Attention in Machine Translation

Nikolay Arefyev
CMC MSU Department of Algorithmic Languages &
Samsung Moscow Research Center

What is Machine Translation?

- Machine Translation (MT): translate a sentence in one language to a sentence in another language.
  - Special case of sequence transduction

***Source language*:** ru

x: Мороз и солнце - день чудесный!

***Target language*:** en

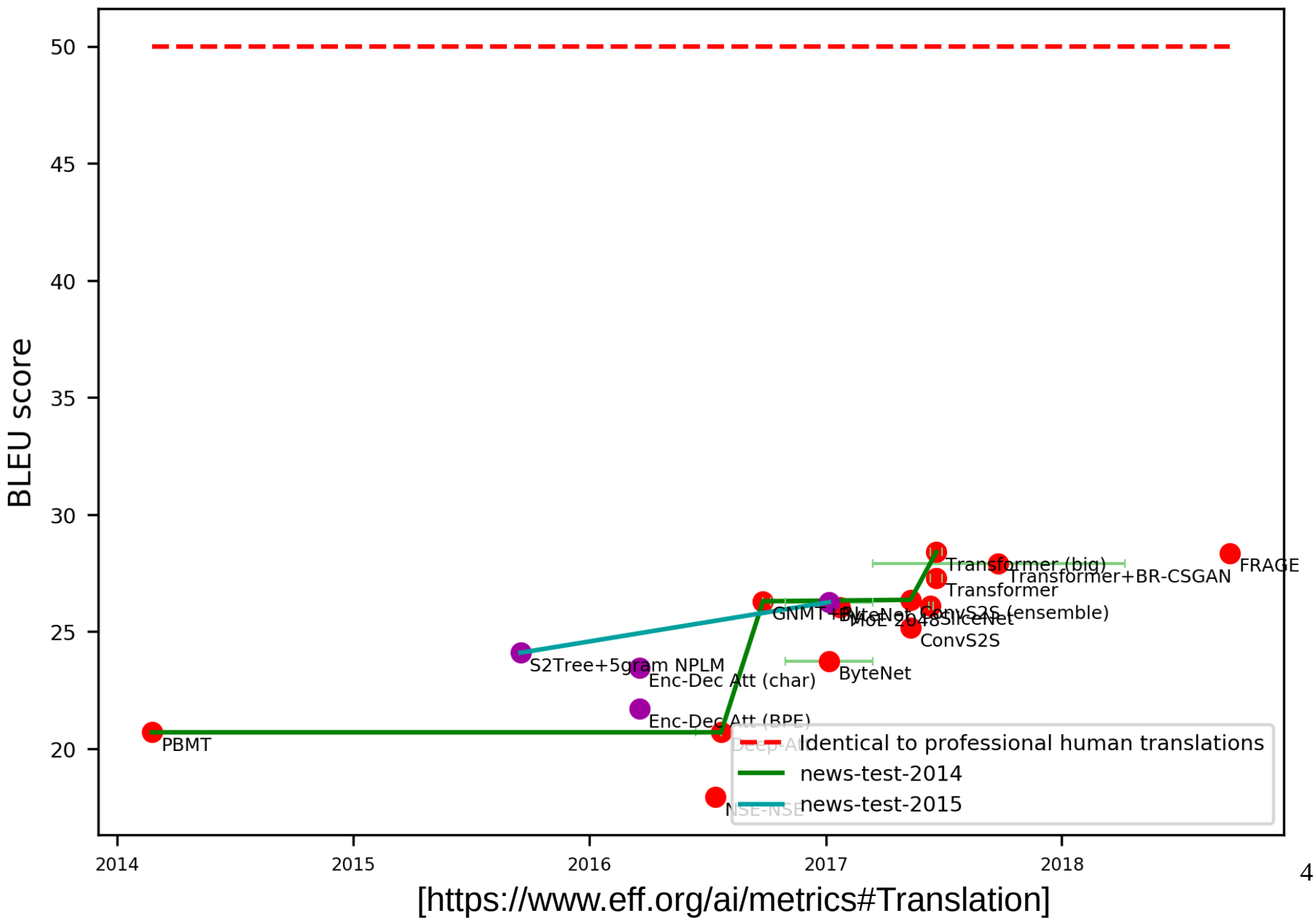
y: Frost and the sun - a wonderful day!

Approaches

- 1950s: Russian→English MT (Cold War)
  - Rule-based systems using bilingual dictionaries
- 1990s-2010s: **Statistical** MT (SMT)
  - Mainly **Phrase-Based** MT (PBMT)
  - Learns from (large) **sentence-aligned** bilingual corpora
    - Millions of sentence pairs
  - Consists of separate very complex components, learnt separately
  - developed by large groups for decades, doesn't generalize to new language pairs
- Since 2014: **Neural** MT (NMT)
  - Single NN learnt **end-to-end**
  - Learns from (large) **sentence-aligned** bilingual corpora
  - A few (good) student / month to implement
  - better quality of translation than SMT systems developed for decades

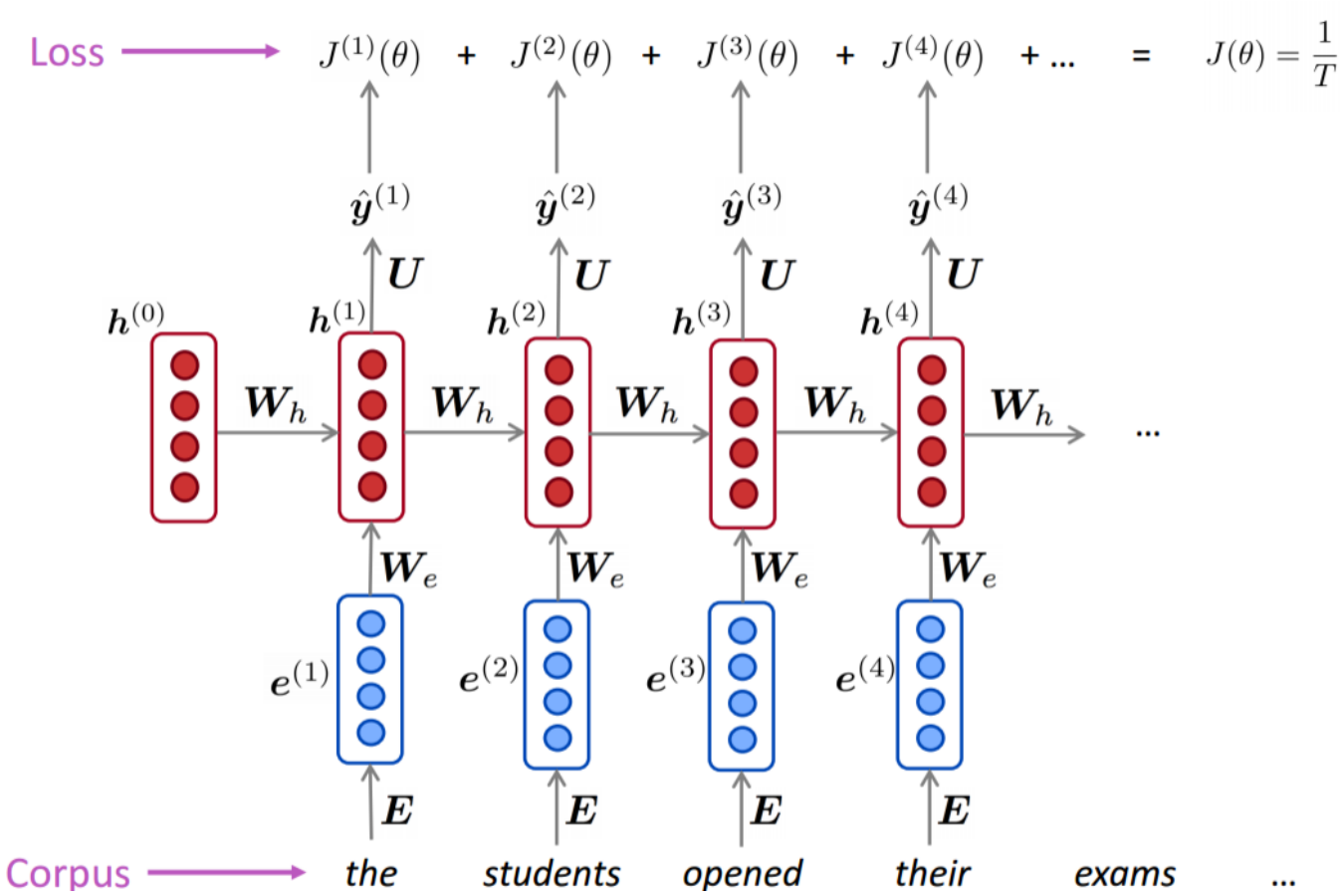
Google Translate: NMT, Yandex Translate: NMT+PBMT

MT vs. Human translation



[https://www.eff.org/ai/metrics#Translation]

RNN LM reminder



output distribution

$$\hat{y}^{(t)} = \text{softmax}\left(Uh^{(t)} + b_2\right) \in \mathbb{R}^{|V|}$$

hidden states

$$h^{(t)} = \sigma\left(W_h h^{(t-1)} + W_e e^{(t)} + b_1\right)$$

$h^{(0)}$ is the initial hidden state

word embeddings

$$e^{(t)} = Ex^{(t)}$$

words / one-hot vectors

$$x^{(t)} \in \mathbb{R}^{|V|}$$

Abigail See. Cs224n, lecture 8

RNN/LSTM LM reminder

$$P(y) = P(y_1)P(y_2|y_1)P(y_3|y_1, y_2)...P(y_T|y_1, ..., y_{T-1})$$

- estimate P(y)
- or generate new y:

  y_1 = random word / <START>

  for i from 2 to T:

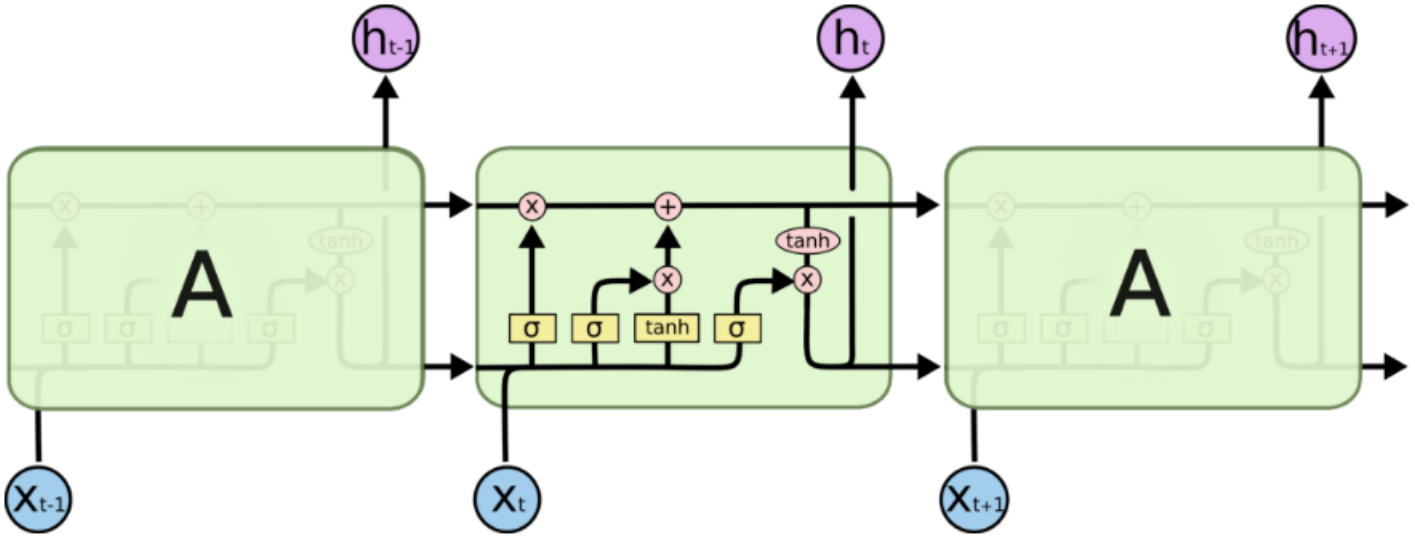
    y_i = argmax_y P(y|$y_{1 ... i-1}$) / sample from P(y|$y_{1 ... i-1}$)

