Economics 2355: Seq2Seq and Machine Translation

Melissa Dell

Harvard University

March 2021

Machine Translation

- Today we are going to talk about machine translation
- While you may not care about machine translation per se, this field has pioneered some of the most successful innovations in neural-based NLP and is the most straightforward context in which to understand those innovations
- Much of what will discuss echoes themes from elsewhere in the course and is more broadly applicable

Machine Translation

- Interest in machine translation stretches back to the beginning of computing
- It was one of the first (perhaps the first) example of using computers as more than just a calculator
- Let's see a video from the 1950s

Outline

Statistical Machine Translation

- Statistical Machine Translation (SMT) dominated in the pre-deep learning era
- These systems were complicated - many, many engineers working over many years to engineer a system that gave reasonable results
- We won't begin to go into all the details, these systems were complicated. Rather, it serves as a comparison for neural machine translation (NMT)

Statistical Machine Translation

Basic idea: use the data to estimate a probabilistic model

We want to find the best sentence in language y given a sentence in language x

$$\arg\max_{y} P(y|x) \tag{1}$$

Use Bayes rule to break this into two components that are learned separately:

$$\arg\max_{y} P(x|y)P(y) \tag{2}$$

Statistical Machine Translation

$$\arg\max_y P(y|x) = \arg\max_y P(x|y)P(y) \tag{3}$$

$P(x|y)$ models how words and phrases should be translated from x to y given parallel data

$P(y)$ models how to write a good sentence in target language y - it is learned only from data on language y

Translation

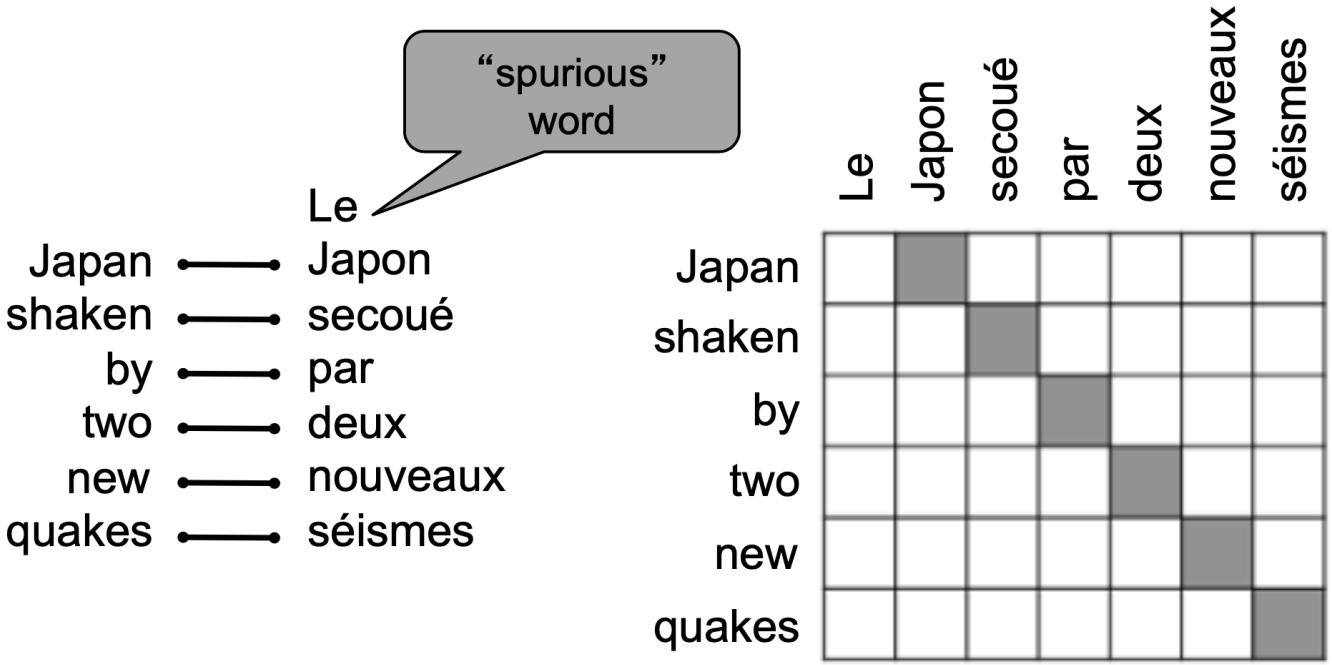
To learn a translation model, we need parallel data: i.e. pairs of human translated Mandarin and English sentences.

To learn translation model $P(x|y)$, rewrite this as

$$P(x, a|y) \tag{4}$$

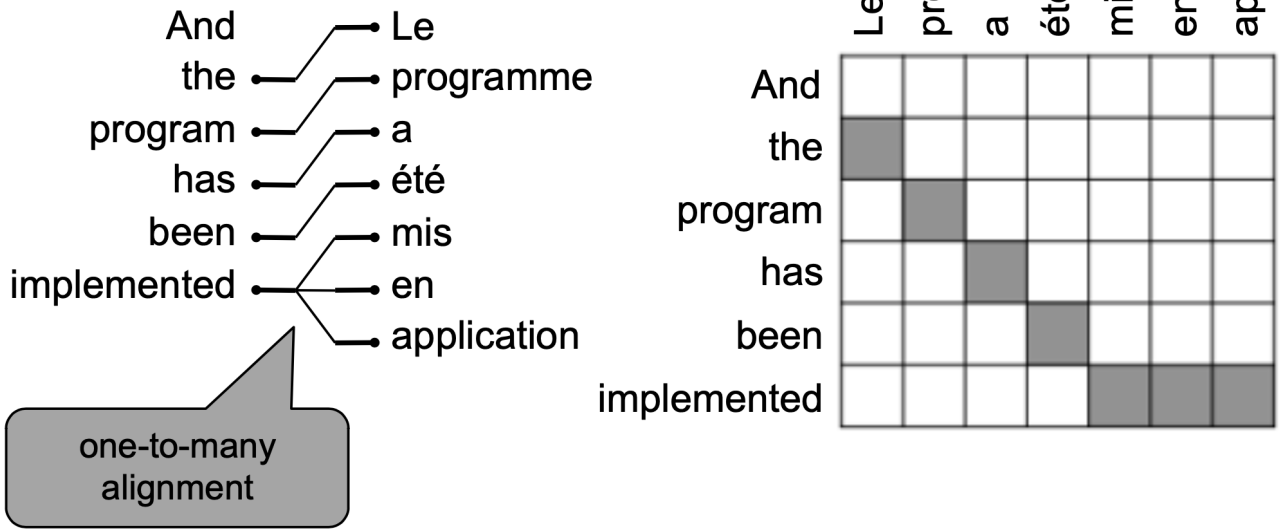
where a is the alignment (word level correspondence)

Example Alignment



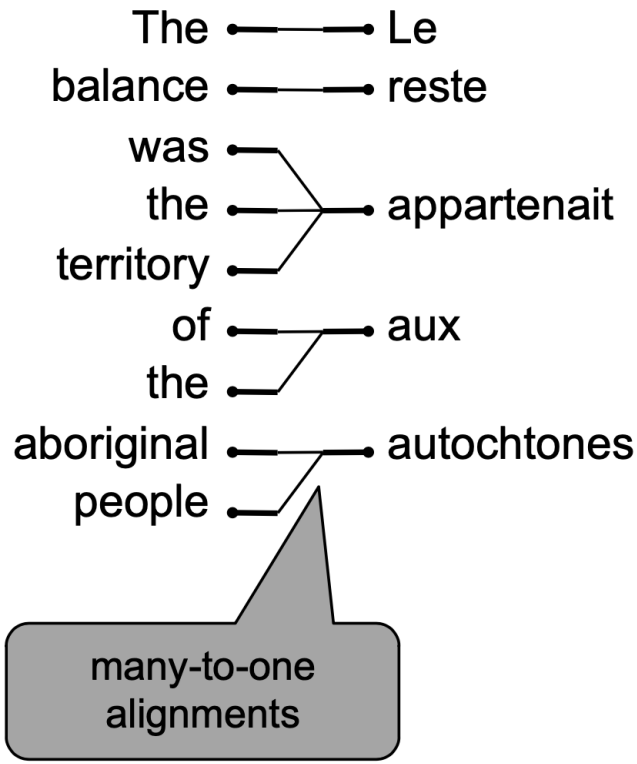
Stanford Linguistics 284

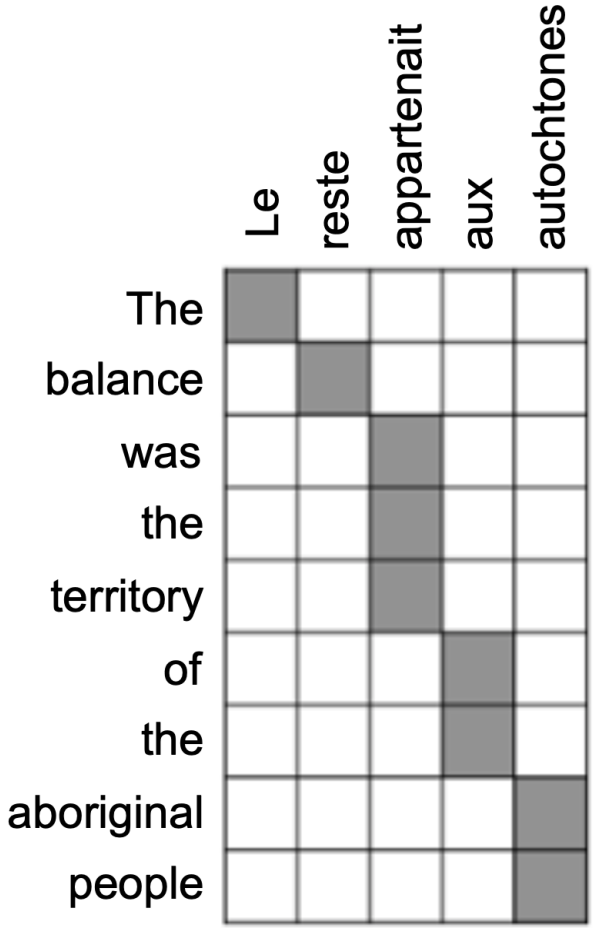
Example Alignment



Stanford Linguistics 284

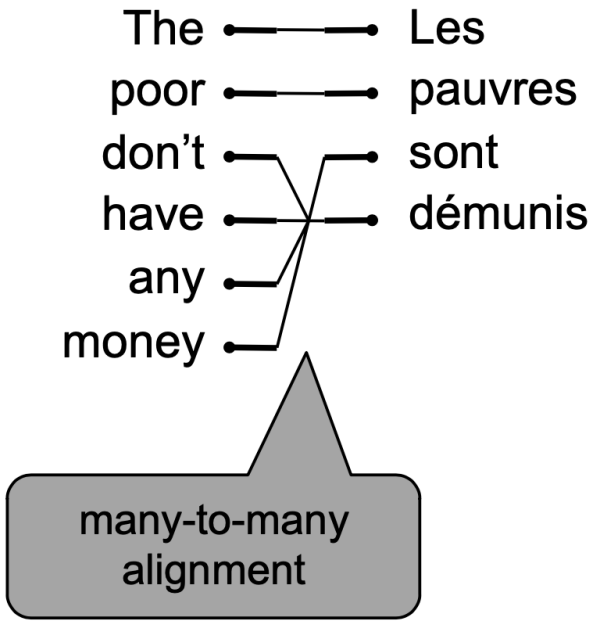
Example Alignment

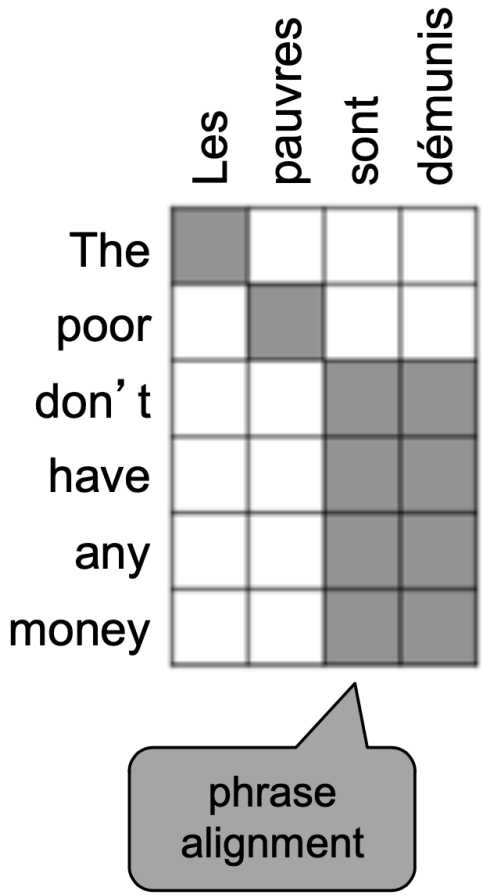




Stanford Linguistics 284

Example Alignment





Stanford Linguistics 284

Learning Alignment

STM learns $P(x, a|y)$ as a combination of many factors:

