CS6101 Deep Learning for NLP

Recess Week

Machine Translation, Seq2Seq and Attention

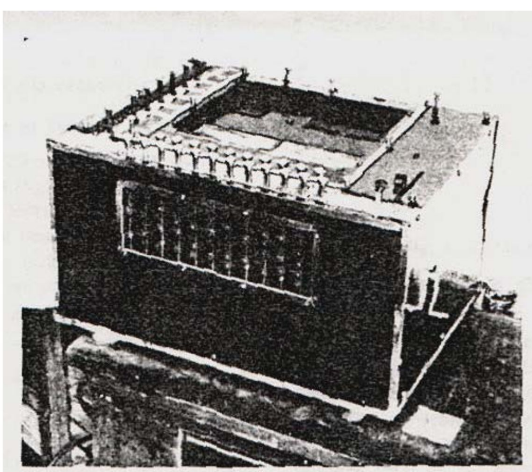
Presenters:
Yesha, Gary, Vikash, Nick, Hoang

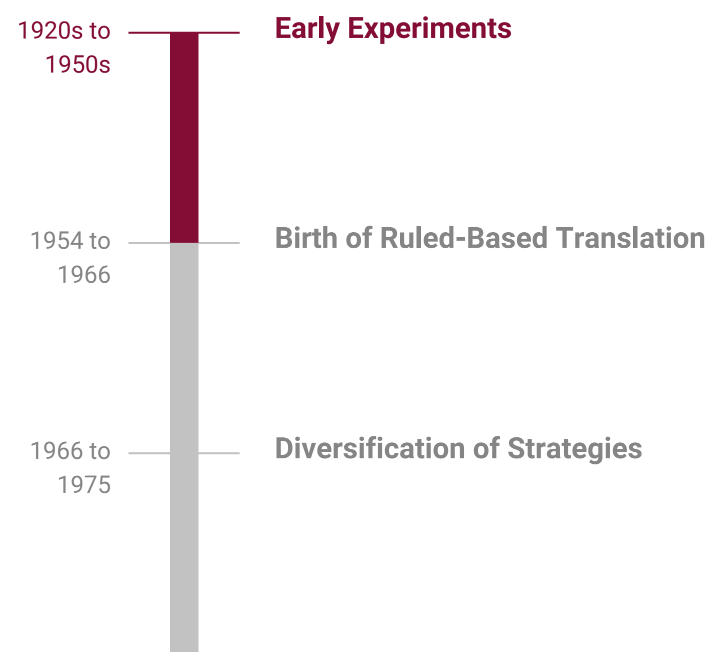
Outline

- Introduction - the evolution of machine translation systems (Yesha)
- Statistical machine translation (Gary)
- Neural machine translation and sequence-to-sequence architecture (Vikash)
- Attention (Nick)
- Other applications of seq2seq and attention (Hoang)

Evolution of Machine Translation

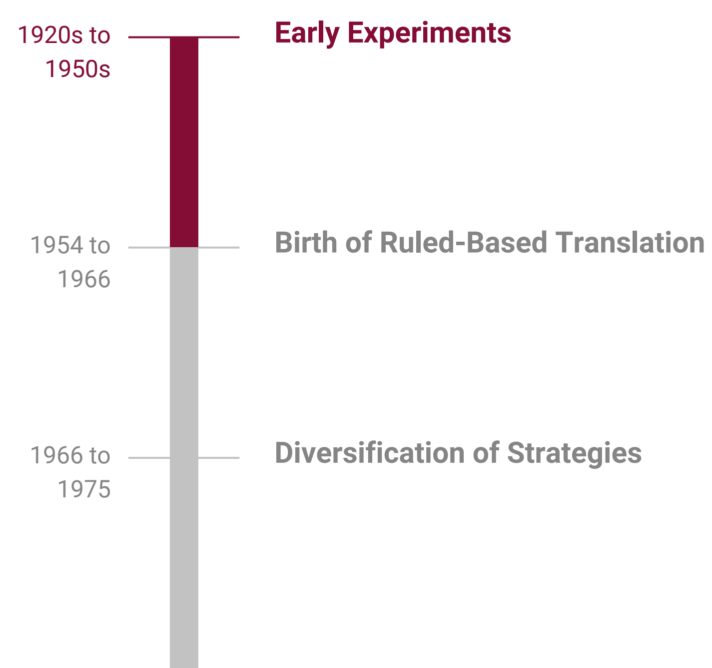
- The first known machine translation proposal was made in 1924 in Estonia and involved a typewriter-translator
- In 1933, French-Armenian inventor Georges Artsrouni received a patent for a mechanical machine translator that looked like a typewriter