LSTM Reminder

- Solves (partially) vanishing gradient problem of vanilla RNNs

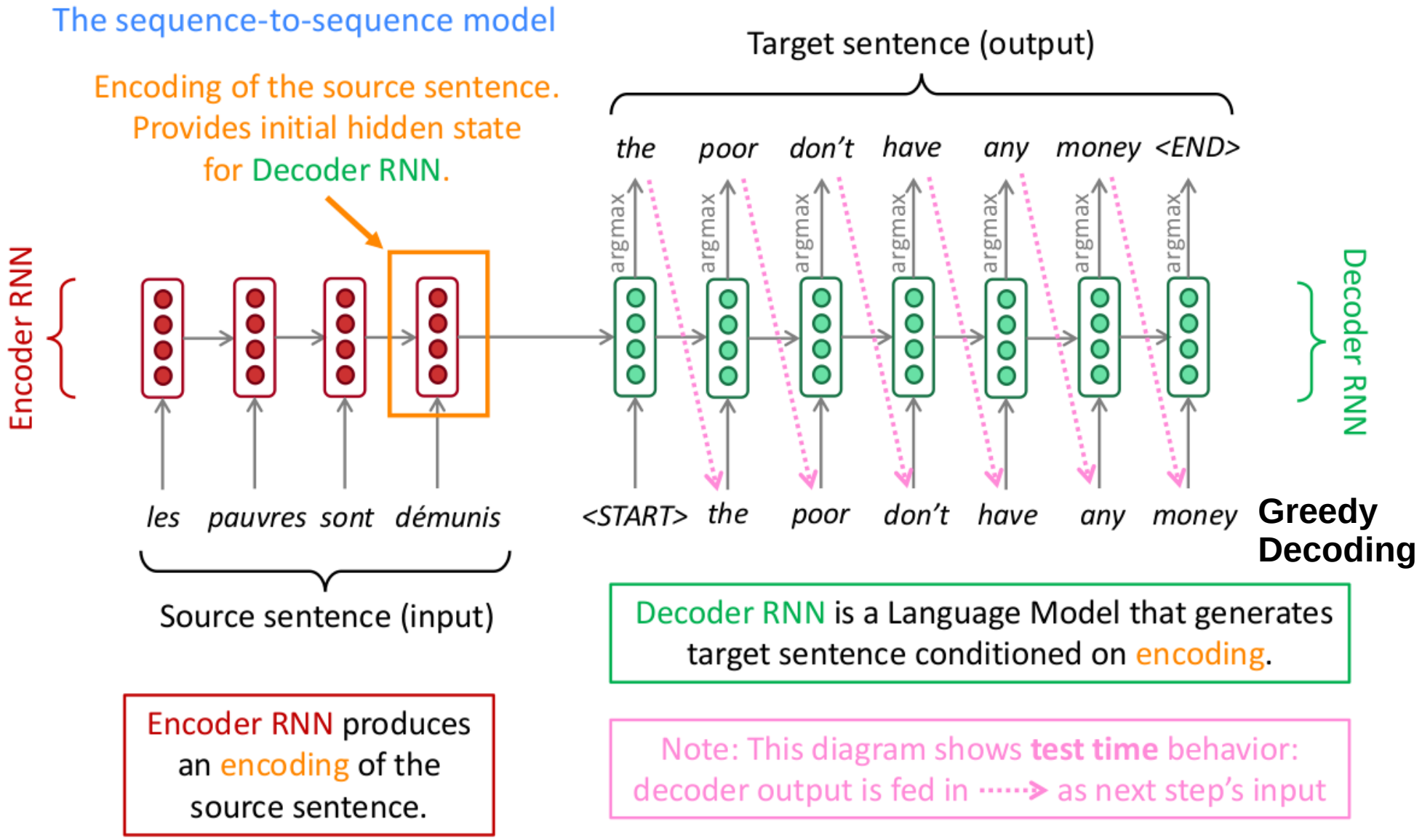
  => better models long range dependencies

$$i_t = \sigma\left(W^{(i)}x_t + U^{(i)}h_{t-1}\right)$$
$$f_t = \sigma\left(W^{(f)}x_t + U^{(f)}h_{t-1}\right)$$
$$o_t = \sigma\left(W^{(o)}x_t + U^{(o)}h_{t-1}\right)$$
$$\tilde{c}_t = \tanh\left(W^{(c)}x_t + U^{(c)}h_{t-1}\right)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$
$$h_t = o_t \circ \tanh(c_t)$$



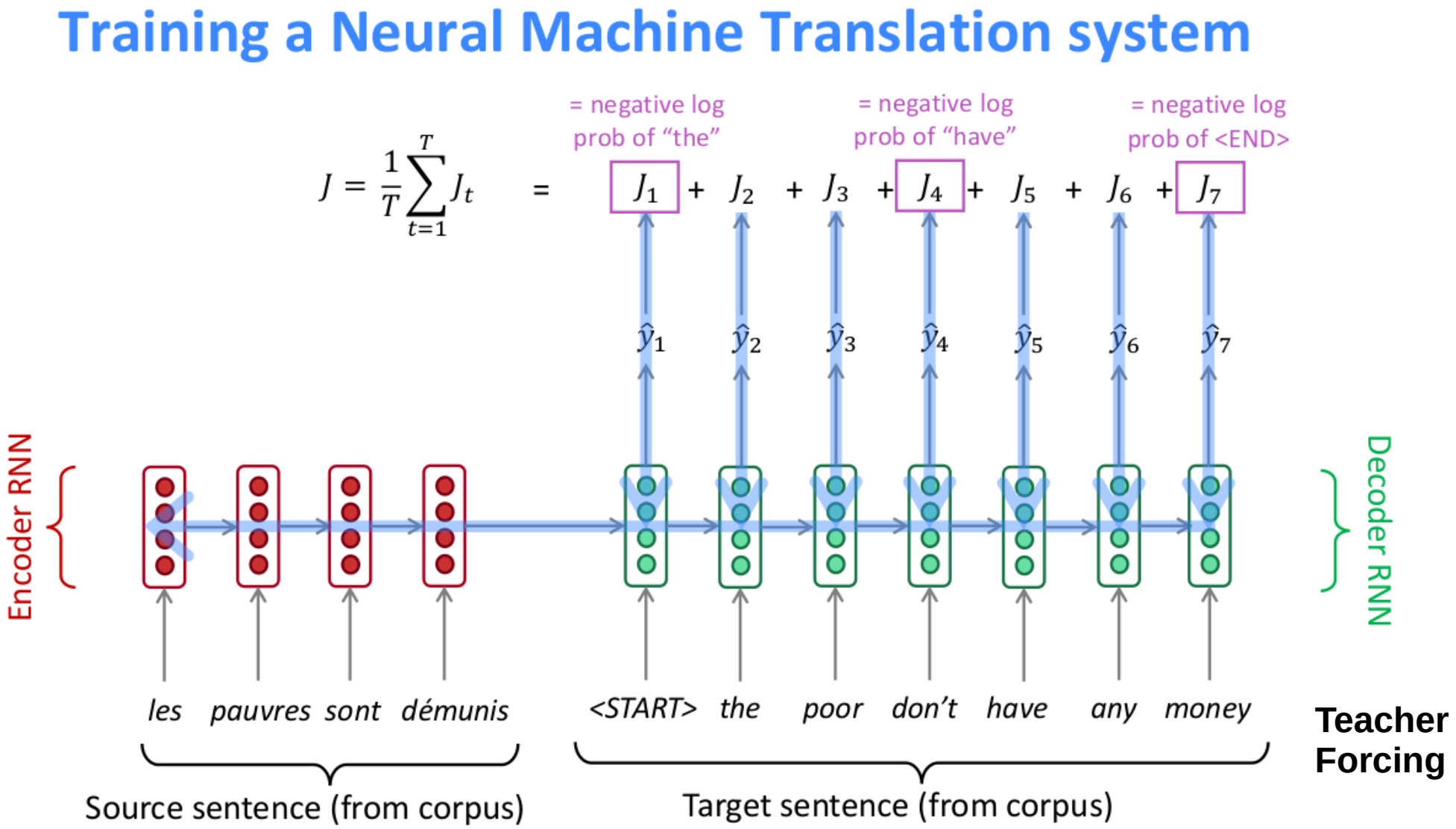
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

NMT: seq2seq



Socher, Manning. Cs224n, 2017

NMT: seq2seq loss



Socher, Manning. Cs224n, 2017

Conditional LM

- Decoder - conditional language model:

$$P(y|x) = P(y_1|x)P(y_2|y_1, x)...P(y_T|y_1, ..., y_{T-1}, x)$$

  - Machine translation
  - Image captioning
  - Speech recognition

- The whole system (encoder + decoder) is trained "**end-to-end**"
  - Calculate gradients of loss w.r.t. parameters of both encoder and decoder & do SGD

Sutskever 2014. Tricks

Sutskever, Vinyals, Le. Sequence to sequence learning with neural networks, 2014

- **reversed input**: shorter path btw. y_1 and x_1
- **stacked LSTMs: 4 layers**, 1000 cells per layer
- different LSTMs for encoder and decoder (different weights)
- **beam-search decoder**
- 1000 dim embeddings

Sutskever 2014. Training

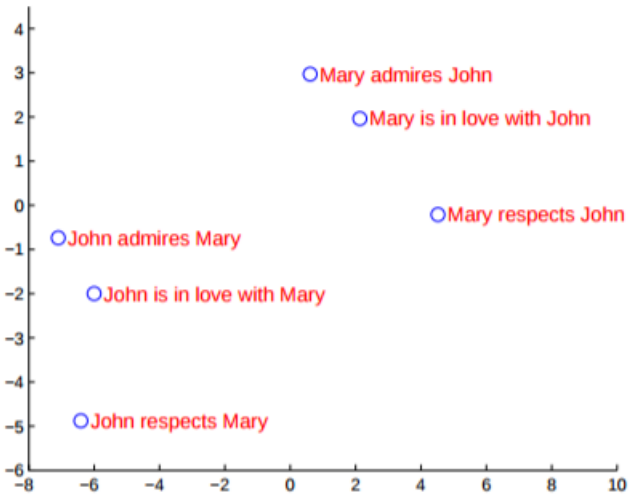
Sutskever, Vinyals, Le. Sequence to sequence learning with neural networks, 2014

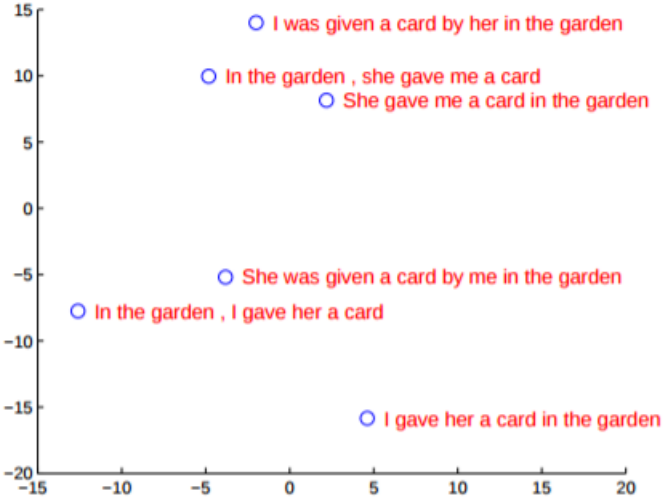
- SGD without momentum, lr=0.7, train for 7.5 epochs, after 5 epochs halve lr every 0.5 epoch
- Bs=128, balanced by length => 2x speedup!
- Gradient clipping: g:=5*g/||g||
- parallel En→Fr corpora: 12M sentence pairs
  - vocabulary: 160K/80K most frequent words + UNK
- Custom C++ impl on 8 GPUs – **<u>10 days</u>**
  - 1GPU per LTSM layer + 4GPUs for softmax