- Frequency with which particular words align (conditional on position in the sentence)
- Fertility of words (on average, how many words in the target language correspond to a given word in the input language)
- And a variety of other statistics computed from parallel data (complicated, human engineered models built up over many years)

Need parallel data on alignment (costly) and a bunch of human engineered statistics (a hard problem); NMT is going to help on both of these fronts

Statistical Machine Translation

$$\arg\max_y P(y|x) = \arg\max_y P(x|y)P(y) \tag{5}$$

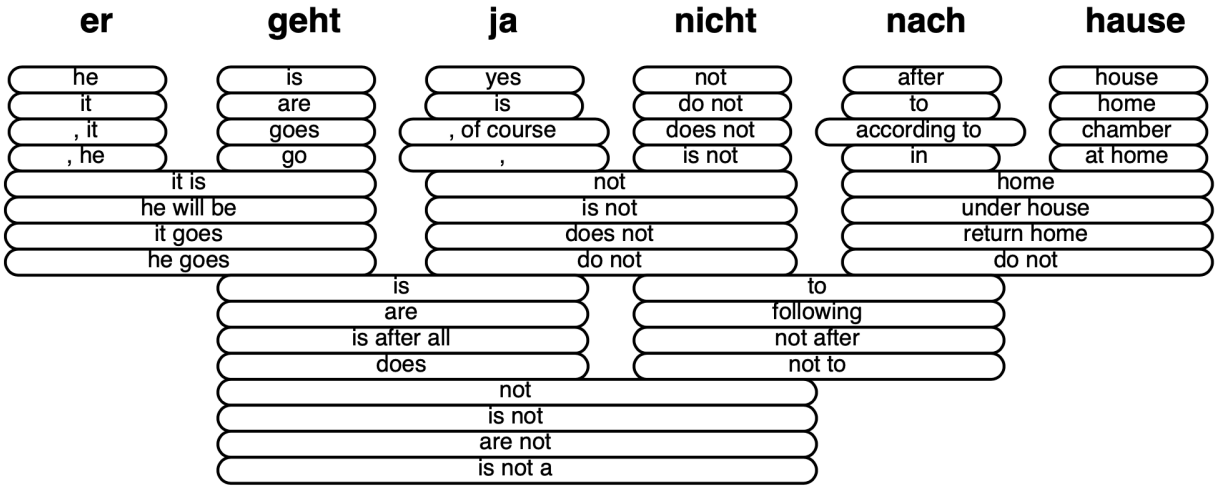
We have some intuition for how to compute $P(x|y)$ from alignment, and $P(y)$ is a language model, which we've already seen. How do we compute the $\arg\max$?

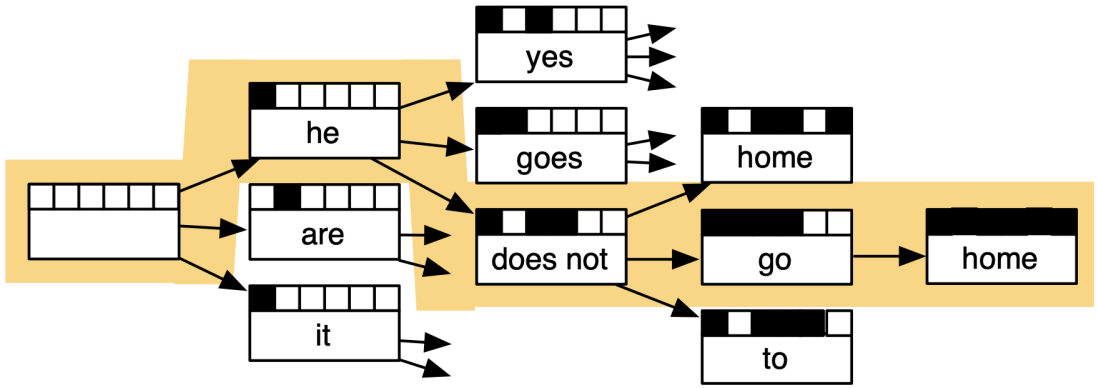
One idea would to be to calculate every possible y - i.e. every possible sentence in English - and compute the probability. Obviously that is computationally unfeasible.

Instead, use a heuristic search algorithm to search for the best translation, discarding possibilities that are too improbable

Decoding

The process of finding the best translation is called decoding. Think of this as considering different hypotheses in a tree, but prune the tree as you go





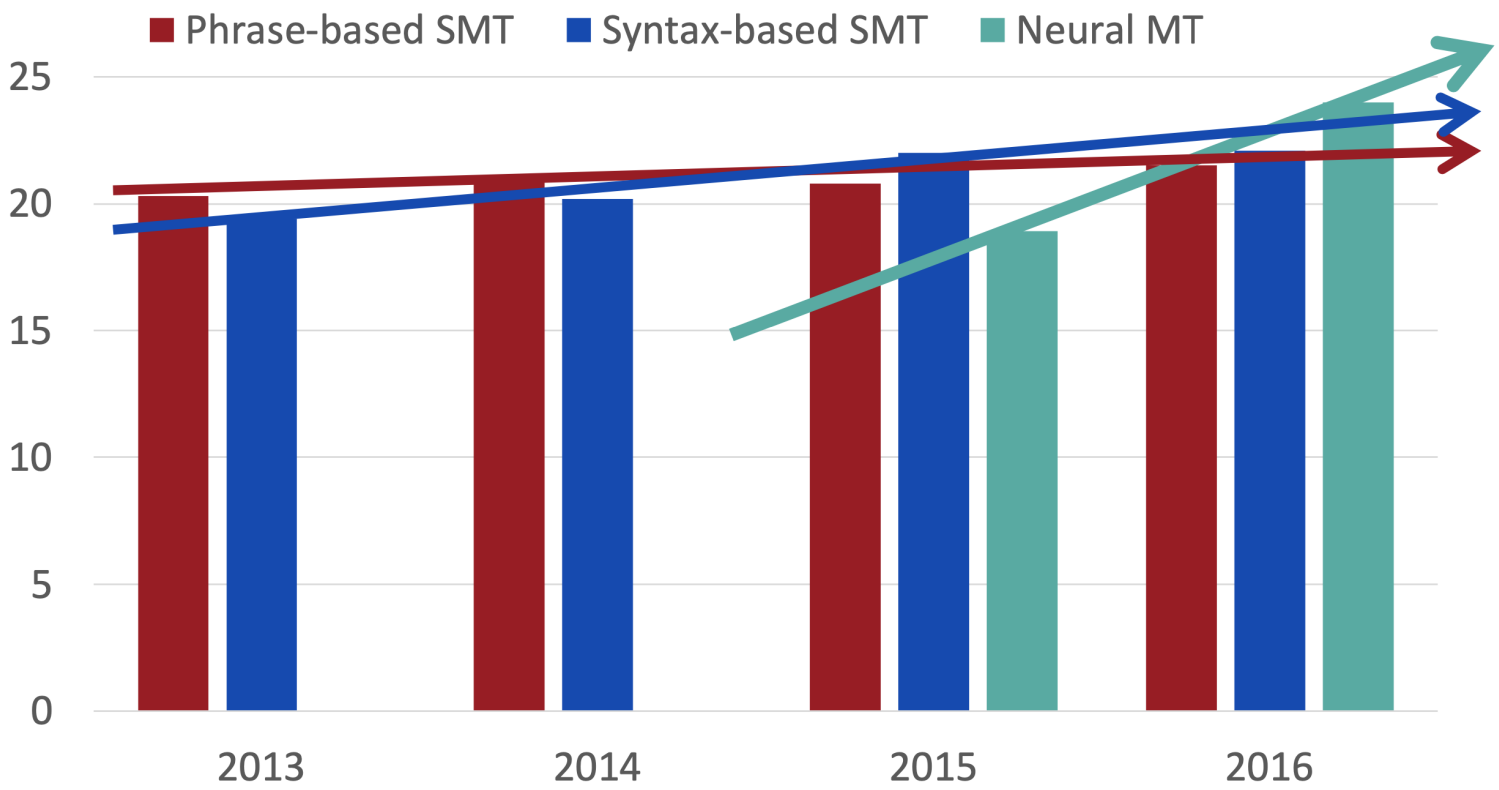
Stanford Linguistics 284

Statistical Machine Translation

- The SoTA statistical translation models were extremely complex
  - Hundreds of important details that are beyond the scope of this overview
  - Systems had many sub-components
  - Tons of features engineering
- Large number of people worked on designing these systems over many years

Outline

Machine Translation Evaluation



Source: http://www.meta-net.eu/events/meta-forum-2016/slides/09_sennrich.pdf

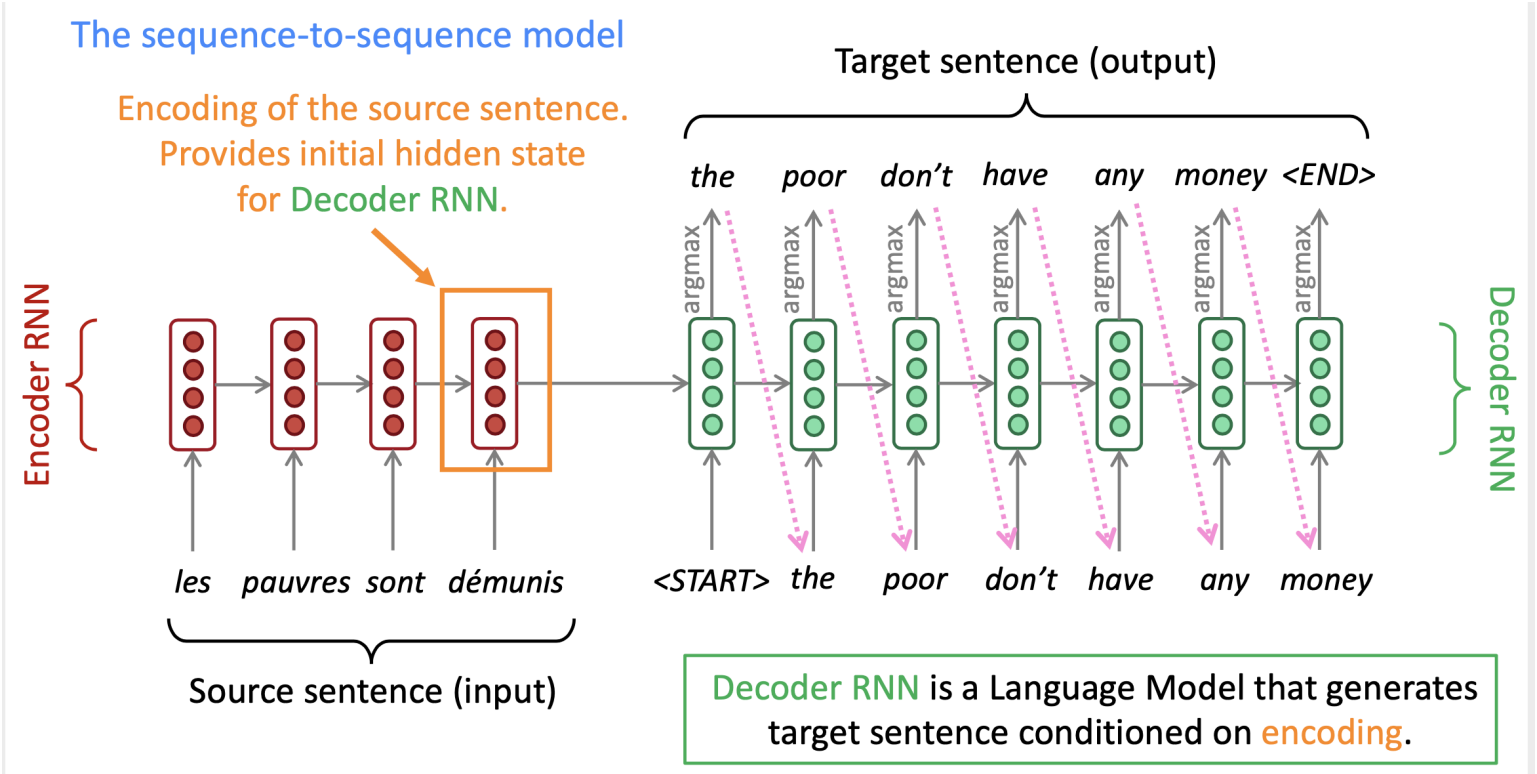
MT Benchmarks

- The benchmark dataset in this domain is BLEU (Bilingual Evaluation Understudy)
- Compare machine translation to several human translations and compute similarity by comparing n-grams and penalizing translations that are too short

Machine Translation Evaluation

- This is pretty mind-blowing, but is the same theme that we've seen elsewhere in the course
- Statistical translation systems engineered by hundreds of people over many, many years beat by a neural machine translation model engineered by a few people over a few months
- It would take an entire course to teach statistical machine translation in detail. After this lecture, you could implement a NMT model with comparatively much more modest effort
- **Key Course Theme**: Deep learning makes models both more accurate and way more efficient to develop (popular concept of a 10x or 100x engineer)
- So how does NMT work?