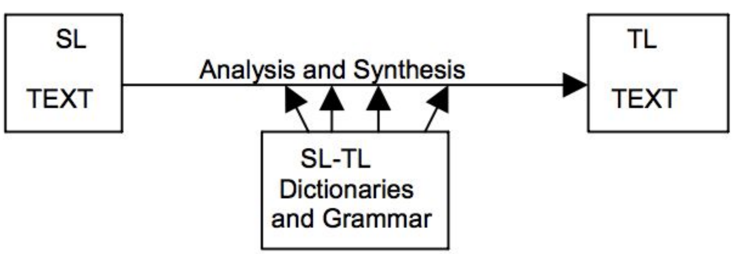
Evolution of Machine Translation

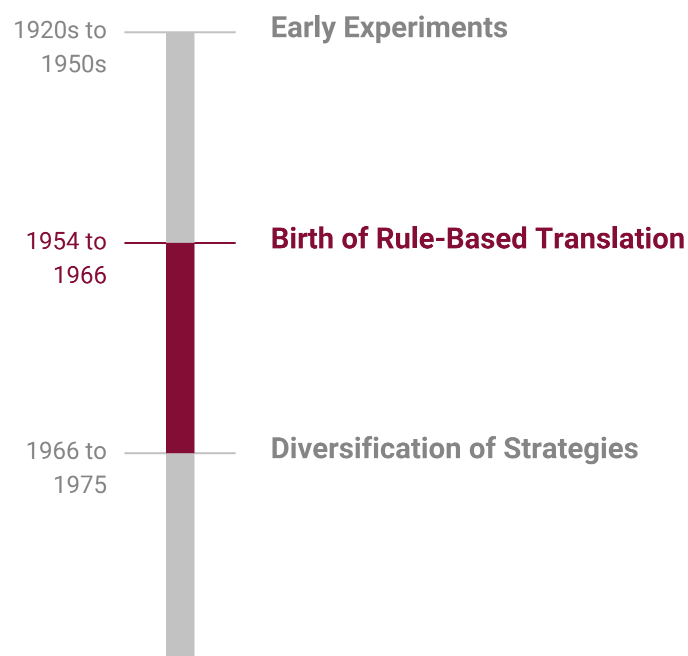
- From then to 1946, numerous proposals were made.
- Russian scholar Peter Petrovich Troyanskii got a patent for proposing a mechanised dictionary
- In 1946, Booth and Richens in Britain did a demo on automated dictionary
- In 1949, Warren Weaver wrote a memorandum that first mentioned the possibility of using digital computers to translate documents between natural human languages



Evolution of Machine Translation

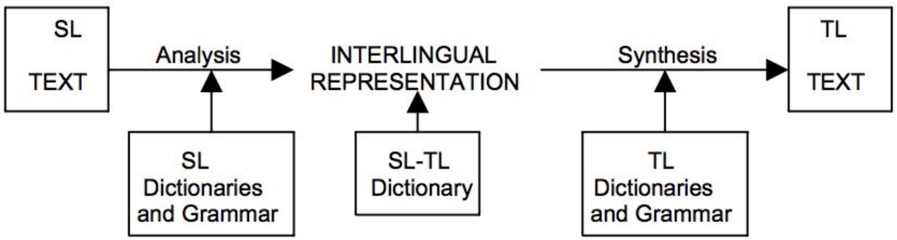
- In 1954, IBM and Georgetown University conduct a public demo
  - Direct translation systems for pairs of languages
  - Example: Russian-English systems to US Atomic Energy Commision
- Direct translation - limited scope in terms of languages and grammar rules used