Sutskever 2014. Results

Sutskever, Vinyals, Le. Sequence to sequence learning with neural networks, 2014

- Little worse than SOTA SMT system, but much much simpler and easier to develop
- LSTM hidden state represents “thought”?
  - PCA of LSTM hidden states:

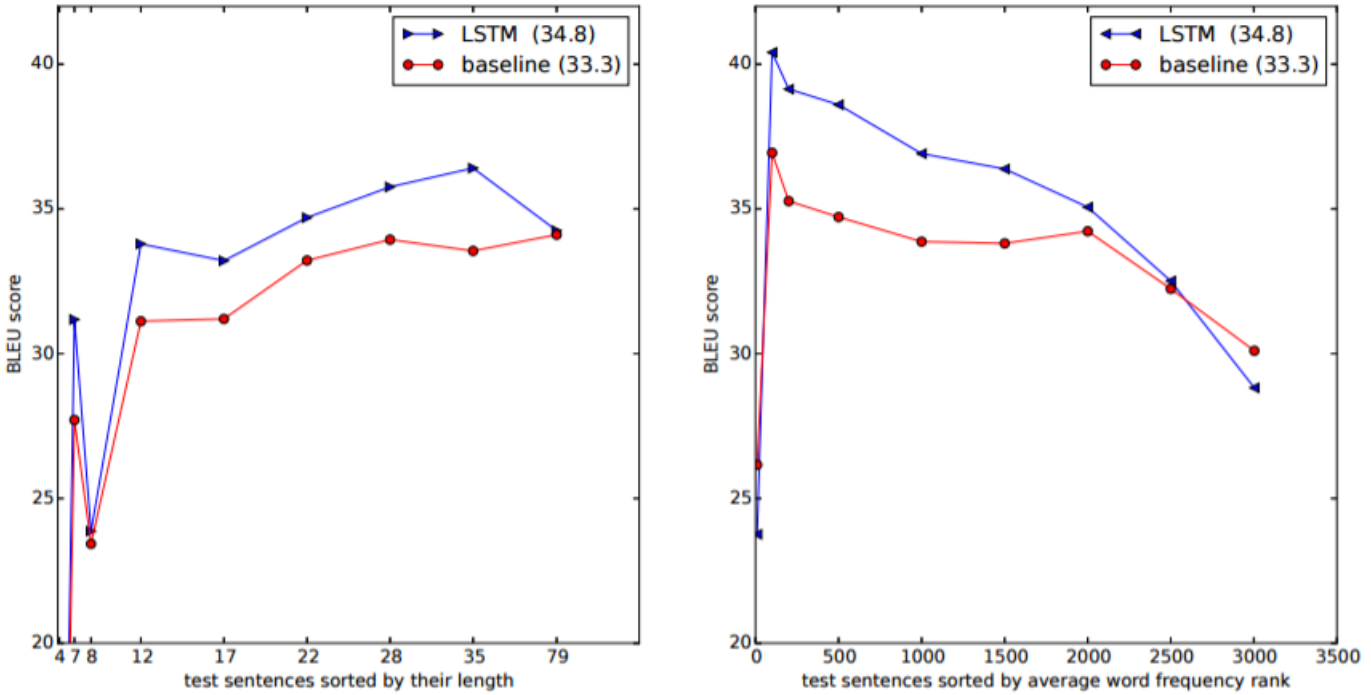




Results

Sutskever, Vinyals, Le. Sequence to sequence learning with neural networks, 2014

- no problems with long sentences (!)
- but problems with rare words



Decoding

- Decoding task: search for sequence y such as

$$y_1 y_2 \ldots y_T = argmax_y P(y|x) = argmax_y P(y_1|x) P(y_2|y_1, x) \ldots P(y_T|y_1, \ldots, y_{T-1}, x)$$

- Cannot enumerate all possible y (do full search)
  - V ^ T, where V=100K-1M, T=10-100
- Use heuristic search (approximate argmax):
  - Ancestral / forward sampling: $y_i \sim P(y \mid y_1, \ldots, y_{i-1})$
  - Greedy search: $y_i = \text{argmax}_y P(y \mid y_1, \ldots, y_{i-1})$
  - Beam search ← most common for MT

Greedy Decoding

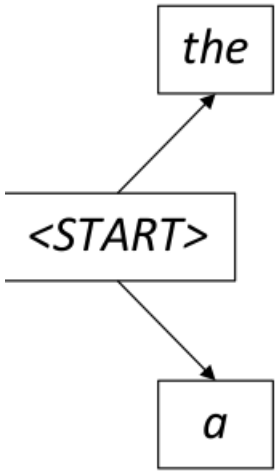
- Greedy decoding has no way to undo decisions!
  - *les pauvres sont démunis (the poor don't have any money)*
  - → *the* ____
  - → *the poor* ____
  - → *the poor are* ____

- Better option: use beam search (a search algorithm) to explore *several* hypotheses and select the best one