Sequence-to-sequence (test time)



Stanford Linguistics 284

Could also use for tasks like summarization

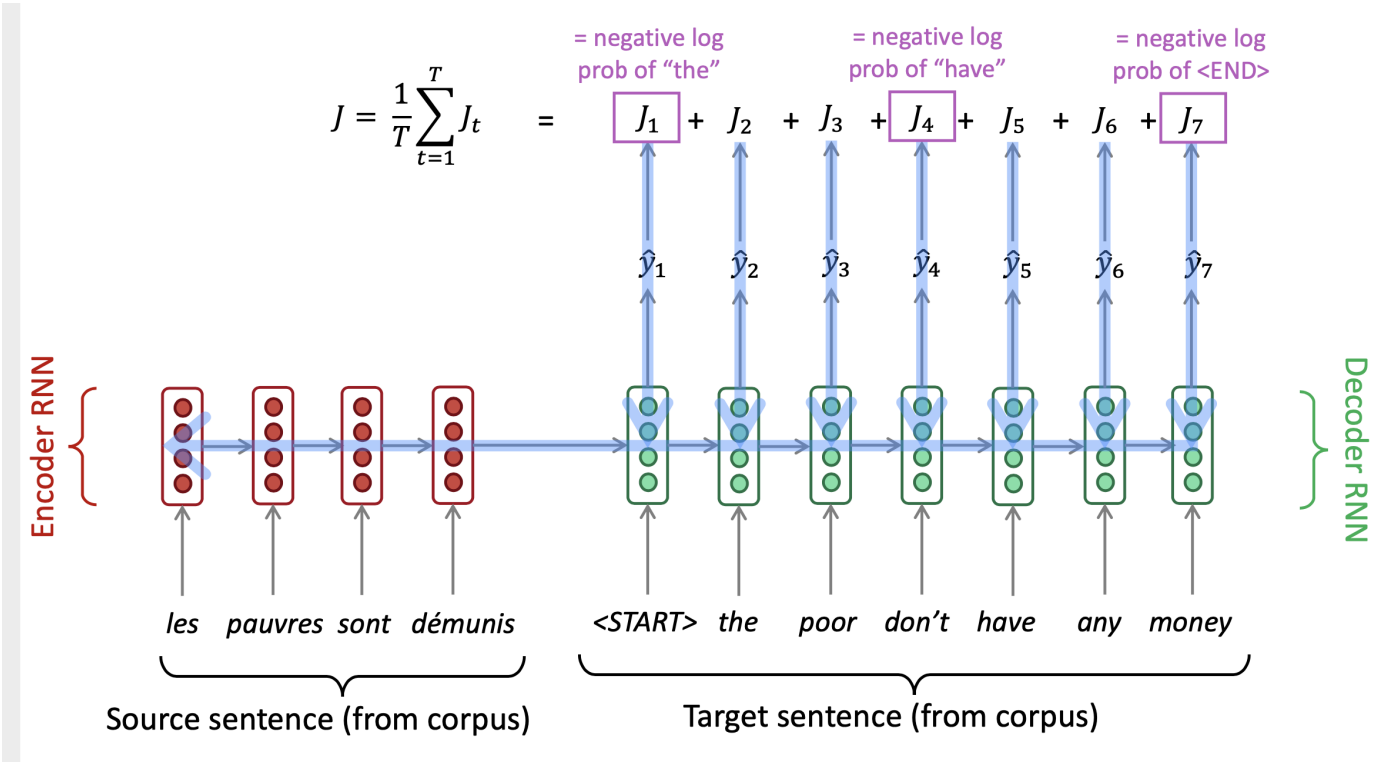
Seq2seq is a conditional language model

- The decoder portion is a language model (it predicts the next word)
- It's condition because its predictions are conditioned on the output from the encoder (which is fed the source sentence)

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

- This is the definition of a language model from last class, but now conditional on x (the input sentence)
- Directly learn this in NMT, unlike SMT

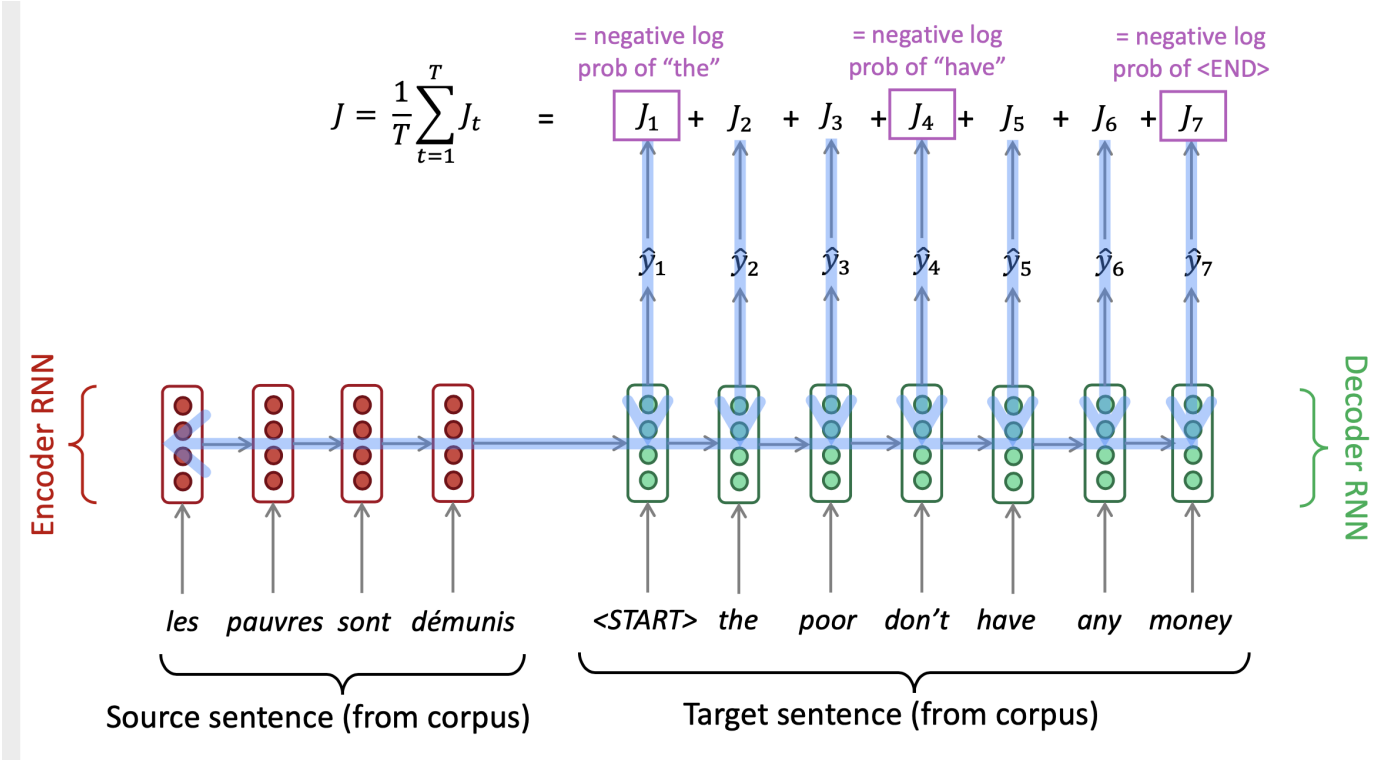
How do we train Seq2seq?



Stanford Linguistics 284

Seq2seq is trained end-to-end.

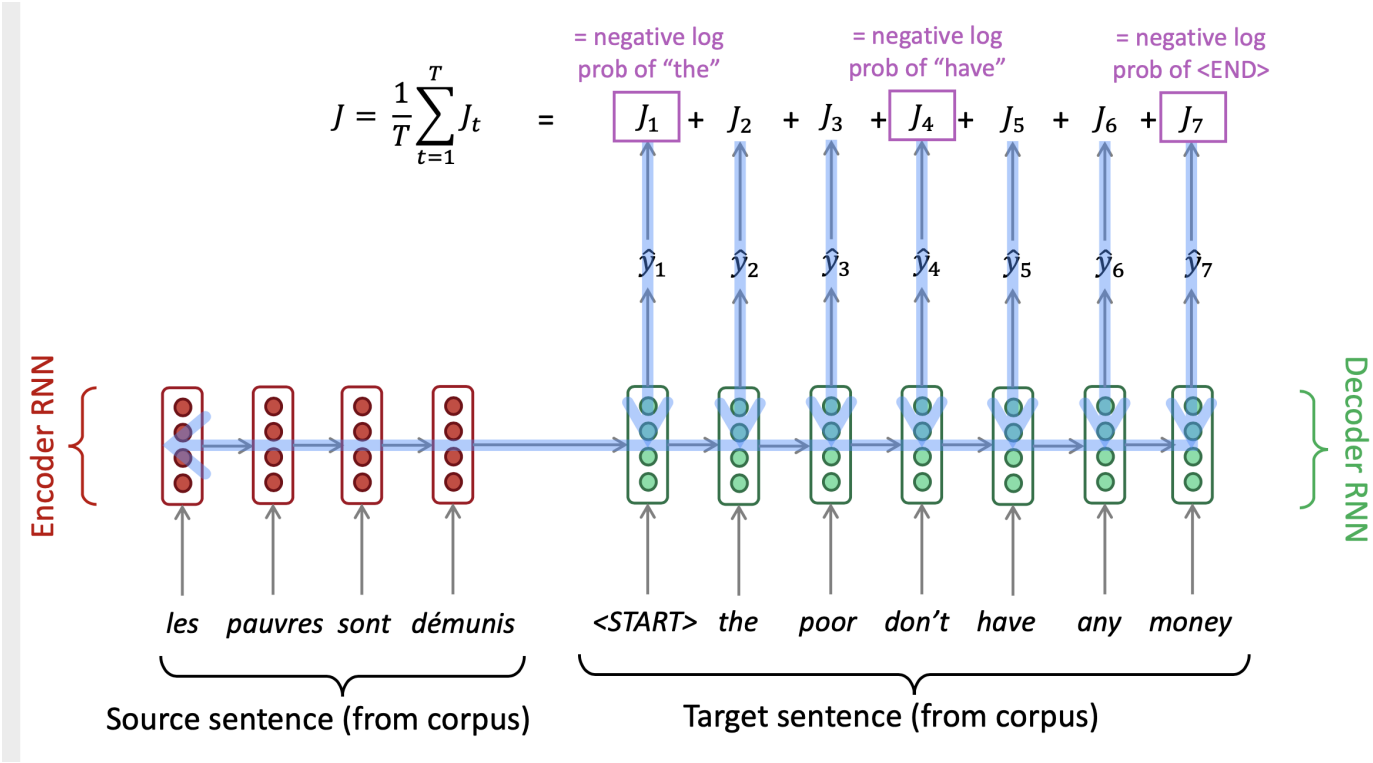
How do we train Seq2seq?



Stanford Linguistics 284

Will need two sets of word embeddings, to feed the French words into the RNN and output the English words (can fine-tune or freeze).

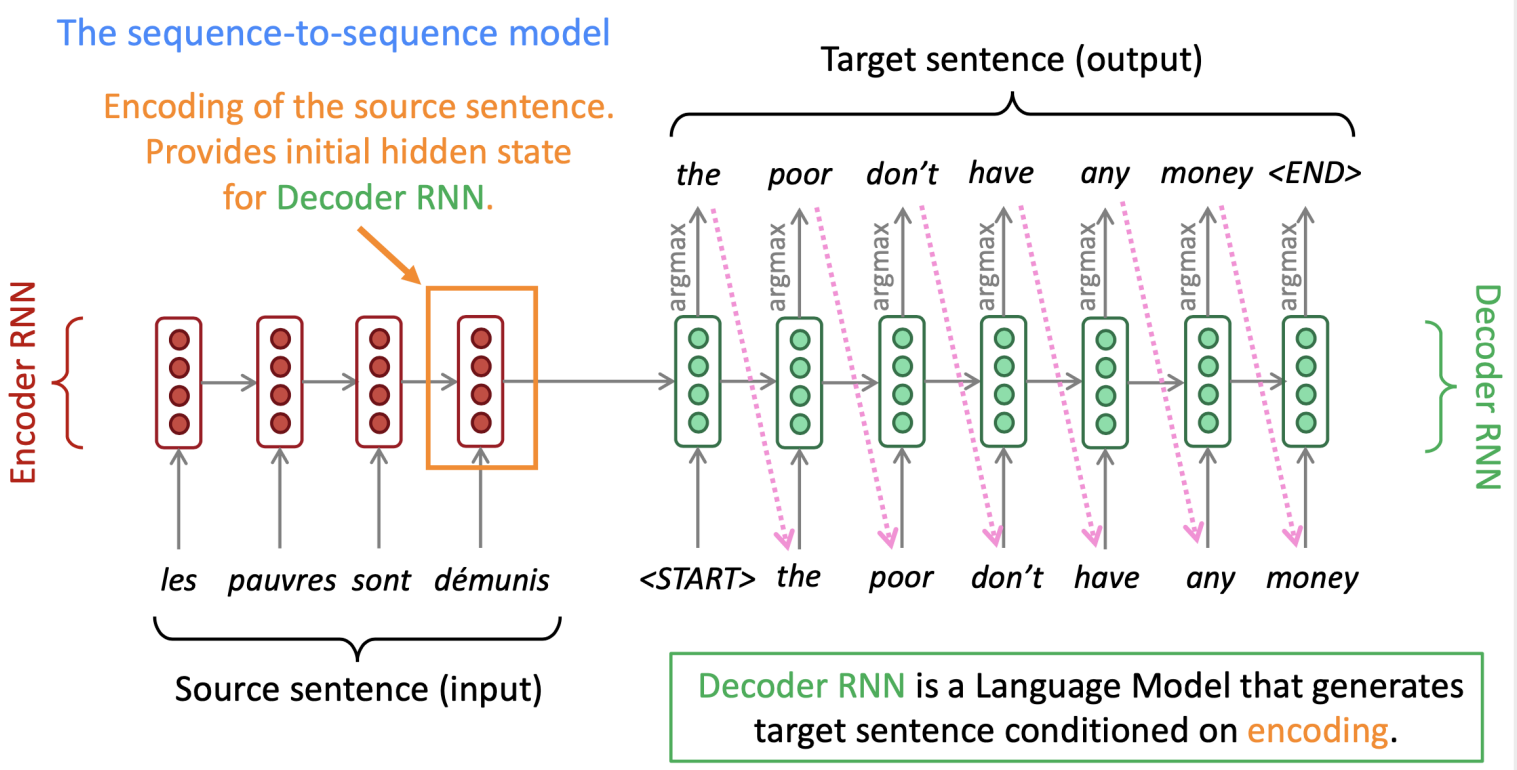
How do we train Seq2seq?