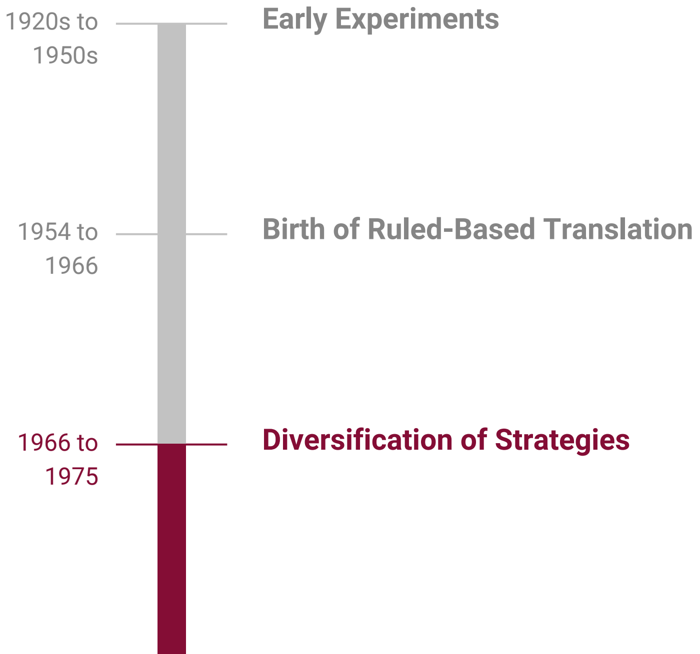




Evolution of Machine Translation

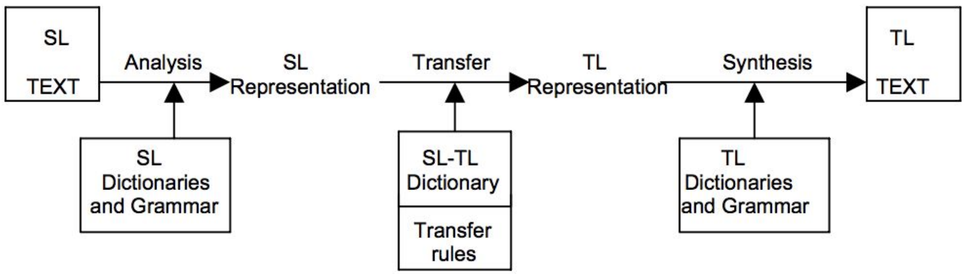
- Research continues - better understanding of linguistics and indirect approaches to system design
- Different types of rule-based translation systems:
  - Interlingual MT