Socher, Manning. Cs224n, 2017

Beam search

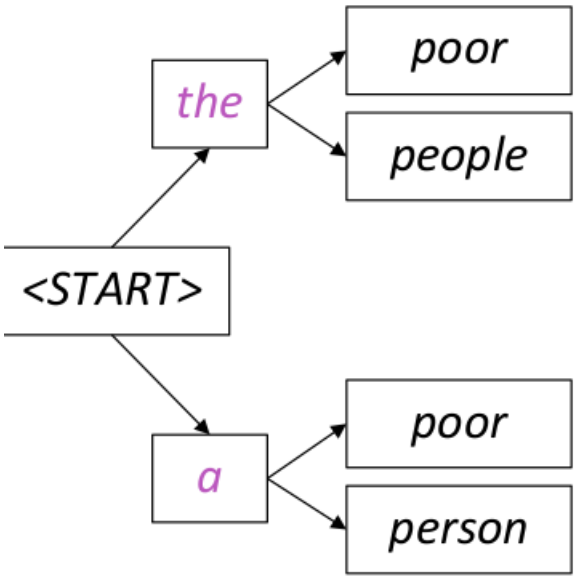
Beam size = 2



Socher, Manning. Cs224n, 2017

Beam search

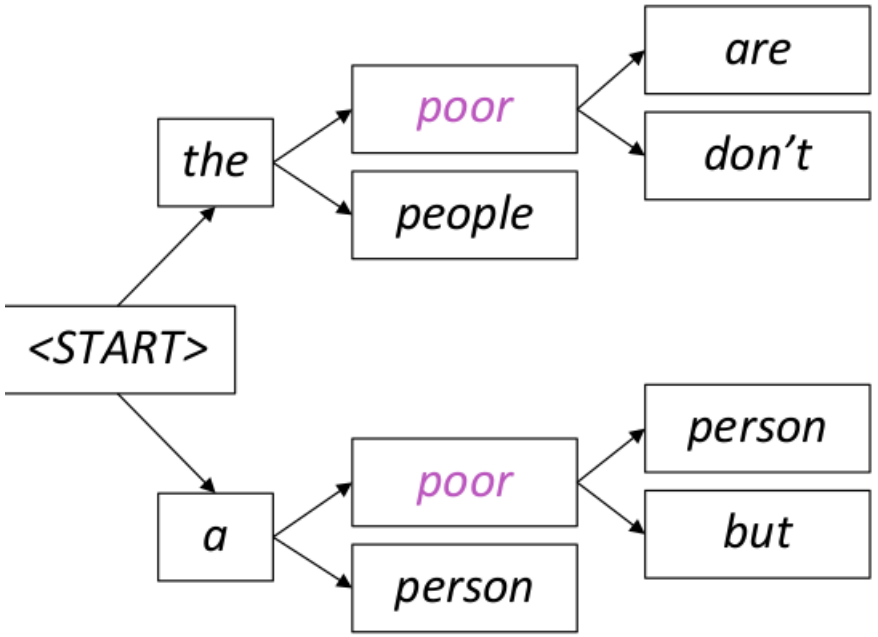
Beam size = 2



Socher, Manning. Cs224n, 2017

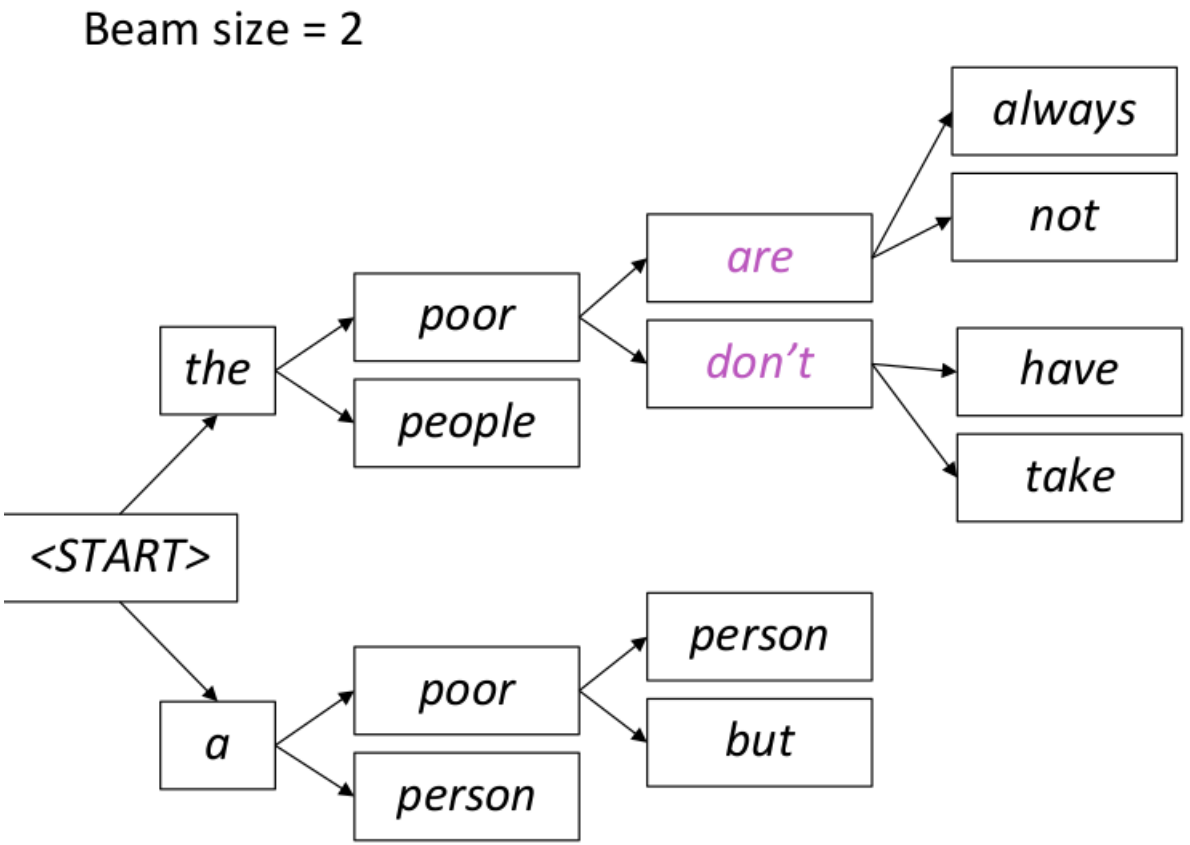
Beam search

Beam size = 2



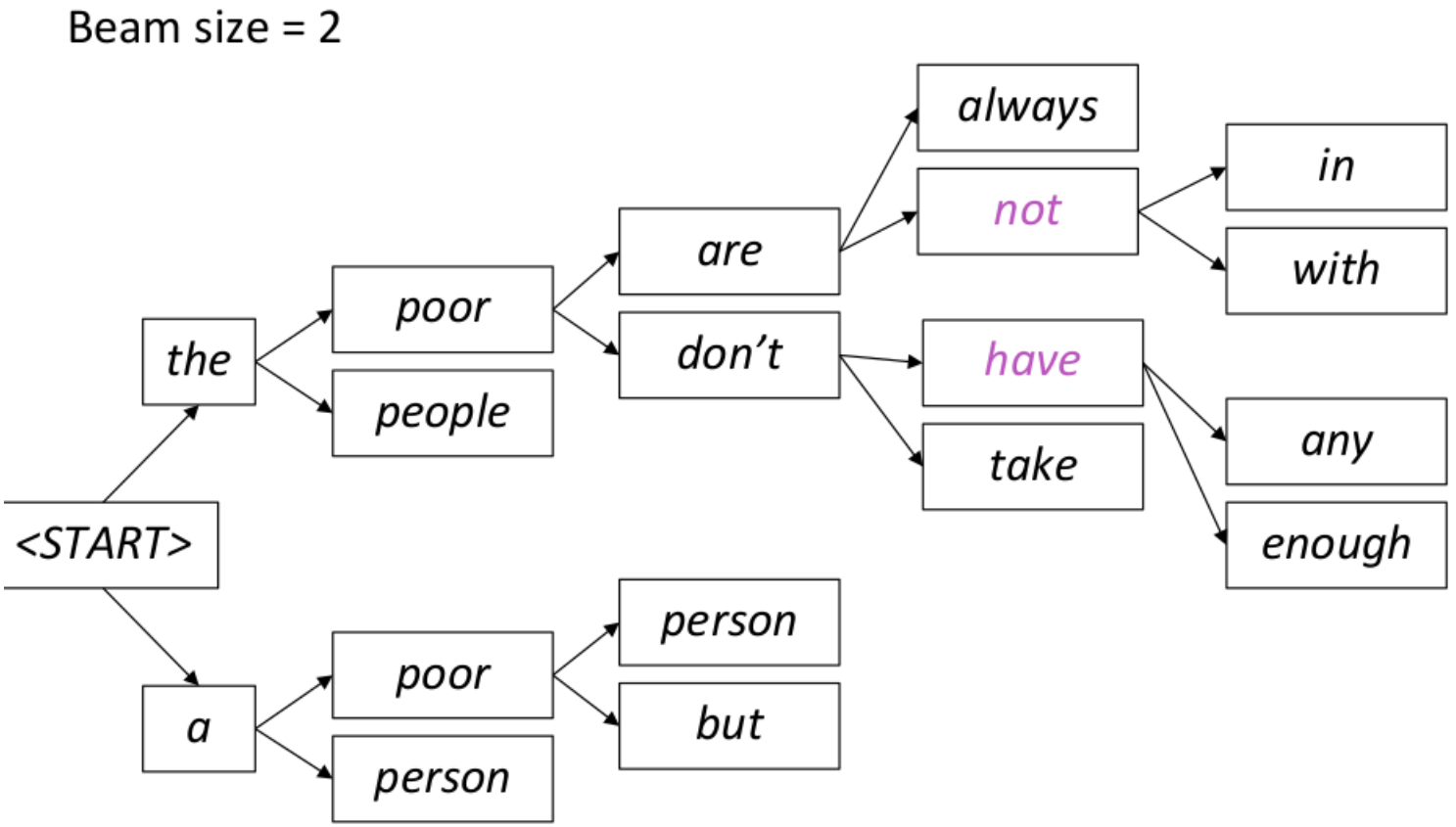
Socher, Manning. Cs224n, 2017

Beam search



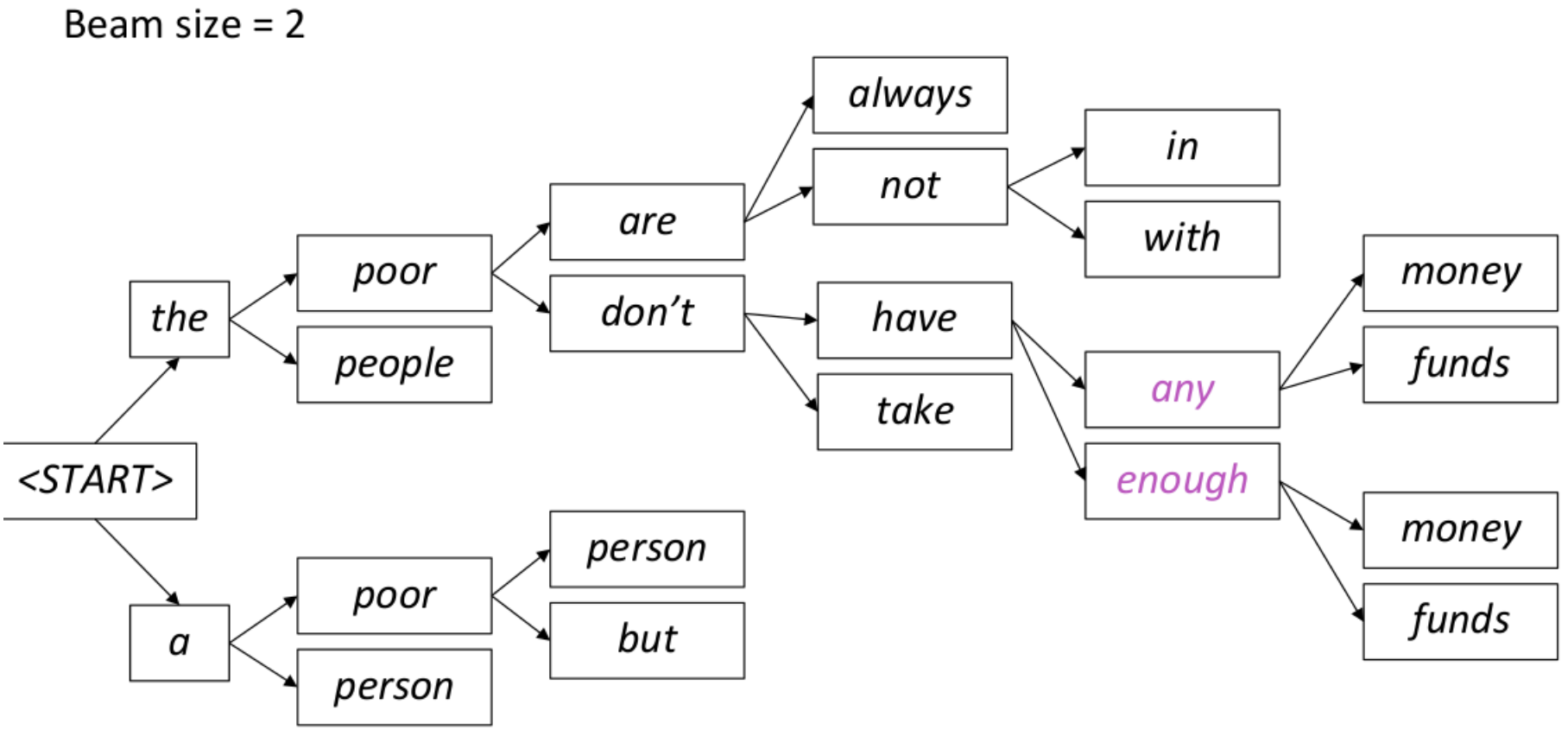
Socher, Manning. Cs224n, 2017

Beam search



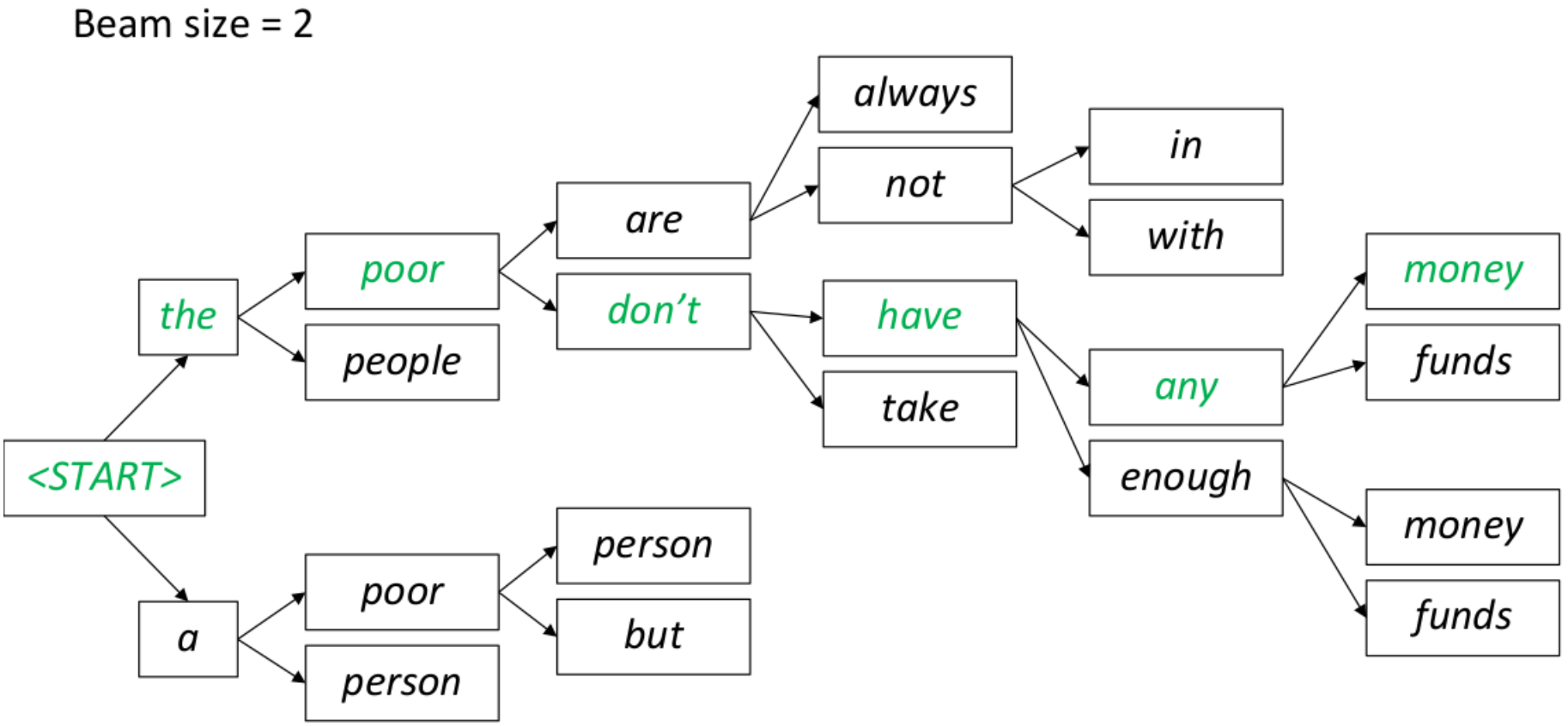
Socher, Manning. Cs224n, 2017

Beam search



Socher, Manning. Cs224n, 2017

Beam search



Socher, Manning. Cs224n, 2017

Beam search

- Ideally we want to find *y* that maximizes

$$P(y|x) = P(y_1|x)\, P(y_2|y_1, x)\, P(y_3|y_1, y_2, x) \ldots, P(y_T|y_1, \ldots, y_{T-1}, x)$$

- We could try enumerating all *y* → too expensive!
  - Complexity $O(V^T)$ where *V* is vocab size and *T* is target sequence length

- **Beam search**: On each step of decoder, keep track of the *k* most probable partial translations
  - *k* is the beam size (in practice around 5 to 10)
  - Not guaranteed to find optimal solution
  - But much more efficient!