Stanford Linguistics 284

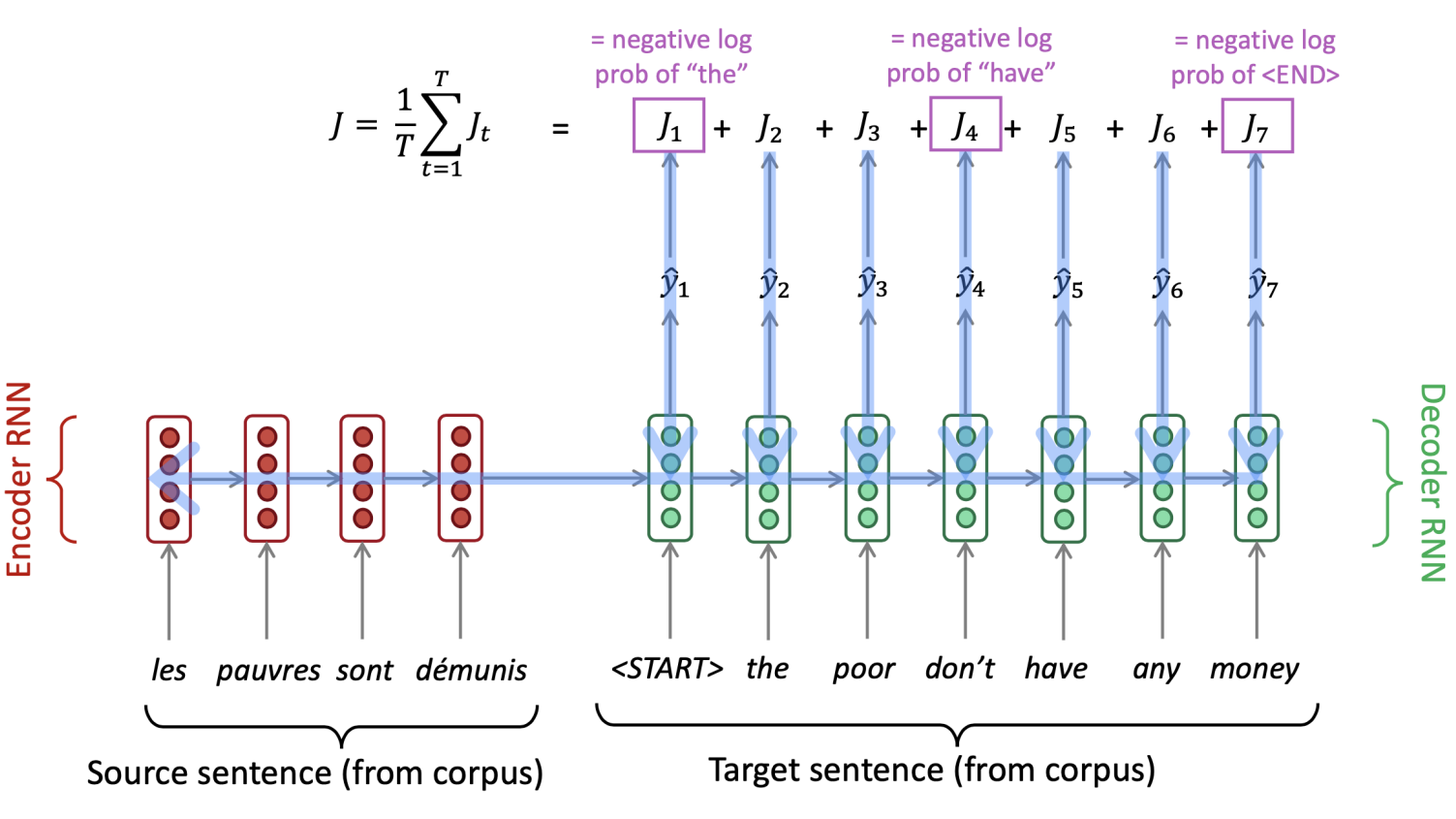
Unlike at test time, don't feed output back in. Feed in the target sentence.

Seq2seq inference



Stanford Linguistics 284

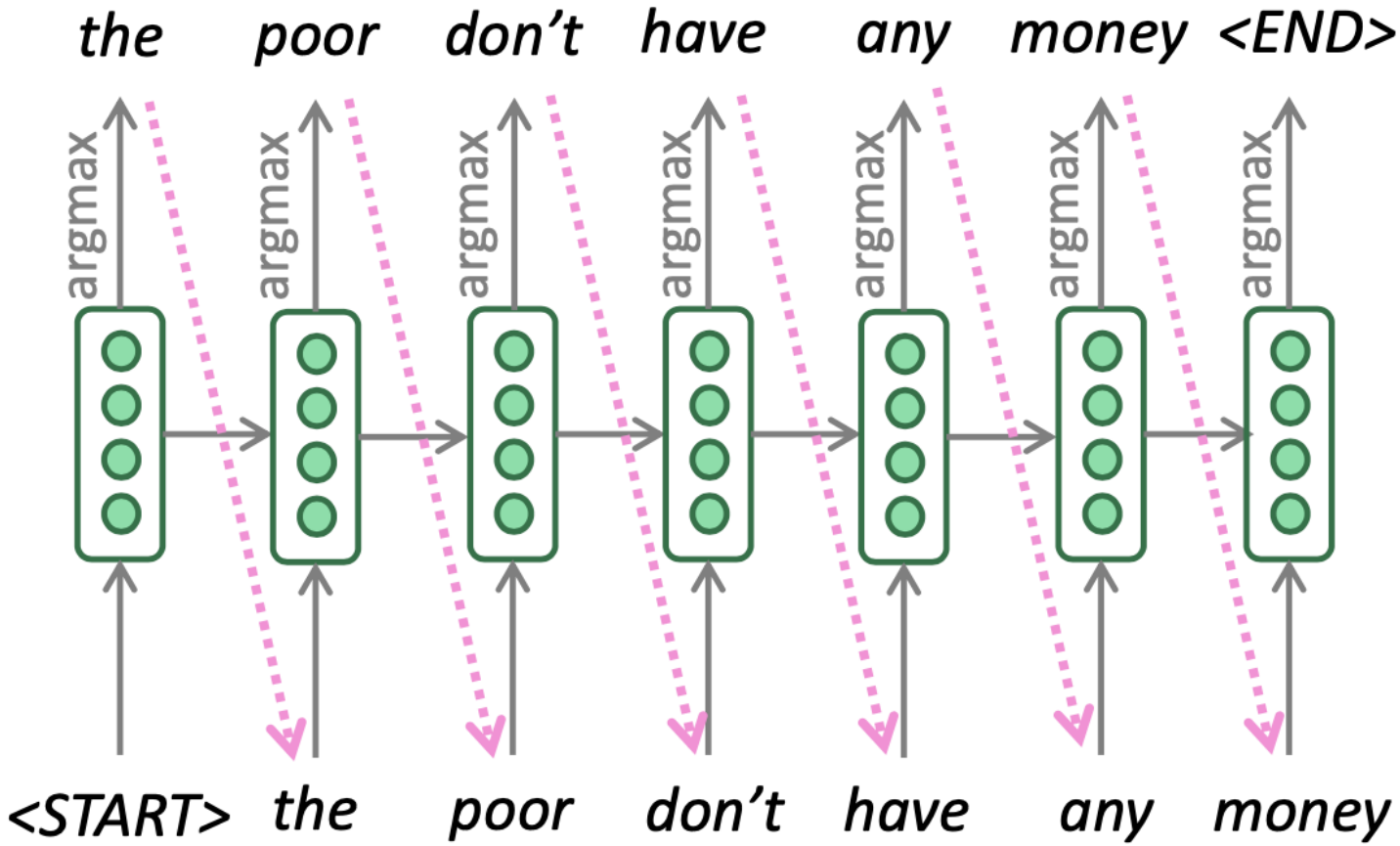
Seq2seq Training



Stanford Linguistics 284

How to decode: greedy decoding

Greedy decoding takes the $\arg\max$ at each step of the decoder



Stanford Linguistics 284

But this leaves you no way to undo decisions

Decoding

- At the other extreme, could consider all possible translations
- At each step t, have to track V^t possible translations, where V is the size of the vocabulary
- Completely unfeasible

Beam Search Decoding

- At each step, keep track of the k most plausible partial translations (hypotheses)
- k is the beam size
- The plausibility score is just the probability of the sequence:

$$logP(y_1 ... y_t|x) = \sum_{i=1}^{t} logP(y_i|y_1 ... y_{i-1}, x)$$

See beam search used a lot in NLP

Beam Search

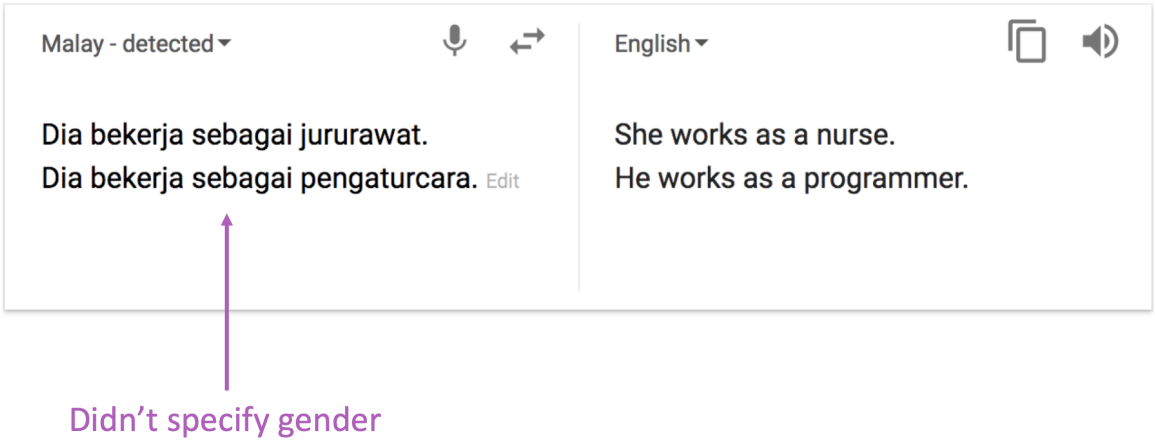
- When a hypothesis produces the $< End >$ token, put it aside
- Continue exploring other hypotheses
- Stop either once reach time step t (hyperparameter) or once collected at least N (hyperparameter) completed hypotheses

Beam Search

- Longer hypotheses will have worse scores
- Fix this by normalizing score by length when selecting the final hypothesis
- Don't need to do this earlier, because only comparing hypotheses of the same length

Neural Translation

As anyone who has ever used Google Translate knows, there is still today scope for improvement. Also, NMT has biases reminiscent of those we saw last class