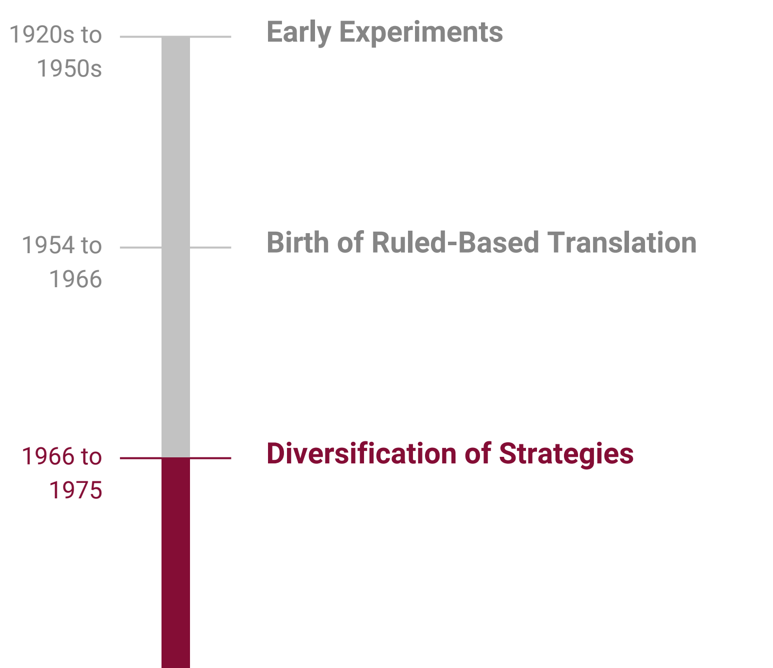




Evolution of Machine Translation

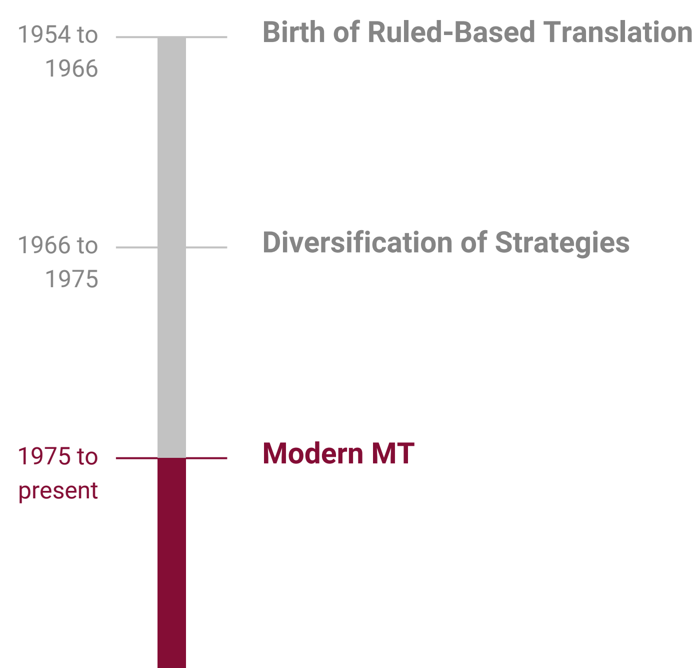
- Different types of rule-based translation systems:
    - Transfer MT





Evolution of Machine Translation

- Statistical Machine Translation
- Example Based Machine Translation
- Neural Machine Translation



Statistical Machine Translation

Basic Approach

- Task: Given *source* text F (foreign language)
  find *target* text E (English)

Basic Approach

- Task: Given *source* text F (foreign language)
  find *target* text E (English)
- Probabilistic formulation (via Bayes Rule)

Bayes rule:

$$P(E|F) = \frac{P(E|F)P(E)}{P(F)}$$

$$\hat{E} = \operatorname*{argmax}_{E} P(E|F) = \operatorname*{argmax}_{E} \underbrace{P(F|E)}_{\text{Translation model}} \underbrace{P(E)}_{\text{Language model}}$$

Basic Approach

- Task: Given *source* text F (foreign language)
  find *target* text E (English)
- Probabilistic formulation (via Bayes Rule)

Bayes rule:

$$P(E|F) = \frac{P(E|F)P(E)}{P(F)}$$

$$\hat{E} = \underset{E}{\operatorname{argmax}} P(E|F) = \underset{E}{\operatorname{argmax}} \underbrace{P(F|E)}_{\text{Translation model}} \underbrace{P(E)}_{\text{Language model}}$$

- Translation model, $P(F|E)$ models the correctness of the translation
- Language model, $P(E)$ models the fluency of the target language
- Two components are trained independently, with different datasets

General Formulation

- Log-linear model

$$\hat{E} = \operatorname*{argmax}_E P(E|F) = \operatorname*{argmax}_E \sum_i^m \lambda_i f_i(E,F)$$

General Formulation

- Log-linear model

$$\hat{E} = \underset{E}{\operatorname{argmax}}\, P(E|F) = \underset{E}{\operatorname{argmax}} \textstyle\sum_i^m \lambda_i f_i(E,F)$$

  - Allows introduction of additional components (m)
    e.g. length of sentence (penalty term)
    additional language models
    external lexicon co-occurrence
    syntactic dependencies, e.g. grammar
  - Assigns weights to different components

General Formulation

- Log-linear model

$$\hat{E} = \underset{E}{\operatorname{argmax}}\, P(E|F) = \underset{E}{\operatorname{argmax}} \sum_i^m \lambda_i f_i(E, F)$$