Socher, Manning. Cs224n, 2017

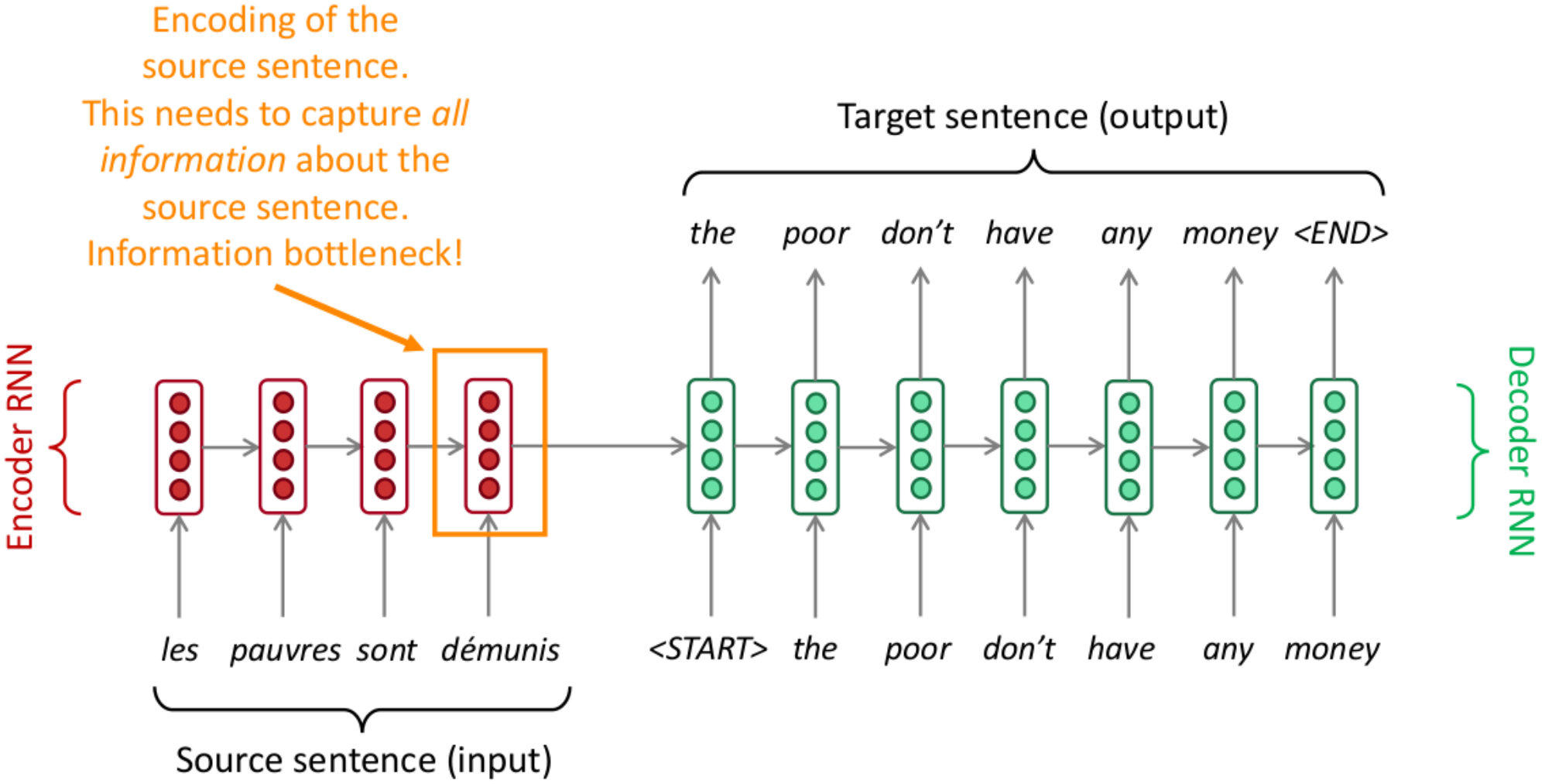
Length bias in NMT

- Maximizing *s(e)=P(e|source):* $s(e) = \sum_{i=1}^{m} \log P(e_i \mid e_{1:i})$

  results in the shorter translations be preferred to longer translations.
- Length normalization: $s'(e) = s(e) / m$
- Google NMT: $s'(e) = s(e) \Big/ \frac{(5+m)^\alpha}{(5+1)^\alpha}$
- Word reward $s'(e) = s(e) + \gamma m$

Murray and Chiang. Correcting length bias in neural machine tranlations, 2018.

Seq2seq bottleneck

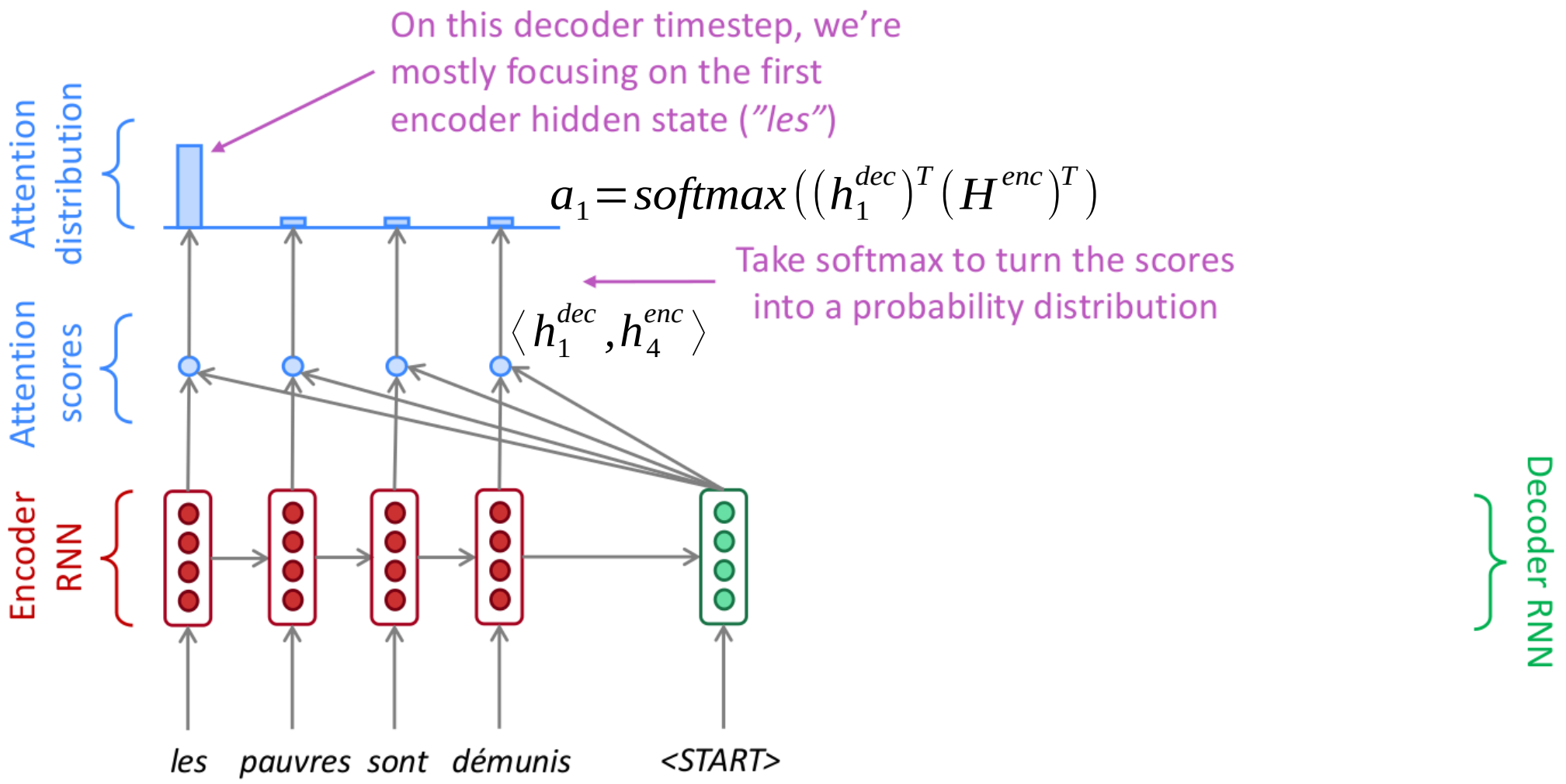


Socher, Manning. Cs224n, 2017

Attention: Idea

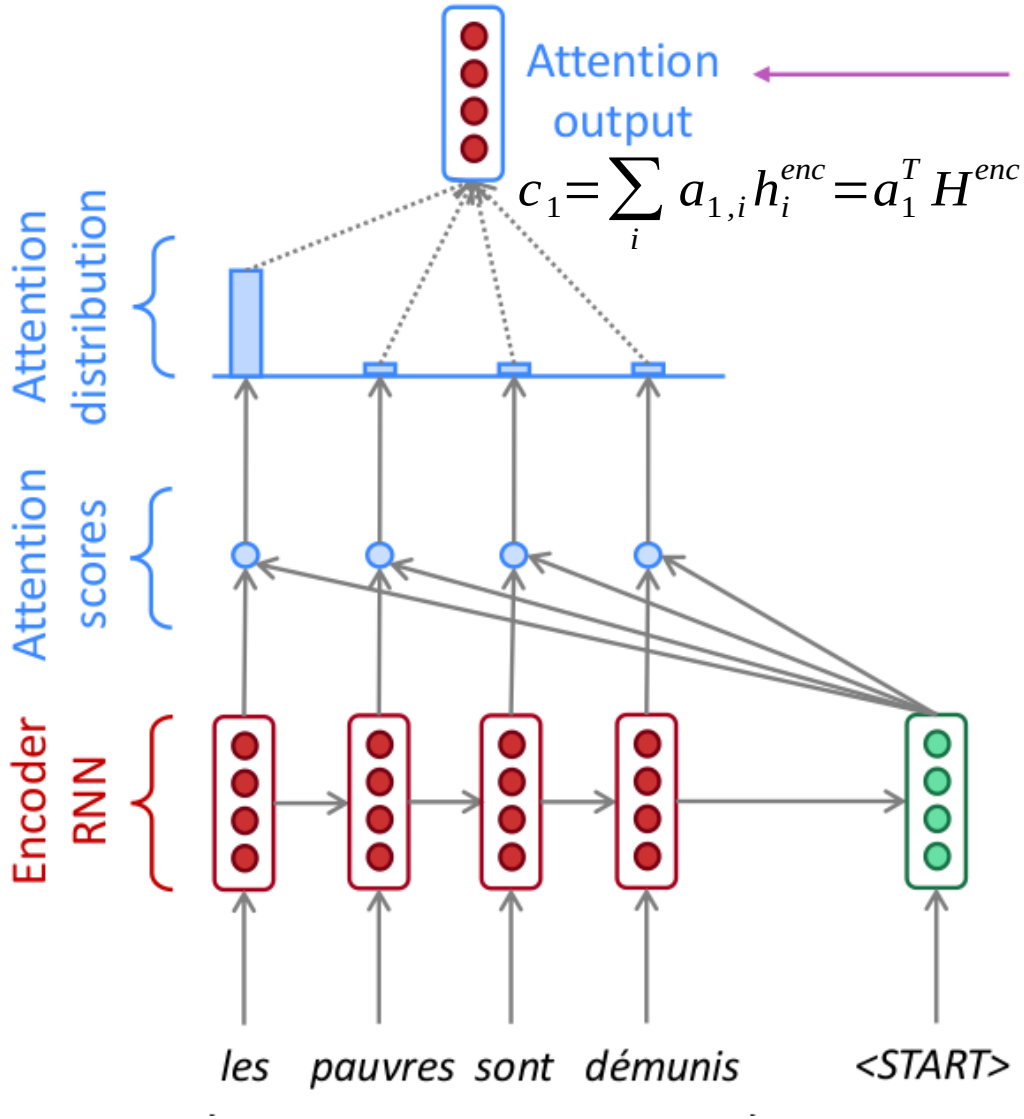
- At each timestep (when generating next word) the decoder should focus on the **most relevant (to this timestep) part** of input sequence
- Hidden state of the decoder at each timestep represents a “rough idea” of what it wants to say next
  - Help it to “put this idea into words”: find a part of input sequence relevant to this “idea” and “remind” this part to him