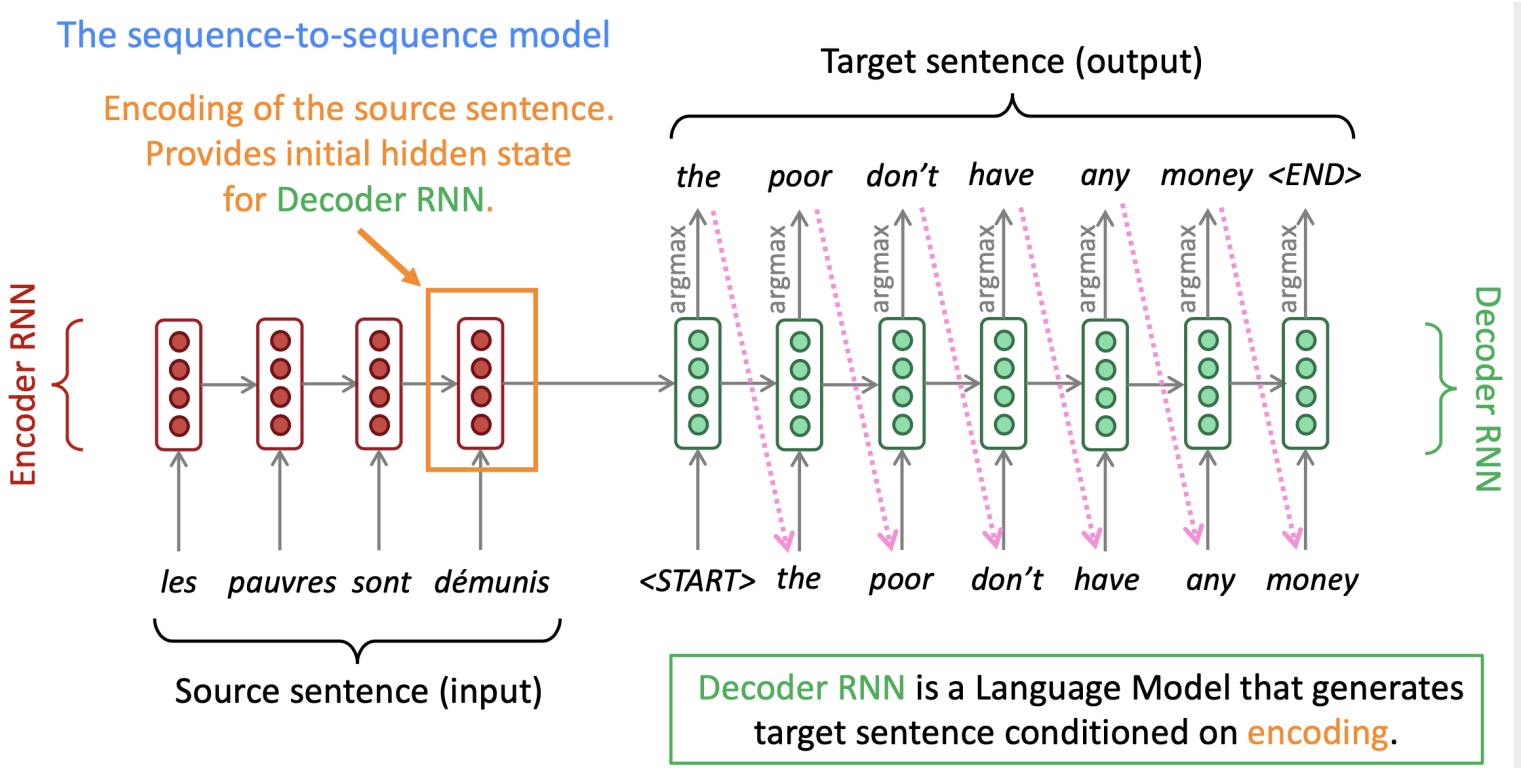
Stanford Linguistics 284

Training corpora likely have bias, relative to occupational patterns. There are many situations where the language you are speaking conditions how much information you provide, most commonly with gender (also social status)

Neural Translation

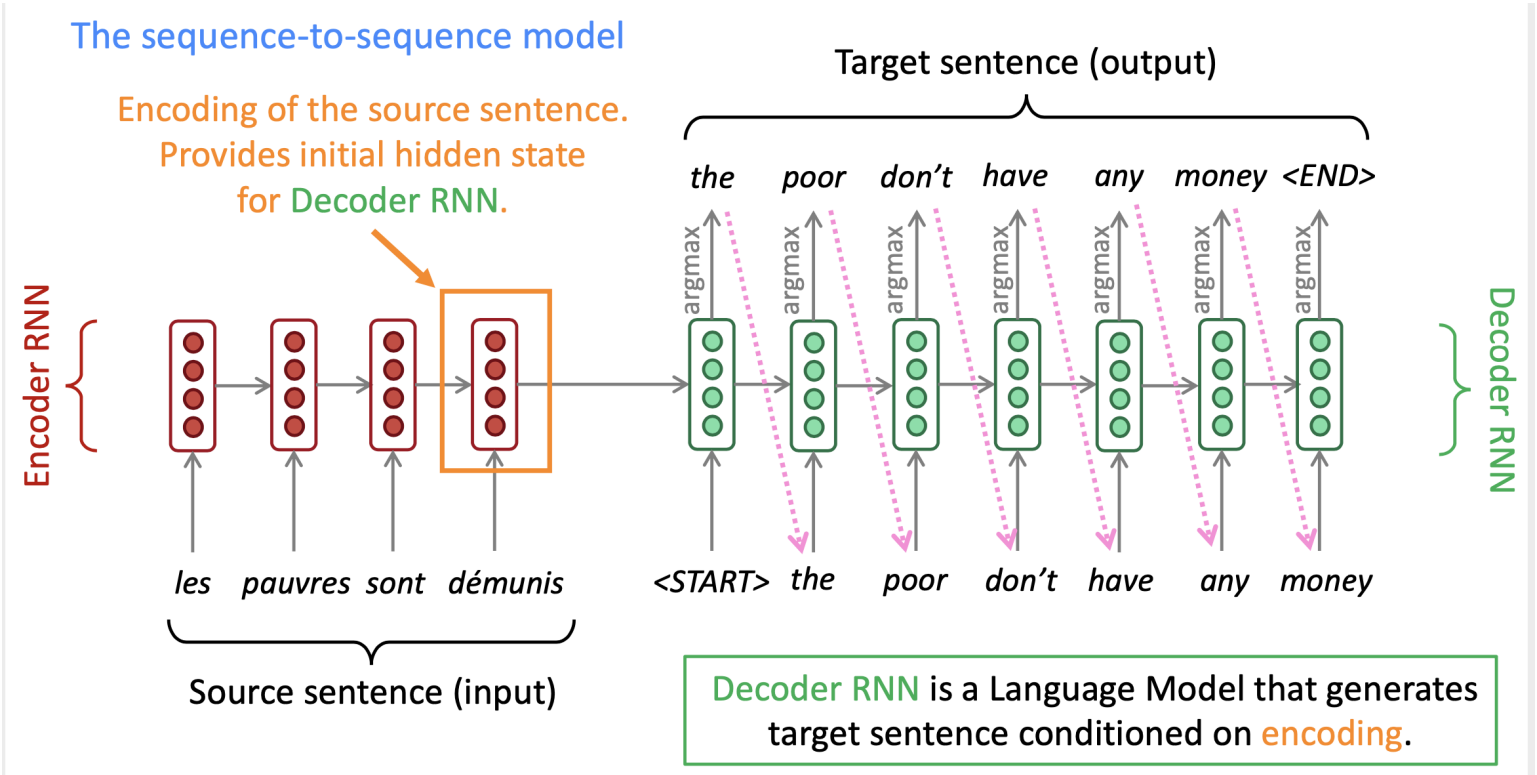
- There have been many improvements to the initial seq2seq model that we just saw
- One is particularly fundamental (and a major motivation for why we are talking about neural translation)

What is the Problem with the Seq2seq Architecture?



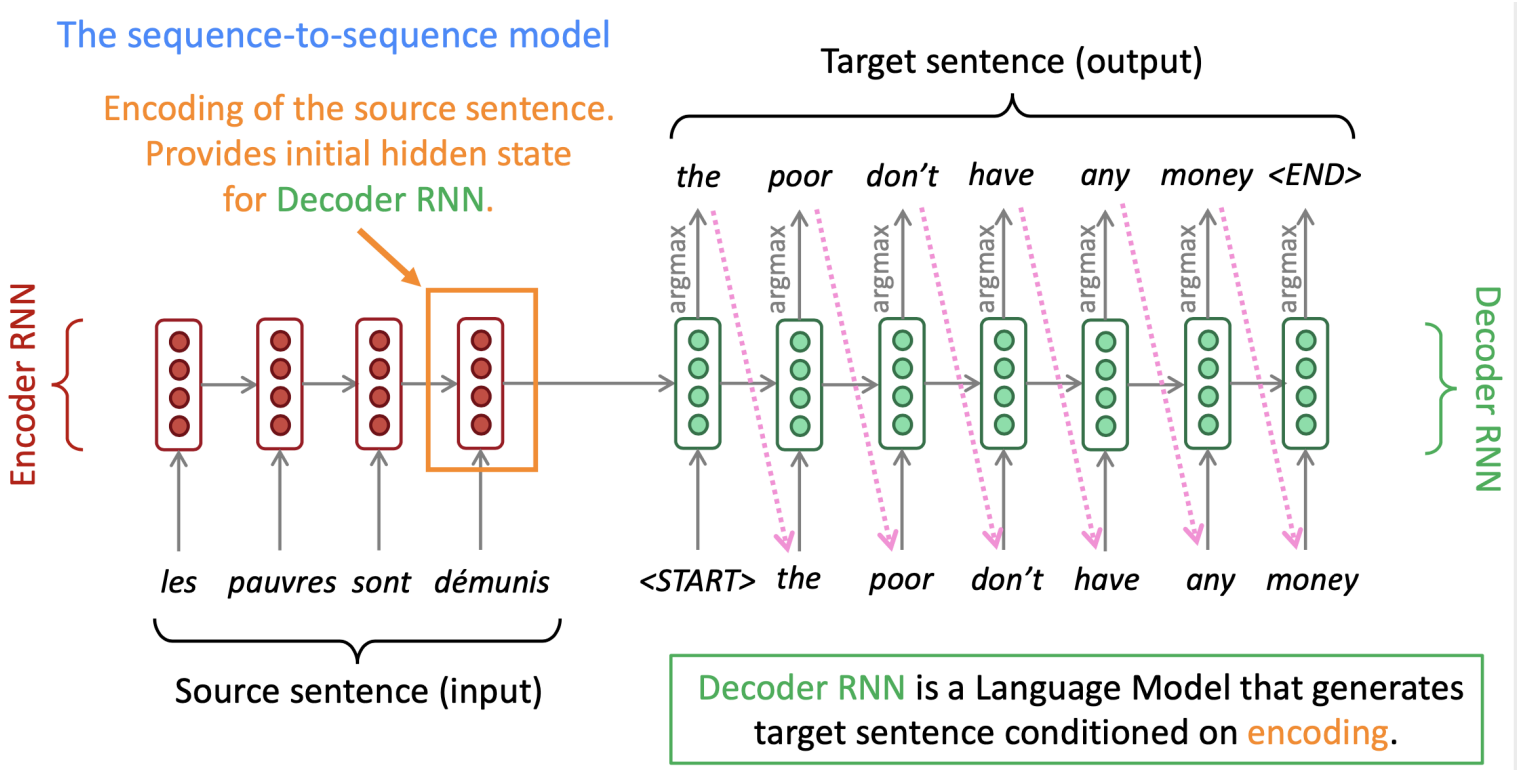
Forcing all information about the sentence to be encoded in that final vector from the encoder, because that is the only thing that gets passed to the decoder

What is the Problem with the Seq2seq Architecture?



Remember at its core a neural network is a bunch of matrix algebra operations

What is the Problem with the Seq2seq Architecture?