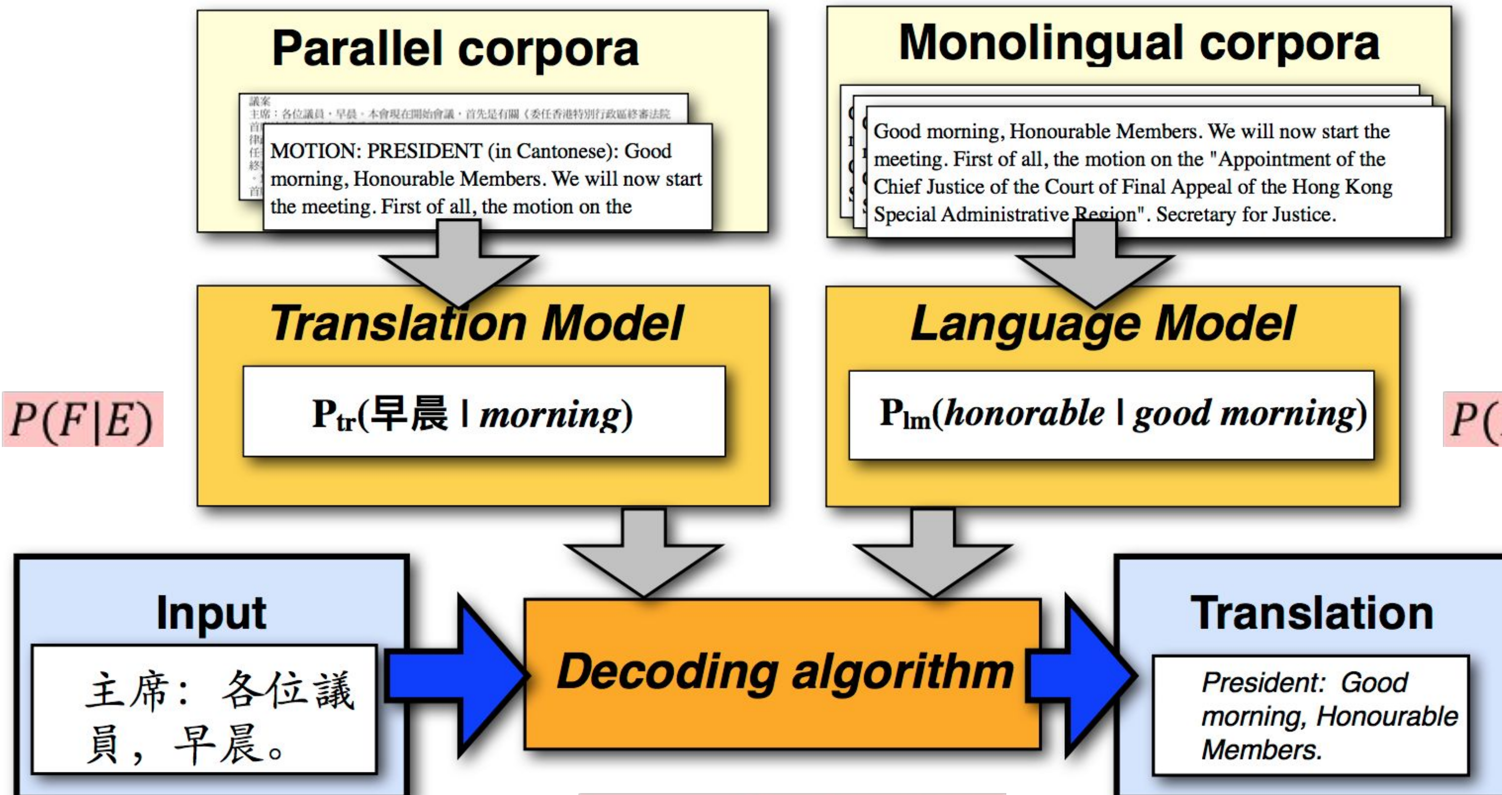
  - Allows introduction of additional components (m)

    e.g. length of sentence (penalty term)
    additional language models
    external lexicon co-occurrence
    syntactic dependencies, e.g. grammar
  - Assigns weights to different components
- If $f_1 = \log(P(F|E),)\ f_2 = \log(P(E)), \lambda_1 = \lambda_2$, then it falls back to earlier simple case
- Can get overly-complex (we shall not go further)

E.g. On voit Jon à la télévision (French)

	good match to French? P(F|E)	good English? P(E)
Jon appeared in TV.	✓	
It back twelve saw.		
In Jon appeared TV.	✓	
Jon is happy today.		✓
Jon appeared on TV.	✓	✓
TV appeared on Jon.	✓	
Jon was not happy.		✓

Overview of SMT (from http://courses.engr.illinois.edu/cs447; lecture 22 slides)



$P(F|E)$

$P(E)$

$$\operatorname{argmax}_E P(F|E)P(E)$$

Recap: Language Model, $P(E)$

- Largely covered in Week 5
- N-grams models

via chain rule

$$P(E) = P(e_1, e_2, \dots, e_N) = P(e_1)P(e_2|e_1)P(e_3|e_2e_1) \dots P(e_N|e_{N-1} \dots e_1)$$

$$P(e_i|e_{i-1} \dots e_1) = P(e_i|e_{i-1} \dots e_{i-n}) = \frac{P(e_i, e_{i-1}, \dots, e_{i-n})}{P(e_{i-1}, \dots e_{i-n})}$$

