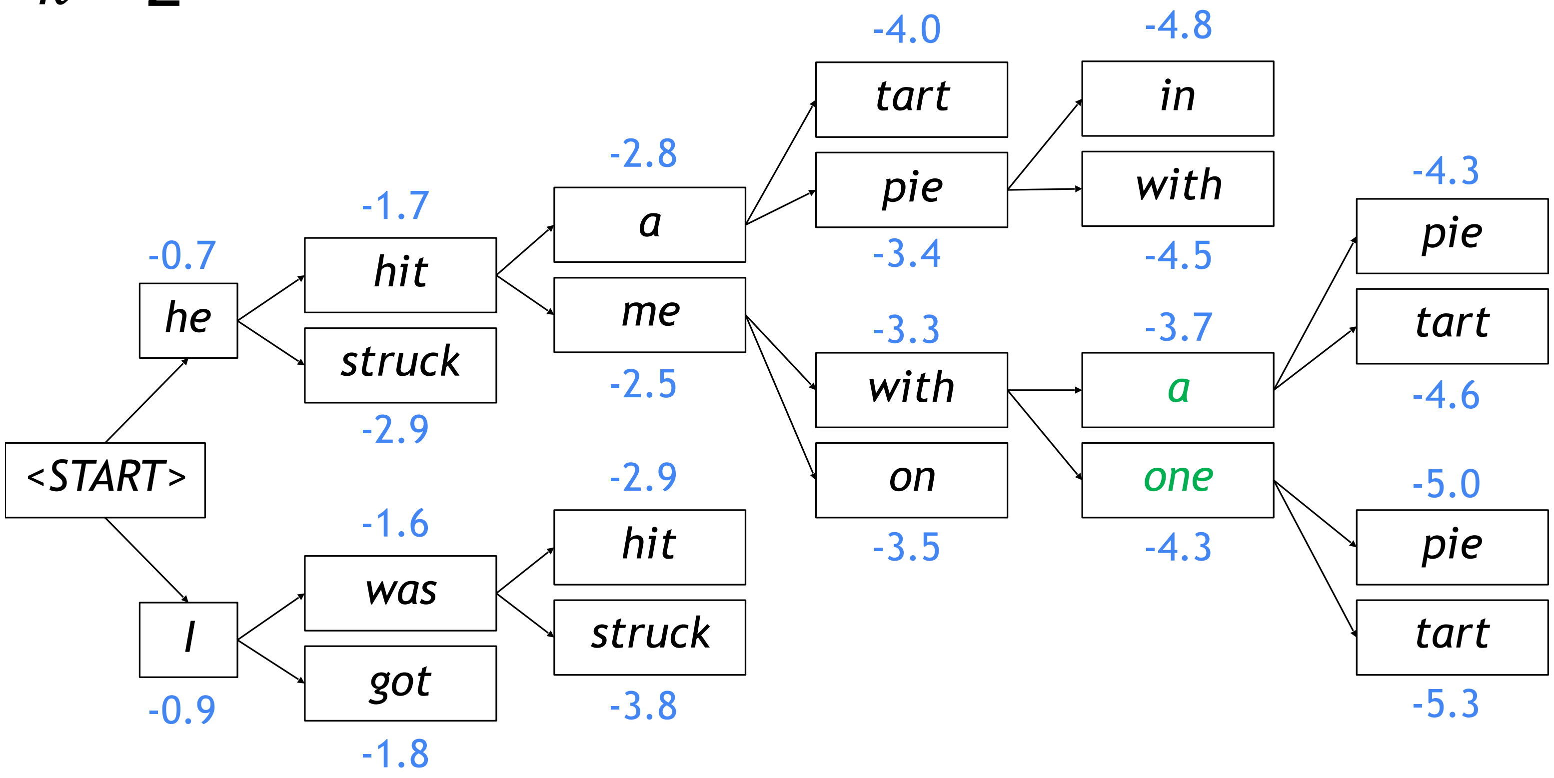


For each of the k hypotheses, find top k next words and calculate scores

https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf

Beam search: Example

- Beam size: k = 2



For each of the k hypotheses, find top k next words and calculate scores

https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf

Beam search: Stopping criterion

- In greedy decoding, usually we decode until the model produces a <END> token
  - For example: *<START> he hit me with a pie <END>*
- In beam search decoding, different hypotheses may produce <END> tokens on different timesteps
  - When a hypothesis produces <END>, that hypothesis is complete
  - Place it aside and continue exploring other hypotheses via beam search