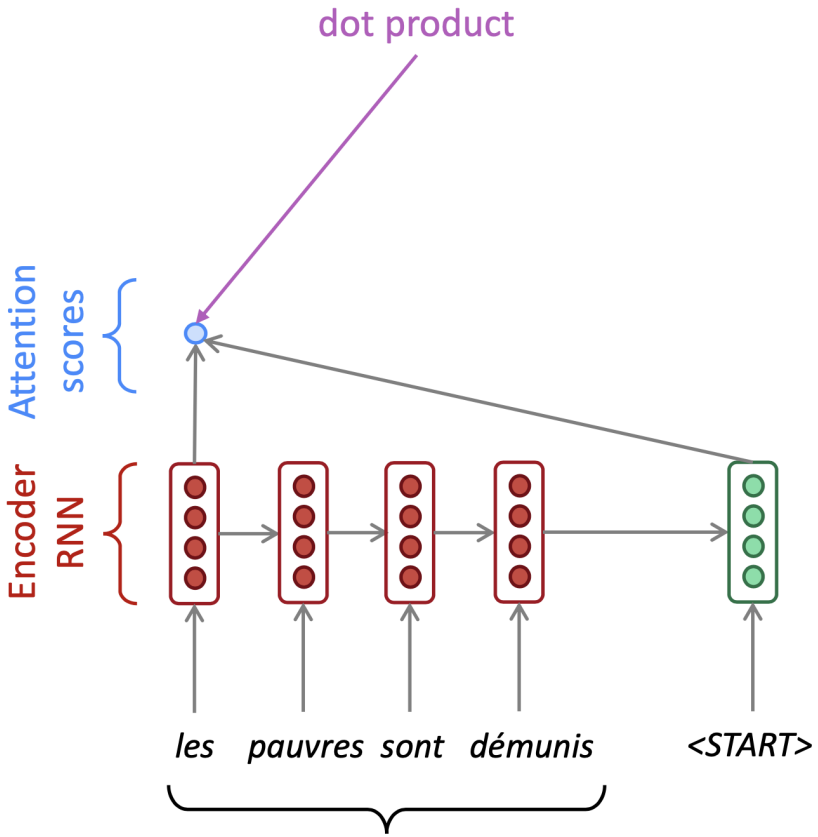


Cannot just concatenate all the encoder hidden states and send to the decoder because this would be a variable sized vector

Attention

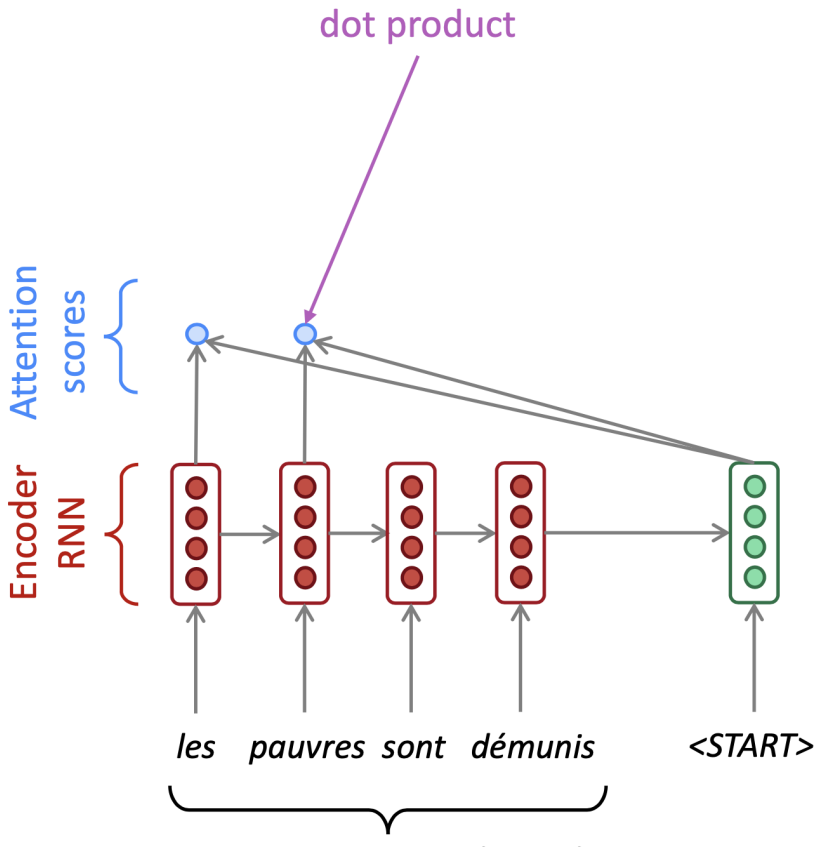
- At each part of the decoder, use a direct connection to the encoder to focus on a specific part of the source sentence
- Effectively allows us to pass information from a variably sized encoder model to the decoder, in a learnable way
- Attention has transformed NLP