Markov assumption

$$\approx \frac{Counts(e_i, e_{i-1}, \dots, e_{i-n})}{Counts(e_{i-1}, \dots e_{i-n})}$$

Maximum Likelihood Estimation

  - estimate the likelihood of words/phrases based on their frequencies found in large corpuses
  - Extensions: Smoothing + Backoffs

Translation Model, $P(F|E)$

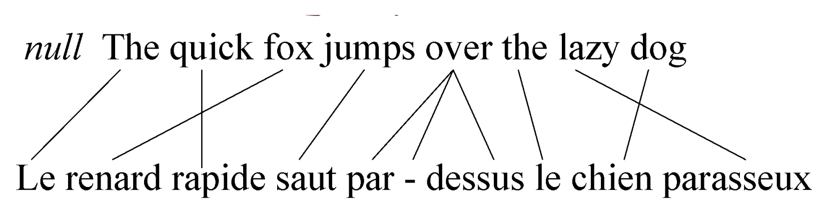
- Learns the lexical mapping between languages + ordering
- Typically focus on word-level *or* phrase-level mapping
  (insufficient data to learn entire sentences mapping directly)

Translation Model, $P(F|E)$

- Learns the lexical mapping between languages + ordering
- Typically focus on word-level *or* phrase-level mapping
  (insufficient data to learn entire sentences mapping directly)

English	**Norwegian**	**Probability**
heavy	tung	0.95
heavy metal	heavy metal	0.61
heavy metal	tungmetal	0.34
smoker	røyker	0.99
heavy smoker	storrøyker	0.99
...	...	...

Translation probability table



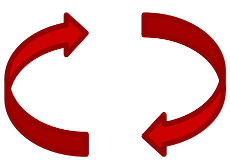
Lexical mapping between languages

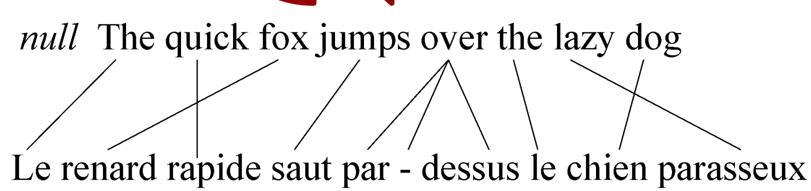
Translation Model, $P(F|E)$

- Learns the lexical mapping between languages + ordering
- Typically focus on word-level *or* phrase-level mapping
  (insufficient data to learn entire sentences mapping directly)
- Chicken-and-egg problem:
  - Given a translation table, we can easily extract the mappings
  - Given the mappings, we can easily estimate the probabilities

English	Norwegian	Probability
heavy	tung	0.95
heavy metal	heavy metal	0.61
heavy metal	tungmetal	0.34
smoker	røyker	0.99
heavy smoker	storrøyker	0.99
...	...	...

Translation probability table





Lexical mapping between languages

Word Alignments

- Assume that every sentence is aligned
- Break it down further: $P(F|E) = \sum_A P(E,A|F)$

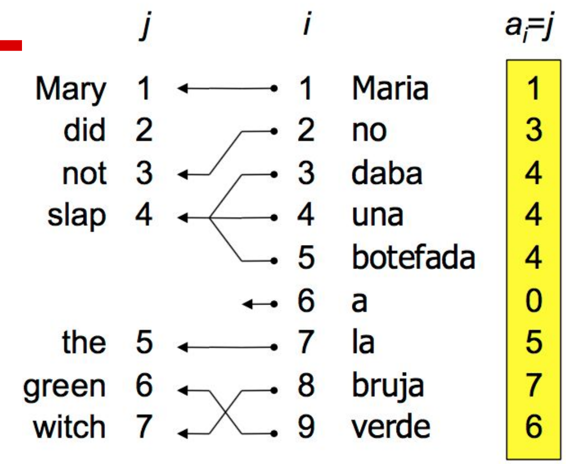
  where A is the alignment

Word Alignments

j=0 → means *NULL*
i=0

- Assume that every sentence is aligned
- Break it down further: $P(F|E) = \sum_A P(E,A|F)$

  where A is the alignment



Word Alignments

- Assume that every sentence is aligned
- Break it down further: $P(F|E) = \sum_A P(E,A|F)$