Beam search: Stopping criterion

- **Usually we continue beam search until:**
  - We reach timestep T (where T is some pre-defined cutoff), or
  - We have at least n completed hypotheses (where n is the pre-defined cutoff)

Beam search: Finishing up

- We have our list of completed hypotheses
- How to select the top one with the highest score?
- Each hypothesis $y_1, \ldots, y_t$ on our list has a score

$$score(y_1, \ldots, y_t) = \log P_{LM}(y_1, \ldots, y_t | x) = \sum_{i=1}^{t} \log P_{LM}(y_i | y_1, \ldots, y_{i-1}, x)$$

- Problem with this: longer hypotheses have lower scores
- Fix: Normalize by length

$$score(y_1, \ldots, y_t) = \frac{1}{t} \sum_{i=1}^{t} \log P_{LM}(y_i | y_1, \ldots, y_{i-1}, x)$$

3.

BLEU score

Precision and Recall

Reference: Half of my heart is in Havana ooh na na

Predicted: Half as my heart is in Obama ooh na

$$precision = \frac{\#(correct\ words)}{length\_of\_prediction} = \frac{7}{9} = 78\%$$

$$recall = \frac{\#(correct\ words)}{length\_of\_reference} = \frac{7}{10} = 70\%$$

$$F - measure = \frac{precision \times recall}{\frac{1}{2}(precision + recall)} = \frac{0.78 \times 0.7}{0.5 \times (0.78 + 0.7)} = 73.78\%$$

Recall in Starcraft

Precision and Recall

Predicted (from model 1): Half as my heart is in Obama ooh na

Reference: Half of my heart is in Havana ooh na na

Predicted (from model 2): Havana na in heart my is Half ooh of na

Metric	Model 1	Model 2
Precision	78%	100%
Recall	70%	100%
F-measure	73.78%	100%

Flaw: no penalty for reordering

BLEU score

- BiLingual Evaluation Understudy (BLEU)
  - N-gram overlap between machine translation output and reference sentence
  - Compute precision for n-grams of size one to four
  - Add brevity penalty (for too short translations)

$$BLEU = \min(1, \frac{length\_of\_prediction}{length\_of\_reference})(\prod_{i=1}^{4} precision_i)^{\frac{1}{4}}$$

- Typically computed over the entire corpus, not on single sentences

BLEU score

Predicted (from model 1): Half as my heart is in Obama ooh na

Reference: Half of my heart is in Havana ooh na na

Predicted (from model 2): Havana na in heart my is Half ooh of na

Metric	Model 1	Model 2
Precision (1-gram)	$^{7}/_{9}$	$^{10}/_{10}$
Precision (2-gram)	$^{4}/_{8}$	$^{0}/_{9}$
Precision (3-gram)	$^{2}/_{7}$	$^{0}/_{8}$
Precision (4-gram)	$^{4}/_{6}$	$^{0}/_{7}$
Brevity penalty	$^{9}/_{10}$	$^{10}/_{10}$
BLEU	$0.9 \times \sqrt{3}/3 = 52\%$	0

boostcamp AI Tech

References

- deeplearning.ai-Beam Search
  - https://youtu.be/RLWuzLLSIgw
- deeplearning.ai-Refining Beam Search
  - https://youtu.be/gb__z7LlN_4
- OpenNMT-beam search
  - https://opennmt.net/OpenNMT/translation/beam_search/
- CS224n-NMT
  - https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf
- Sequence to sequence learning with neural networks, ICML'14
  - https://arxiv.org/abs/1409.3215
- Effective Approaches to Attention-based Neural Machine Translation, EMNLP 2015
  - https://arxiv.org/abs/1508.04025

End of Document
Thank You.