Attention



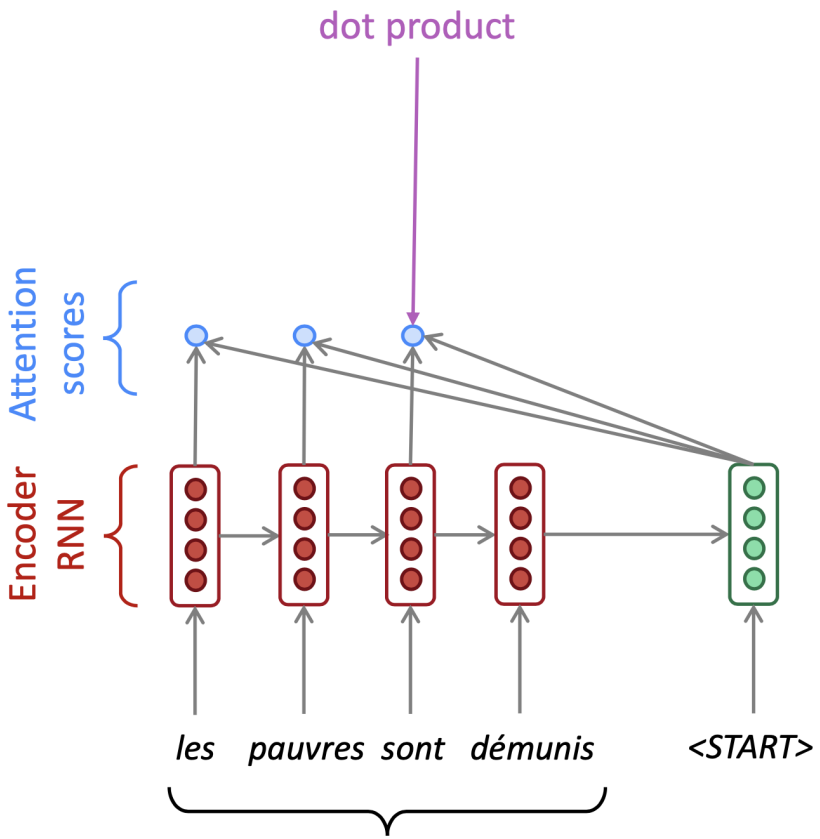
Stanford Linguistics 284

Attention



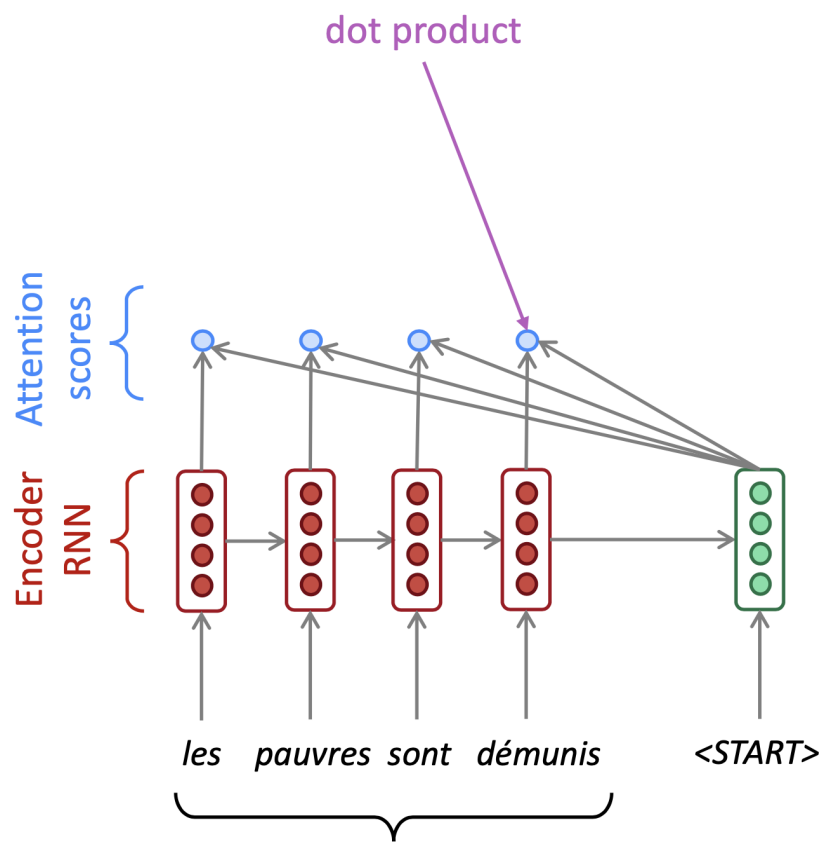
Stanford Linguistics 284

Attention



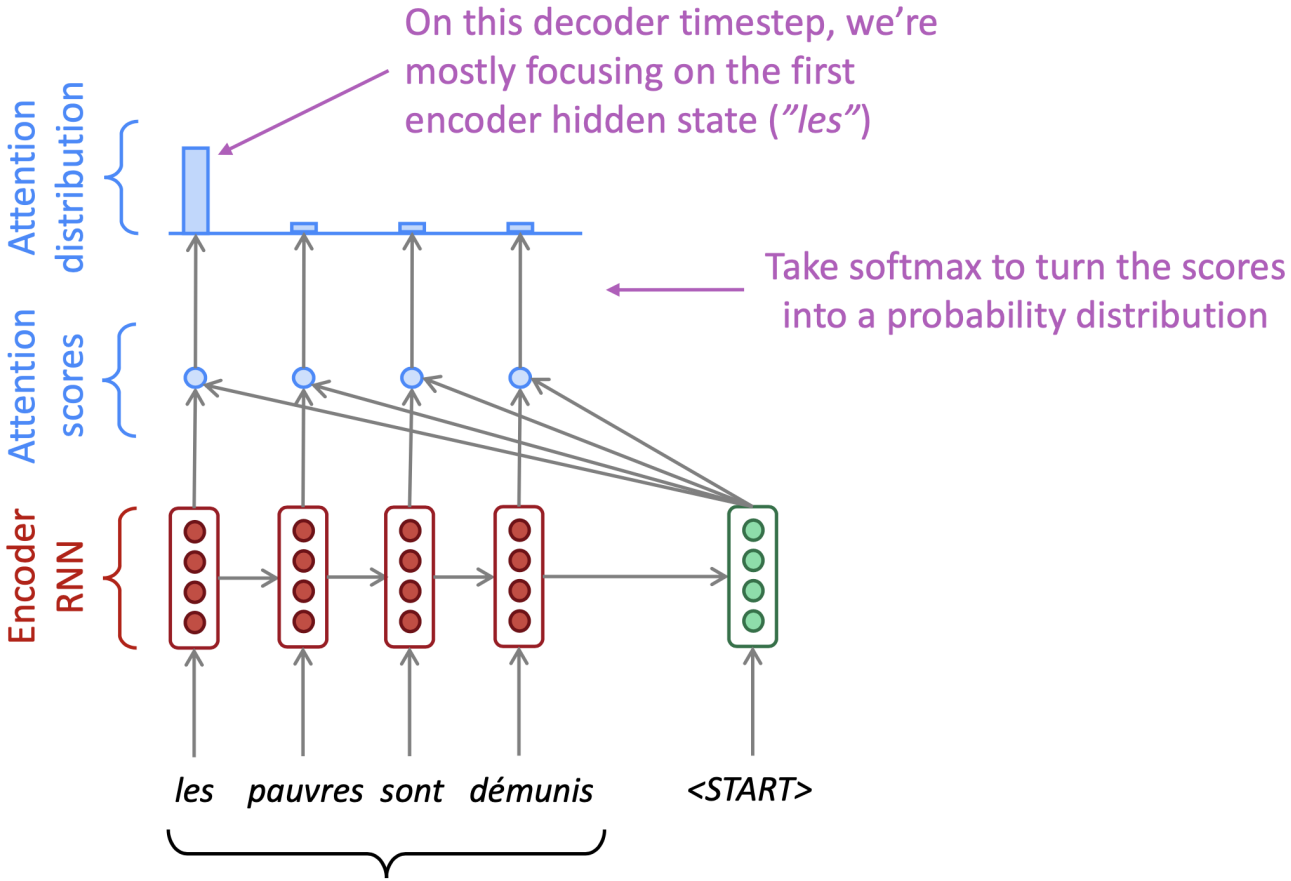
Stanford Linguistics 284

Attention



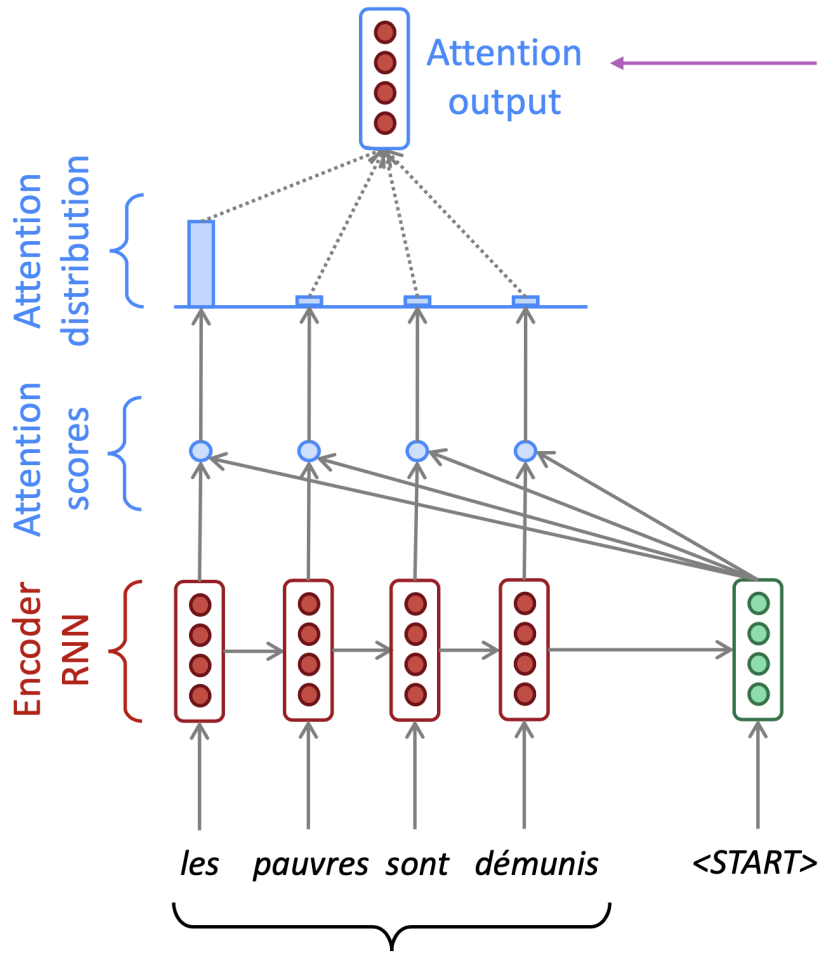
Stanford Linguistics 284

Attention



Stanford Linguistics 284

Attention

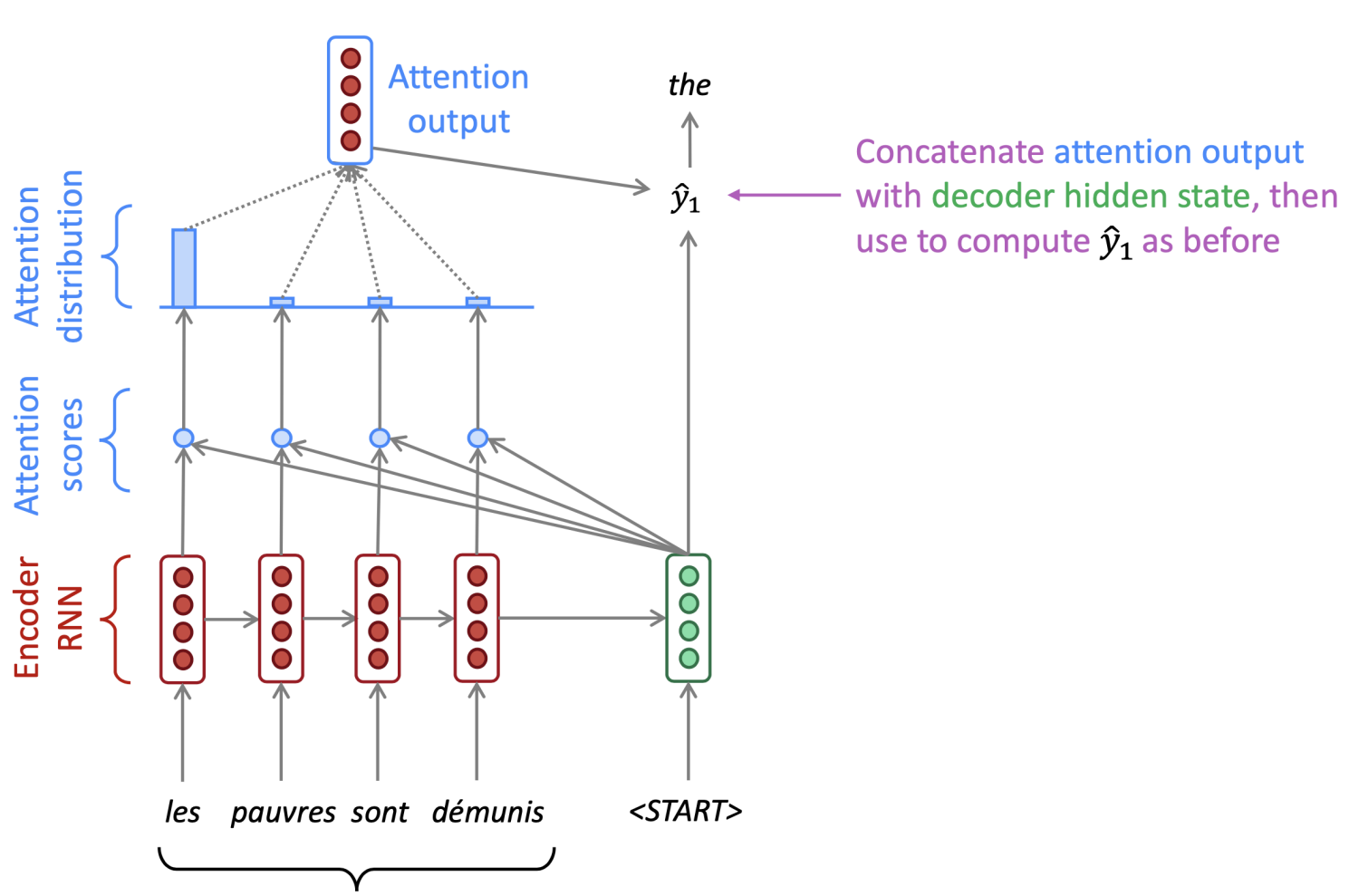


Use the attention distribution to take a **weighted sum** of the encoder hidden states.