  where A is the alignment

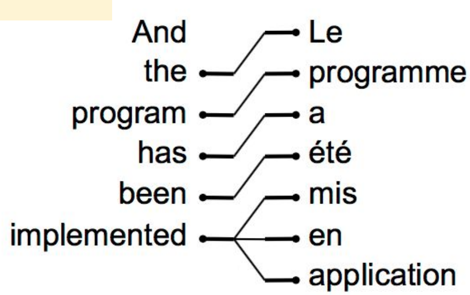
j=0 → means *NULL*
i=0

	j	i		a_i=j
Mary	1	1	Maria	1
did	2	2	no	3
not	3	3	daba	4
slap	4	4	una	4
		5	botefada	4
		6	a	0
the	5	7	la	5
green	6	8	bruja	7
witch	7	9	verde	6

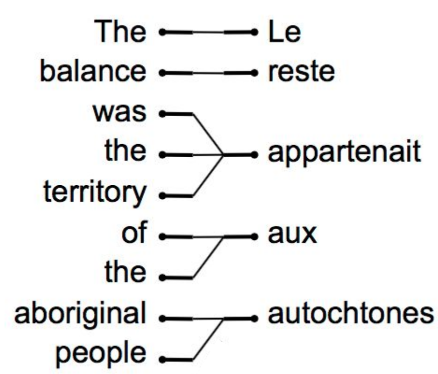
1-1 mapping

Spurious words

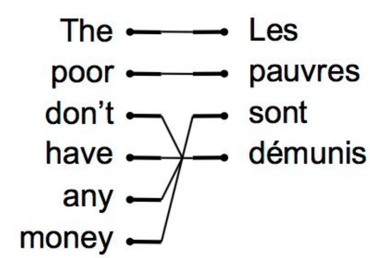
1-many mapping



many-1 mapping

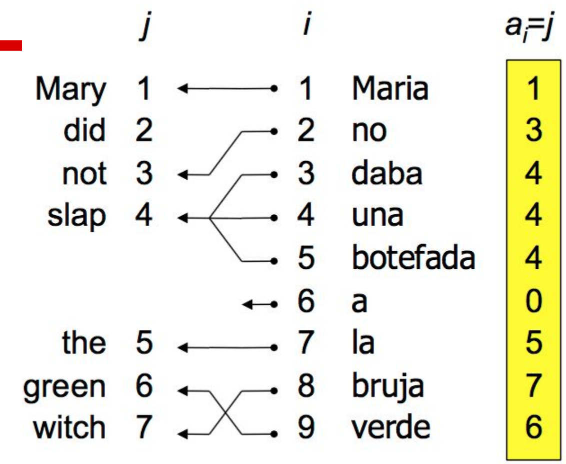


many-many mapping



Word Alignments

j=0 → means *NULL*
i=0



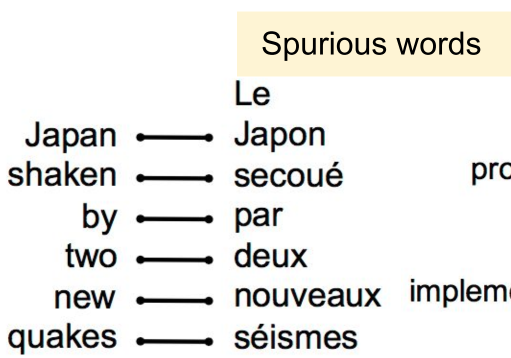
- Assume that every sentence is aligned
- Break it down further: $P(F|E) = \sum_A P(E, A|F)$

  where A is the alignment
- Qn: How many possible alignments are there?

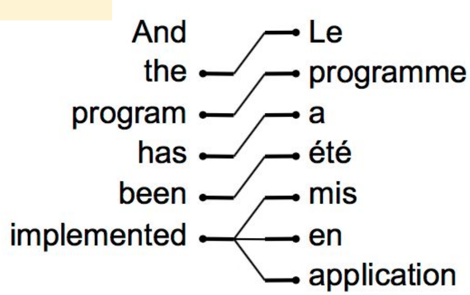
  where l is the # of words in E sentence (target)

  m is the # of words in F sentence (source)