Attention: Example



Socher, Manning. Cs224n, 2017

Attention: Example



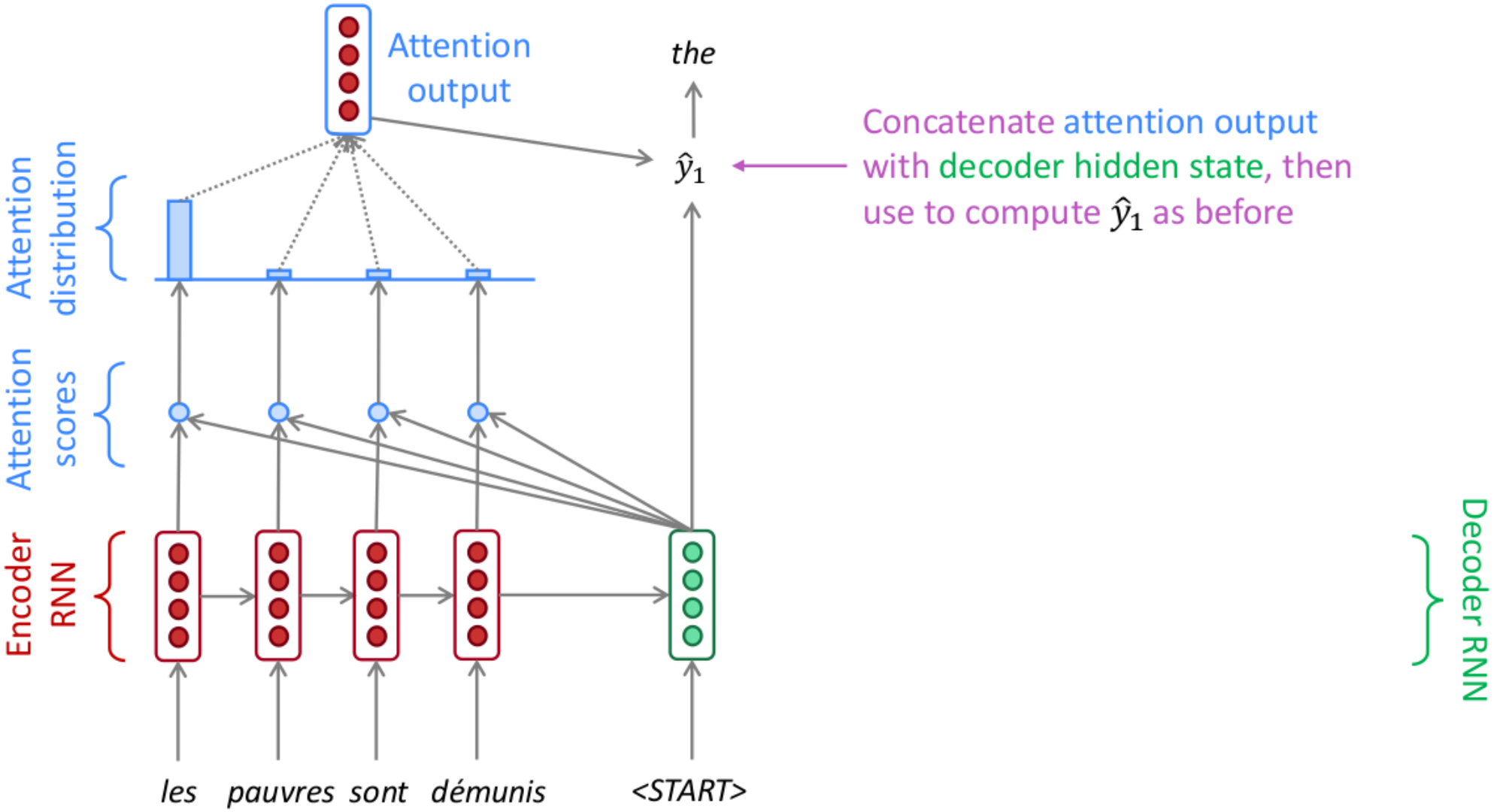
Use the attention distribution to take a **weighted sum** of the encoder hidden states.