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

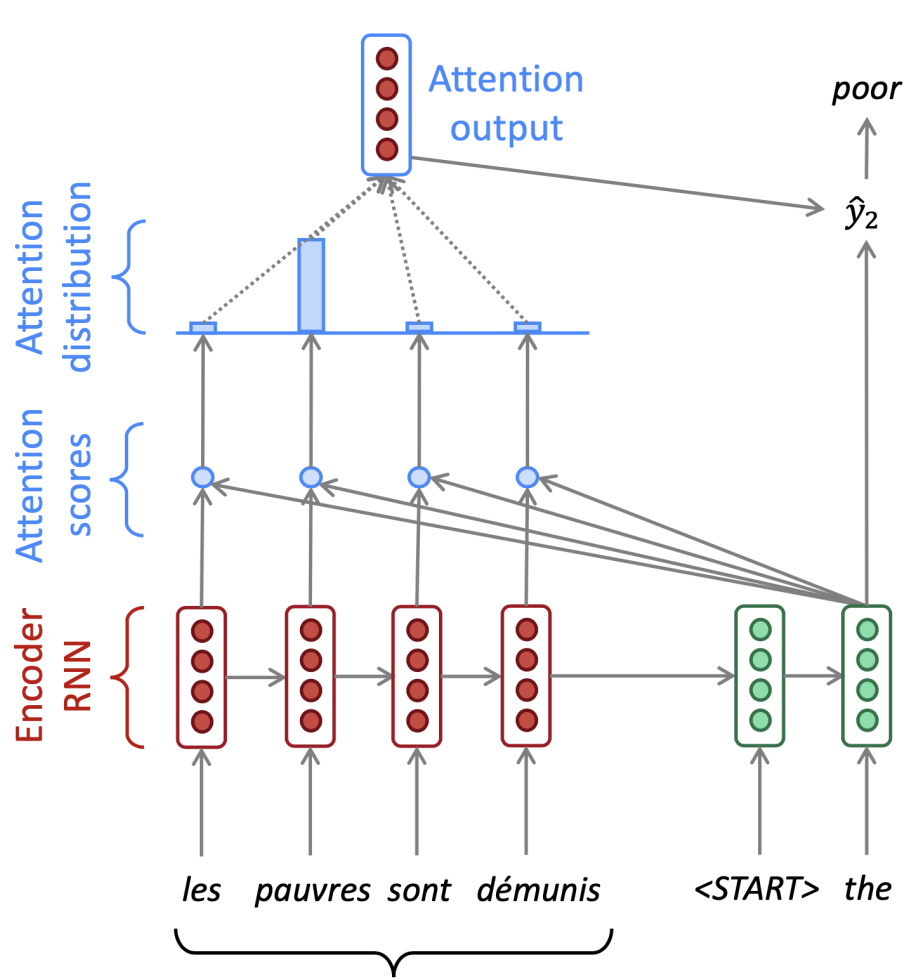
Stanford Linguistics 284

Attention



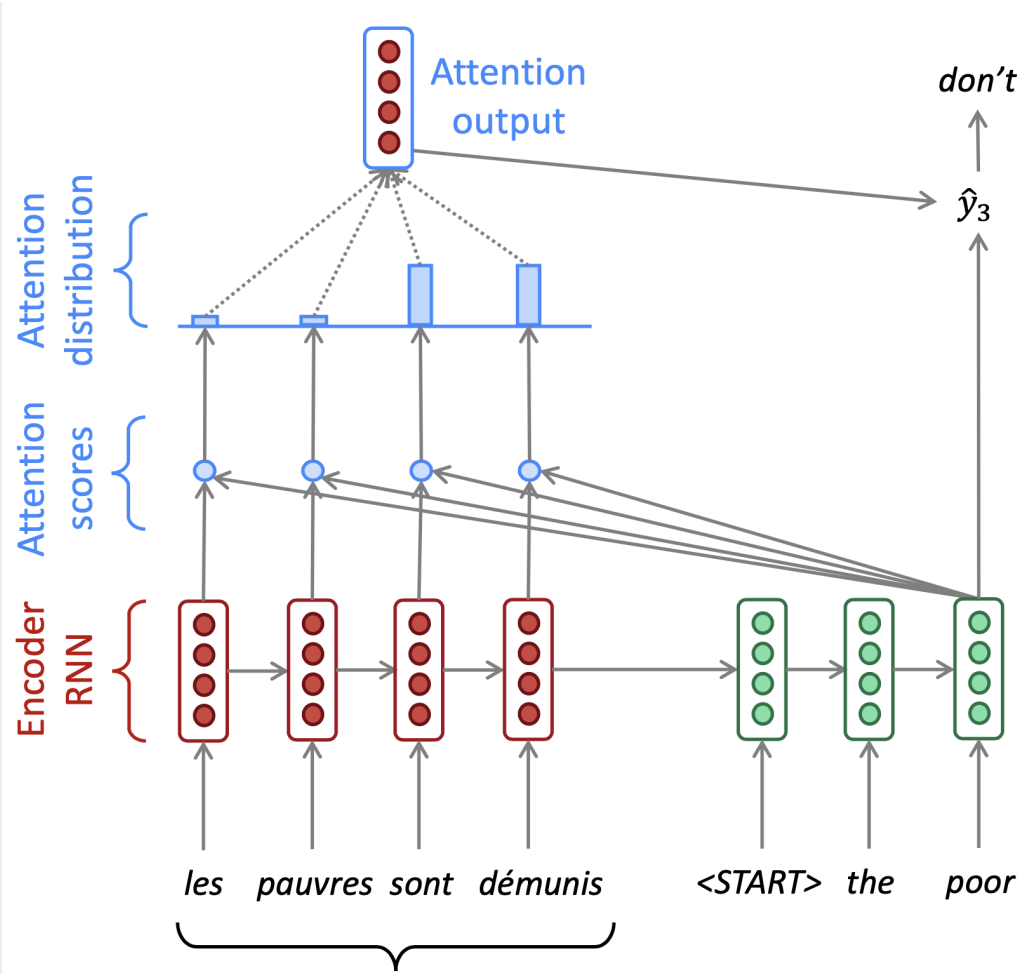
Stanford Linguistics 284

Attention



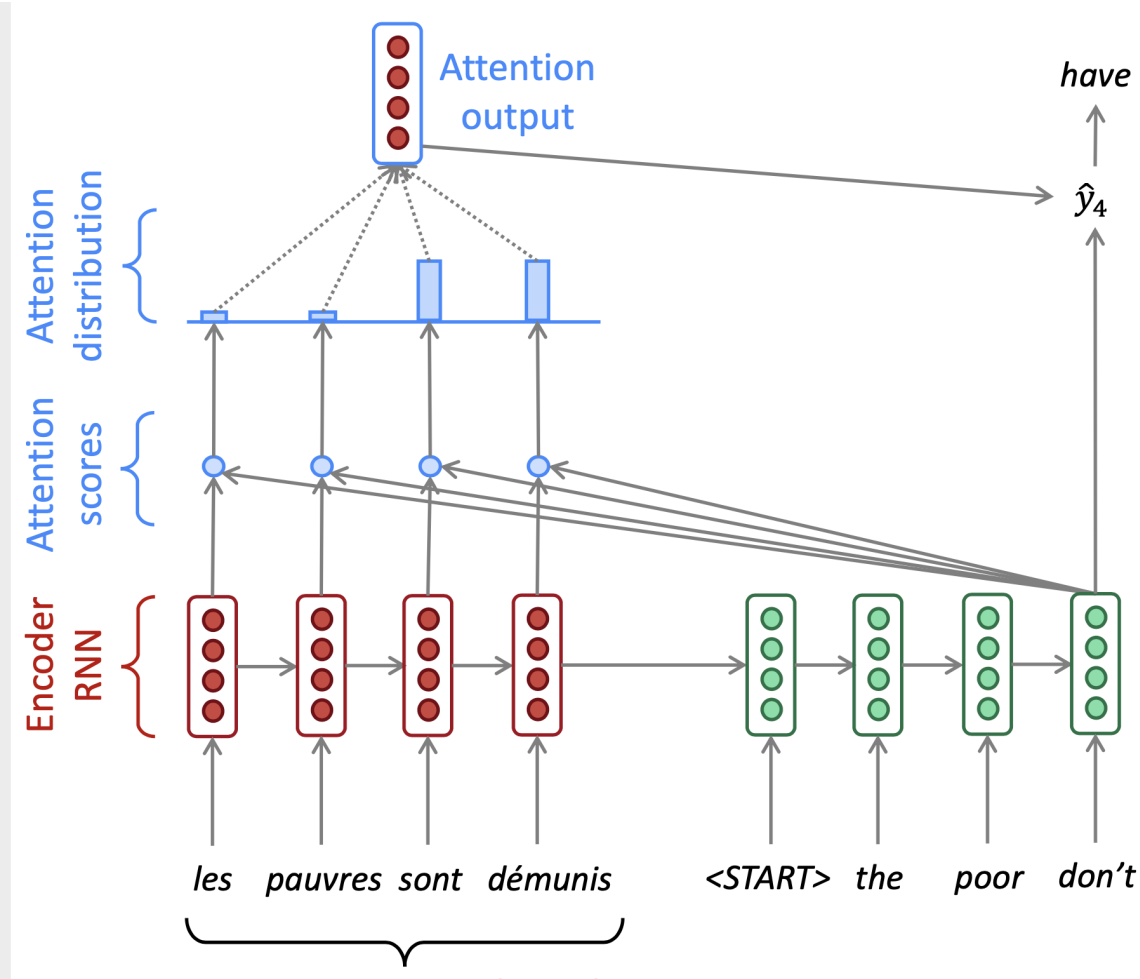
Stanford Linguistics 284

Attention



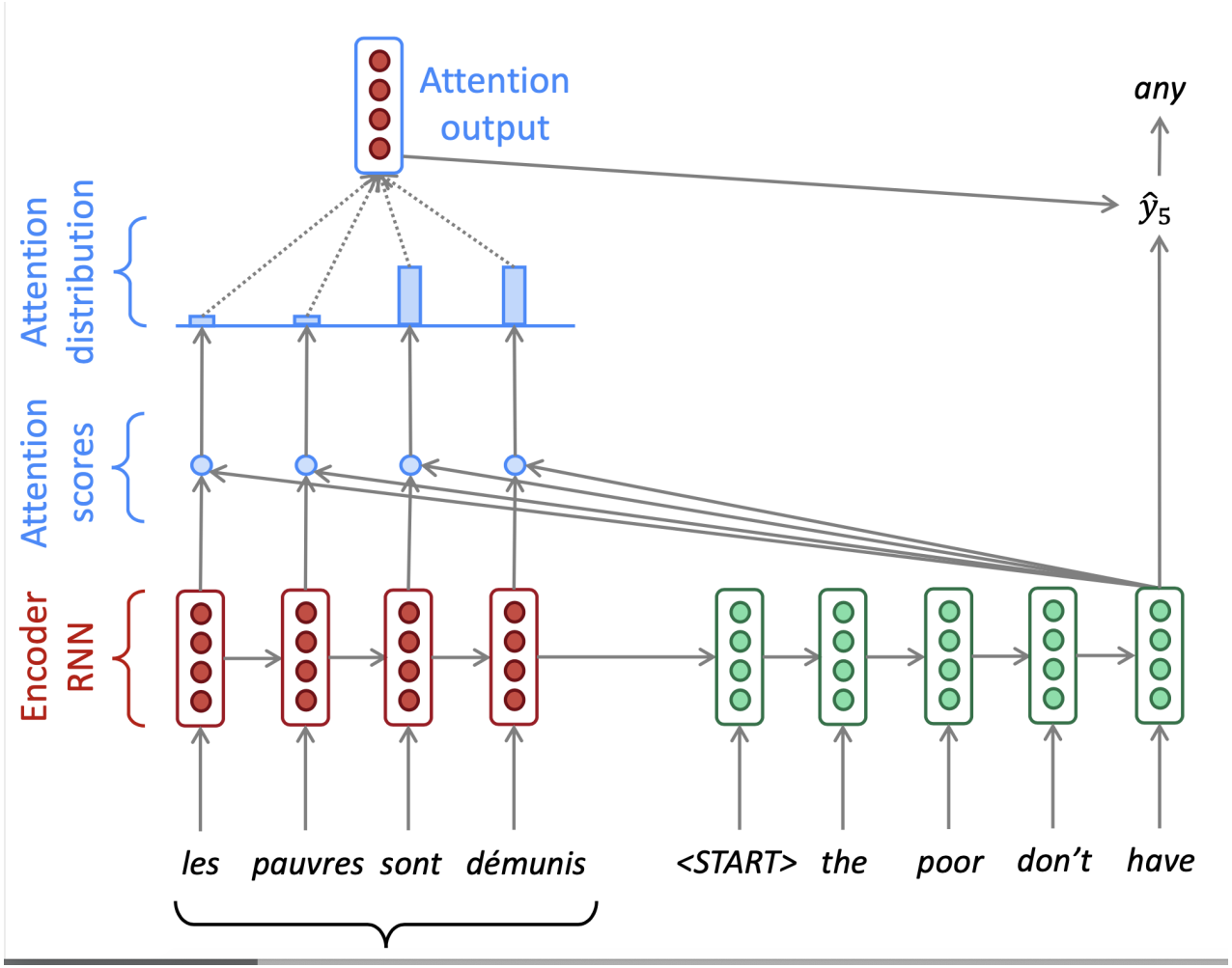
Stanford Linguistics 284

Attention



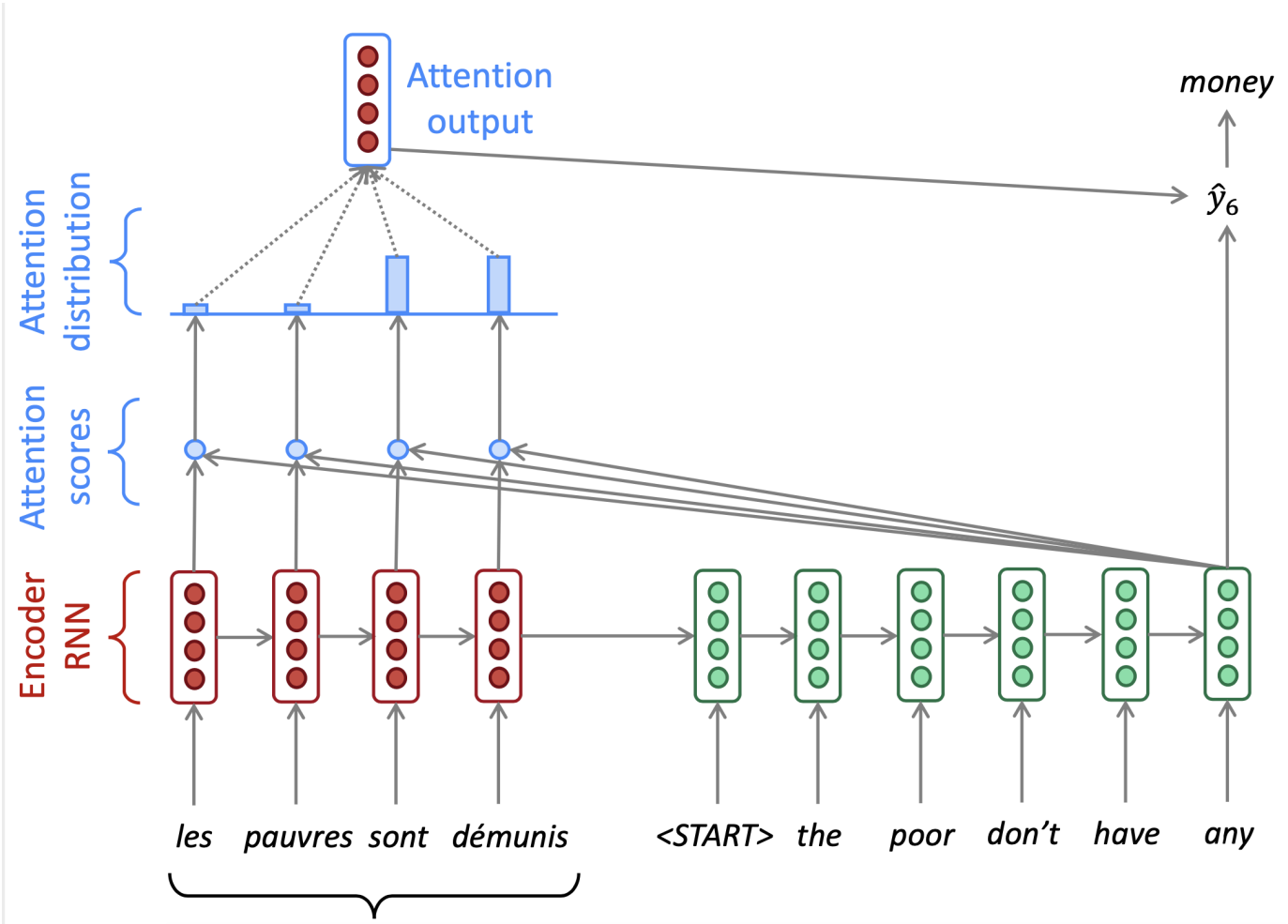
Stanford Linguistics 284

Attention



Stanford Linguistics 284

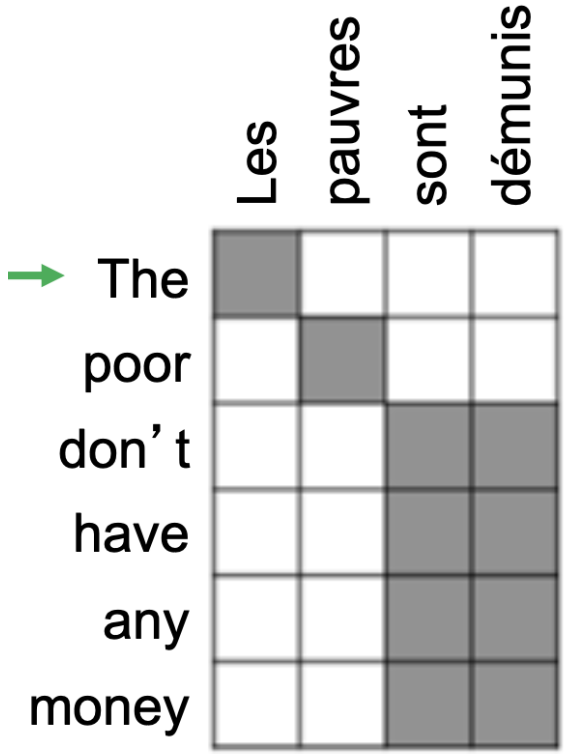
Attention



Stanford Linguistics 284

Attention and Alignment

What does the attention distribution look like?



Stanford Linguistics 284

We're not training the model to give us alignment, it learns this through attention. Don't need labels for alignment, which are very costly to produce.

Other Advantages of Attention

- Solves the bottleneck problem
- Helps with vanishing gradients by providing a connection to far away states (another **recurring theme** of this course, when it comes to discussing major advances in the literature; goes back to the fact that at its core, DL is linear algebra and MV calculus)

Attention and Deep Learning

- Attention has implications far beyond MT
- **Attention definition:** Given a set of vector *values* and a vector *query*, attention computes a weighted sum of the values that is dependent on the query
  - The query tells us where to focus in creating a weighted sum of the values
- In short, attention creates a *fixed-size* representation of an arbitrarily sized set of representations, given other representations (the query)

Attention Variants

Suppose $h_1 ... h_n$ (i.e. our encoder states) are values and s is the query (i.e. our first decoder hidden state)

Attention always requires computing an attention output a from the attention scores e:

$$\alpha = softmax(e)$$

$$a = \sum_{i=1}^{N} \alpha_i h_i$$

Several ways to compute e

Attention Variants

1. *Dot product attention*: $e_i = s^T h_i$
   - Need dimensions of s and h to be the same
2. *Multiplicative attention*: $e_i = s^T W h_i$
   - W is a weight matrix, dimensions reflect the dimensionality of s and h
3. *Additive attention*: $e_i = v^T tanh(W_1 h_i + W_2 s)$
   - W_1 and W_2 are weight matrices and v is a vector of weights

Wouldn't this be useful for vision?

- Yes, and it is a heavily studied area of current research
- This is how human vision works. When you see something, you don't scan the entire image equally like convolutional filters, you attend to particular parts of it. Compared to human vision, CNNs are hugely computationally wasteful
- Yet, this is a very nascent and quickly evolving field
- We can come back to it at the end of the course if there's time. Maybe next year they'll be a clearly revolutionary new architecture to teach (but CNNs are likely to remain important for awhile)