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

Decoder RNN

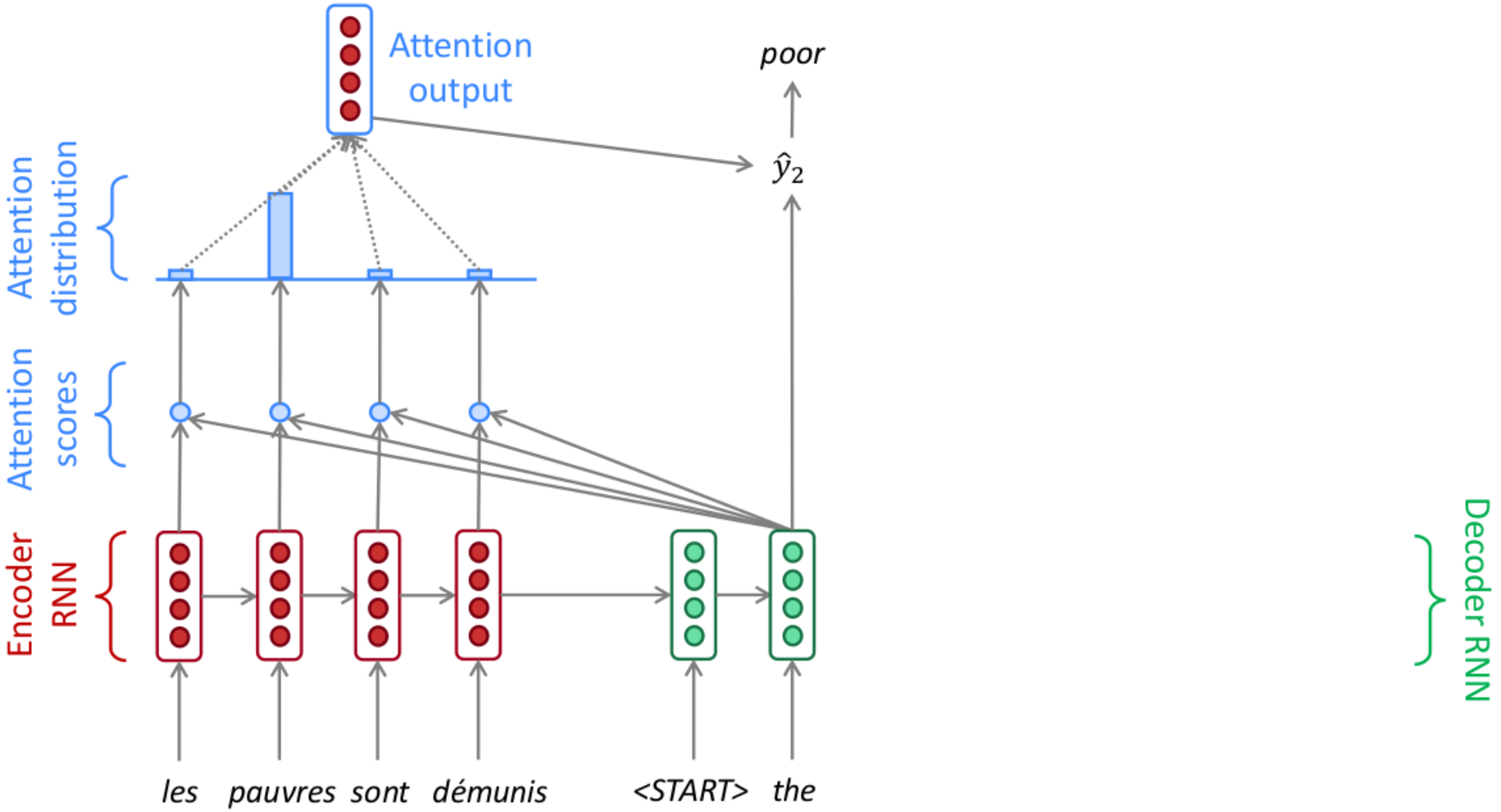
Socher, Manning. Cs224n, 2017

Attention: Example



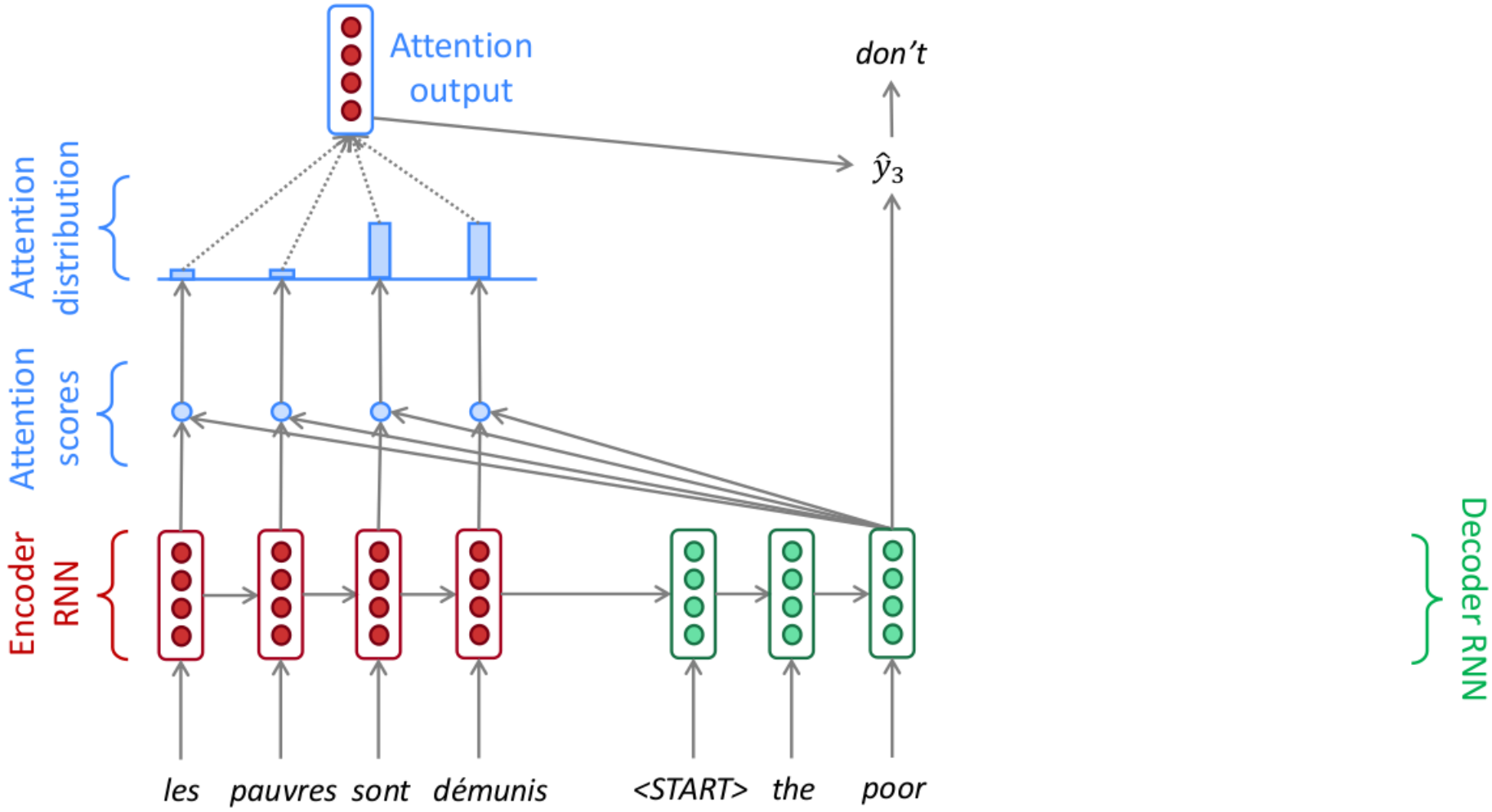
Socher, Manning. Cs224n, 2017

Attention: Example



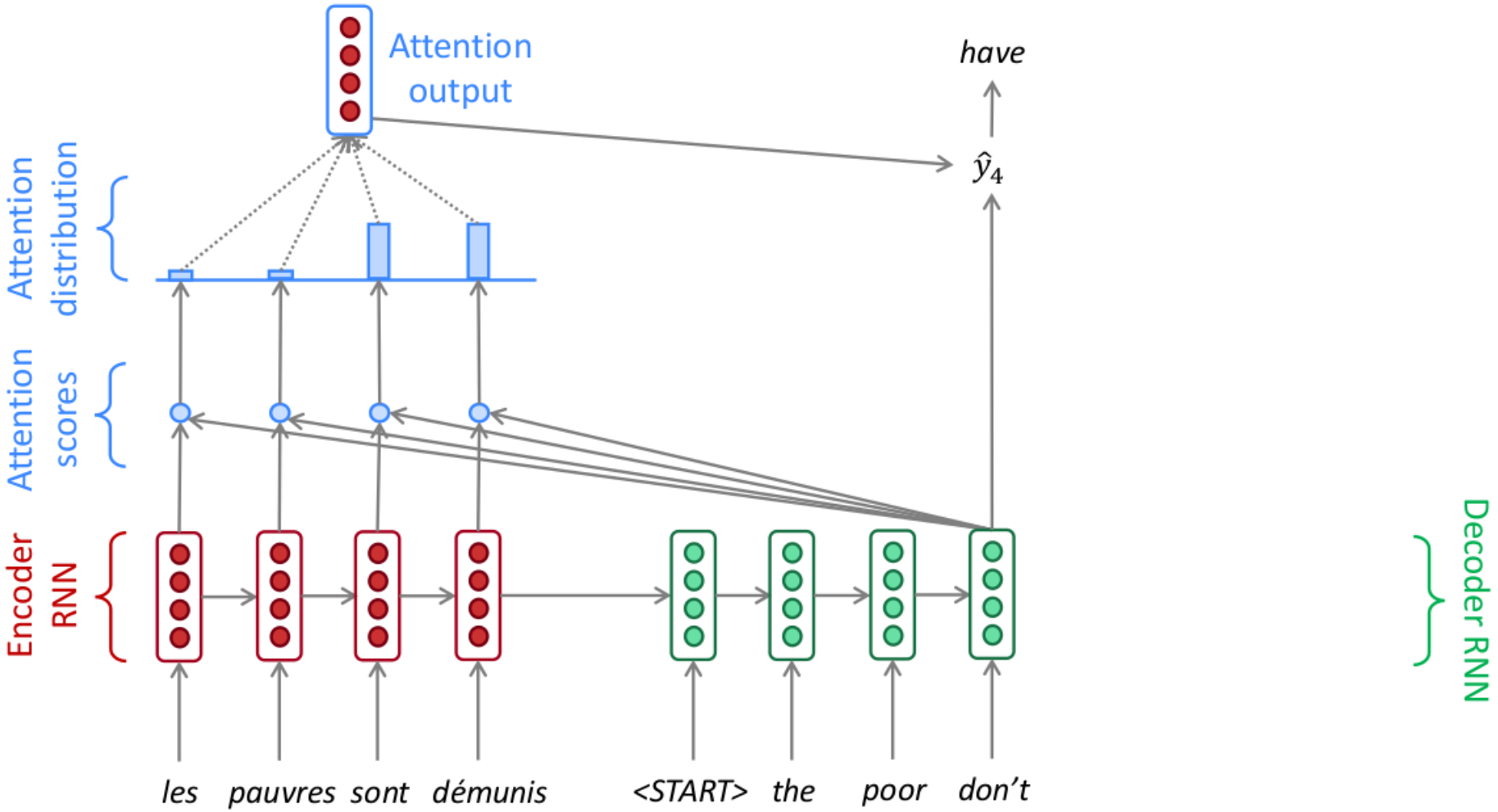
Socher, Manning. Cs224n, 2017

Attention: Example



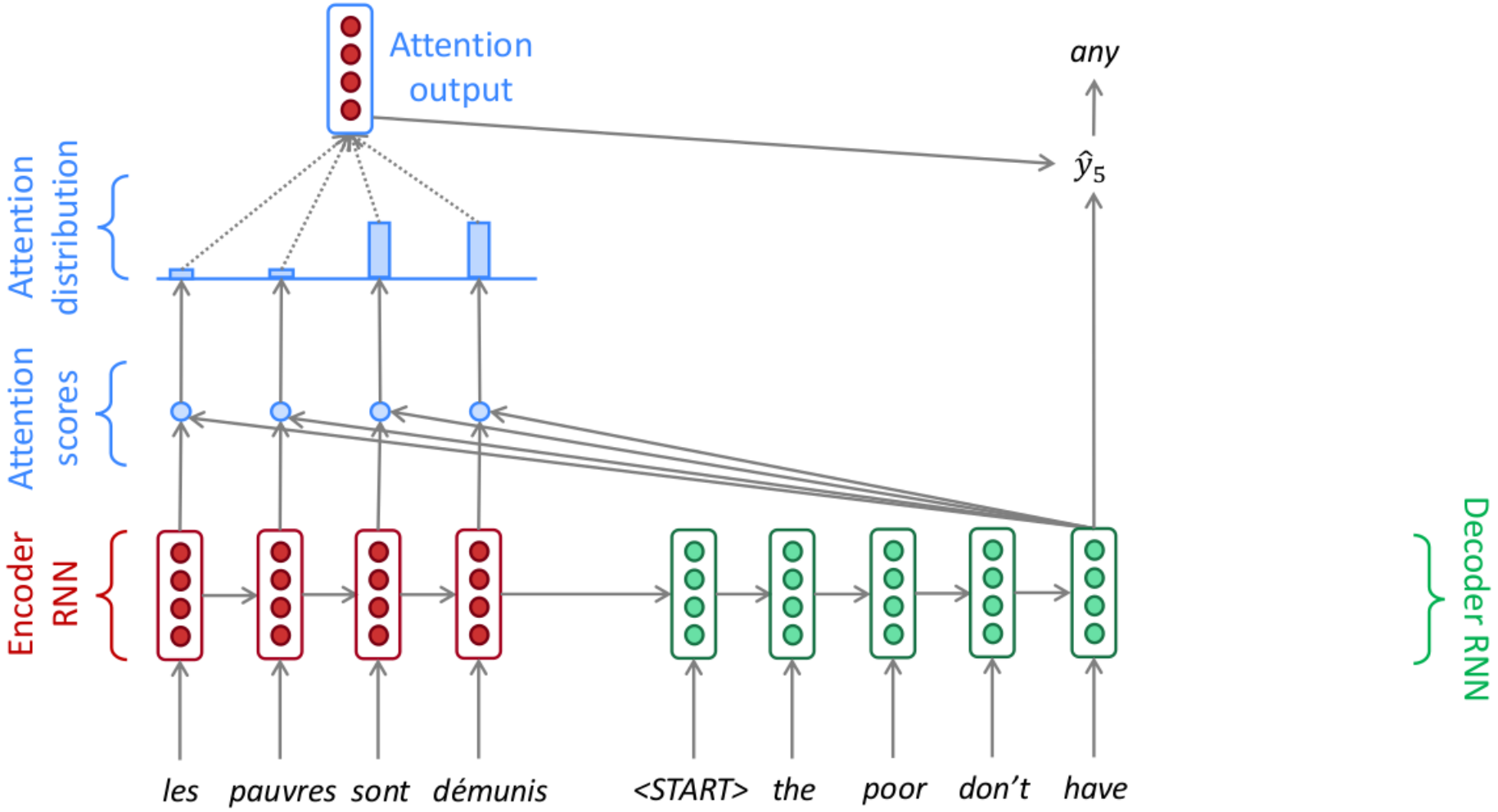
Socher, Manning. Cs224n, 2017

Attention: Example



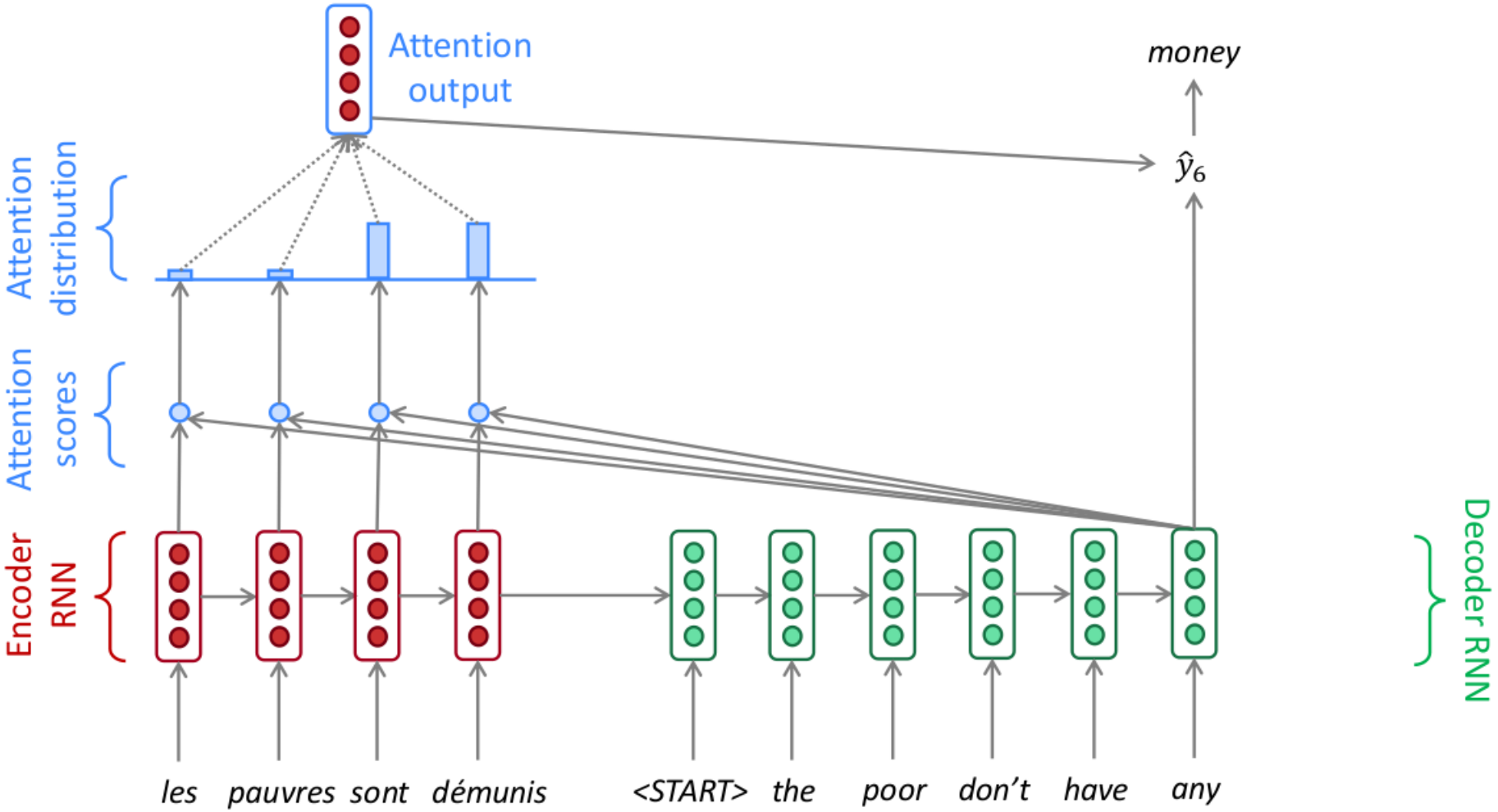
Socher, Manning. Cs224n, 2017

Attention: Example



Socher, Manning. Cs224n, 2017

Attention: Example

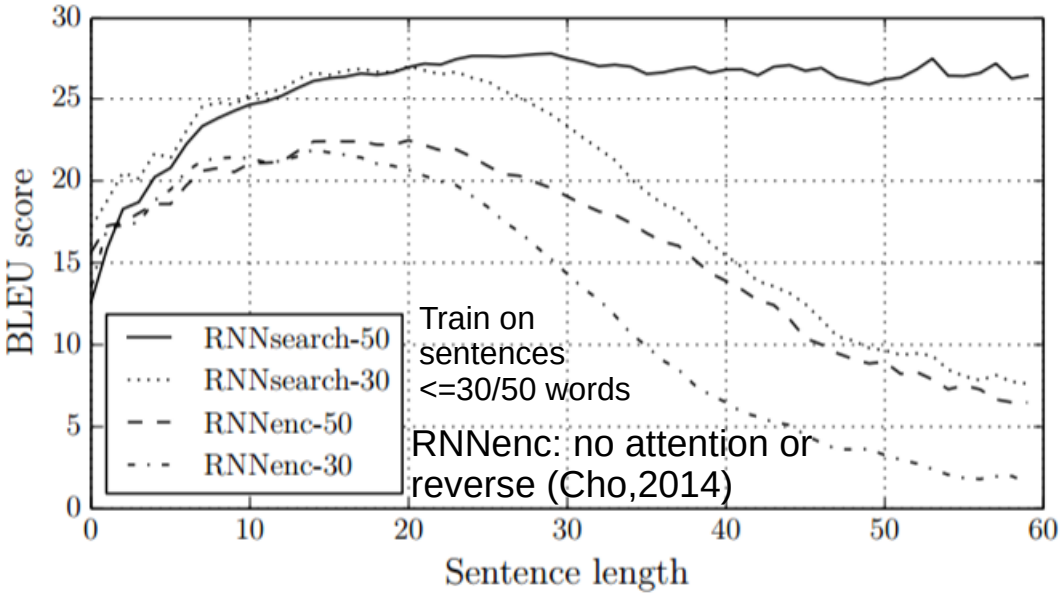


Socher, Manning. Cs224n, 2017

Bahdanau 2015

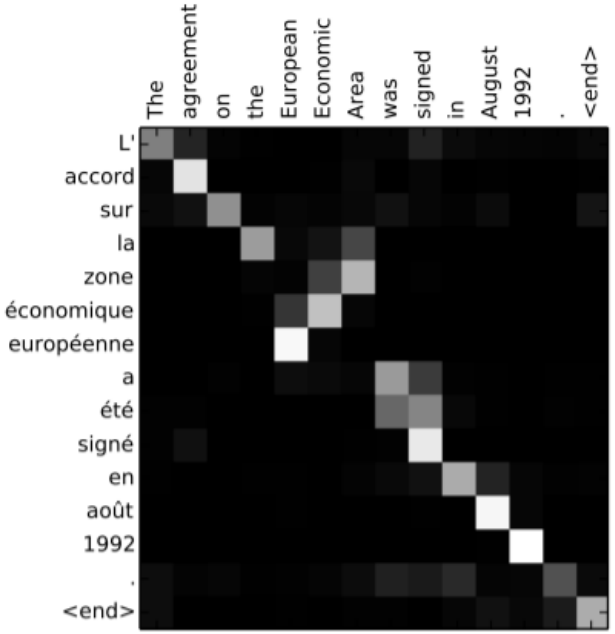
Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate, 2015

- Solves the problem of low quality for long sequences
  - Alternative to reversing

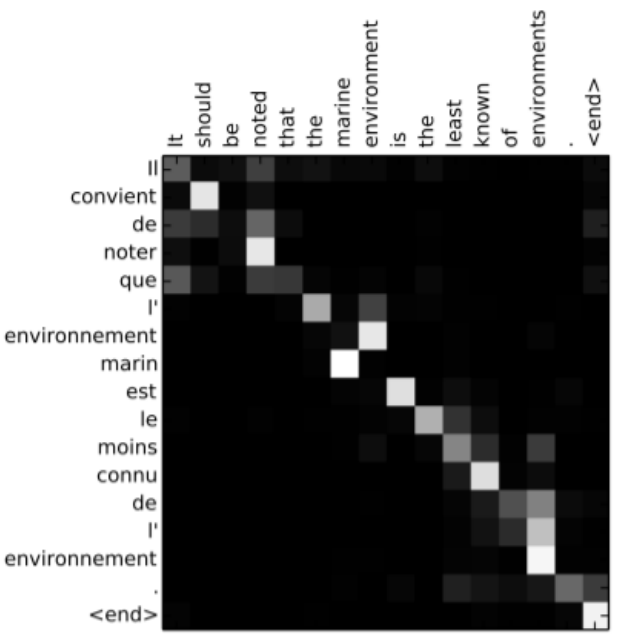


Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate, 2015

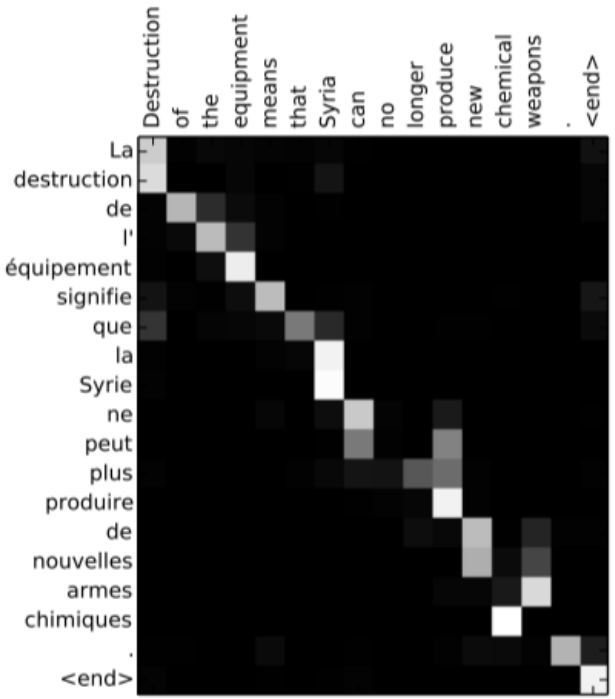
Bahdanau 2015

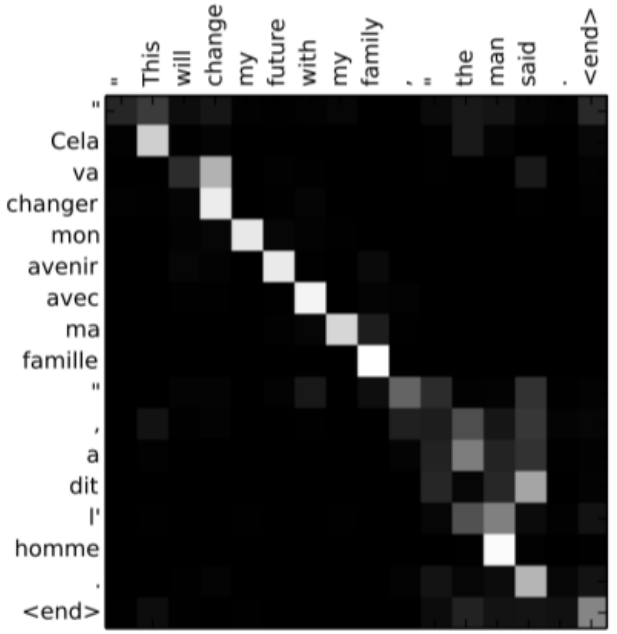


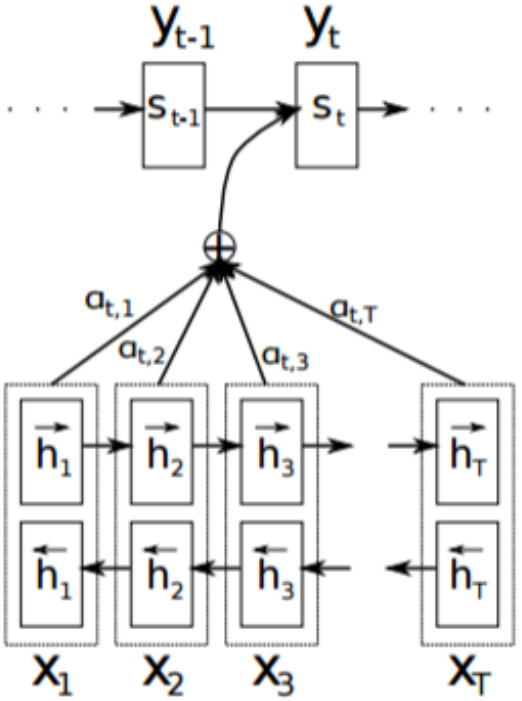
(a)



(b)







Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate, 2015

Attention

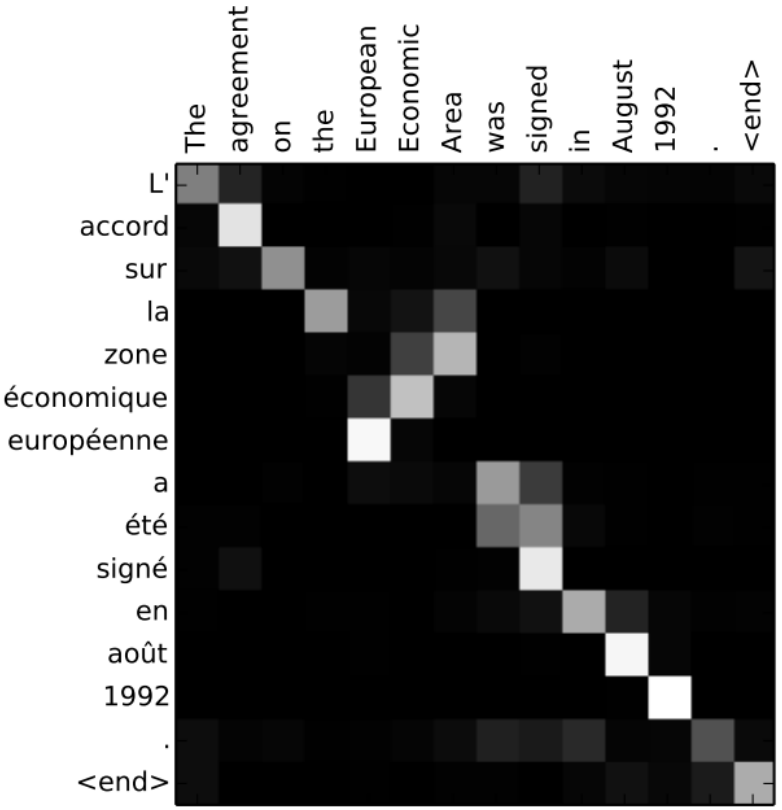
- Improves translation quality significantly
  - Solves **information bottleneck** problem, acts like an unlimited size memory
  - Helps **gradient propagation** from encoder to decoder, especially in long sequences
    - Remember vanishing gradient problem partially solved by LSTMs? But they have sigmoid forget gates! (<1)

Britz et al., 2017. Massive Exploration of Neural Machine Translation Architectures:

“we found that the attention-based models exhibited significantly larger gradient updates to decoder states throughout training. This suggests that the attention mechanism acts more like a ”weighted skip connection” that optimizes gradient flow than like a ”memory” that allows the encoder to access source states, as is commonly stated in the literature”

Attention

- Improves interpretability
  - Networks learns alignment as byproduct of translation
  - We can look at which fragments were translated to which



Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate, 2015

Attention

Attention is a new basic layer type (along with feed-forward, convolutional and recurrent)

- Works with variable length inputs (texts)
  - like RNNs, CNNs
  - unlike feed-forward
- Used in SOTA models in lots of tasks (question answering, image captioning, ...)

Do I hold your attention?