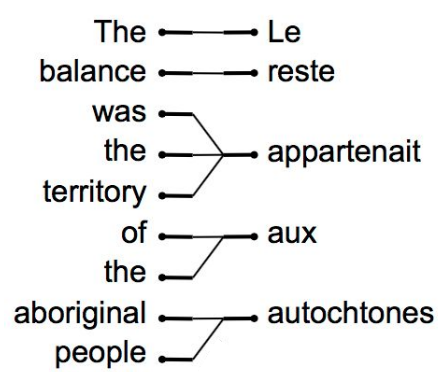
1-1 mapping



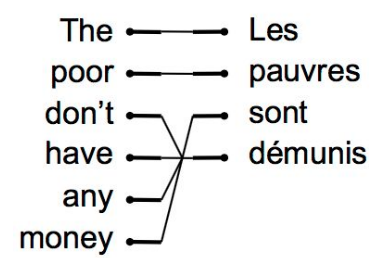
1-many mapping



many-1 mapping



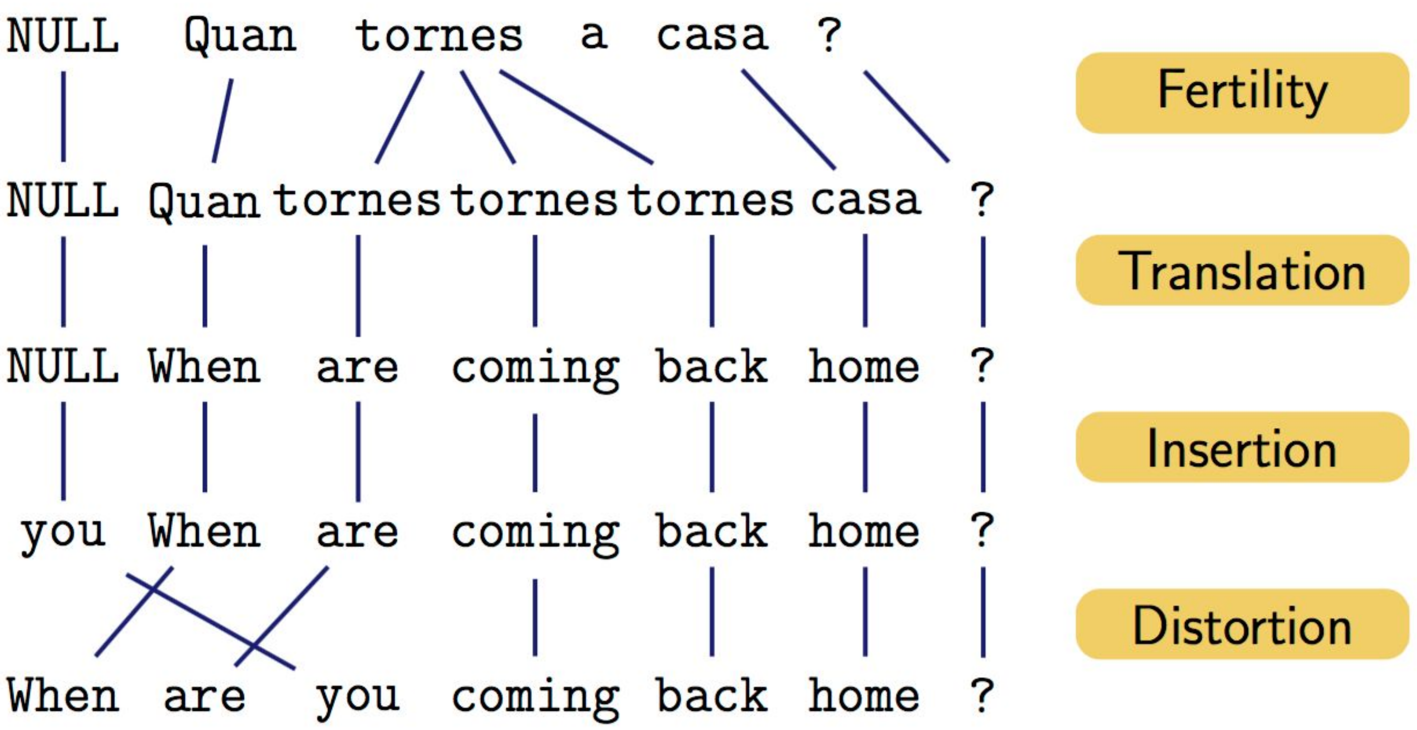
many-many mapping



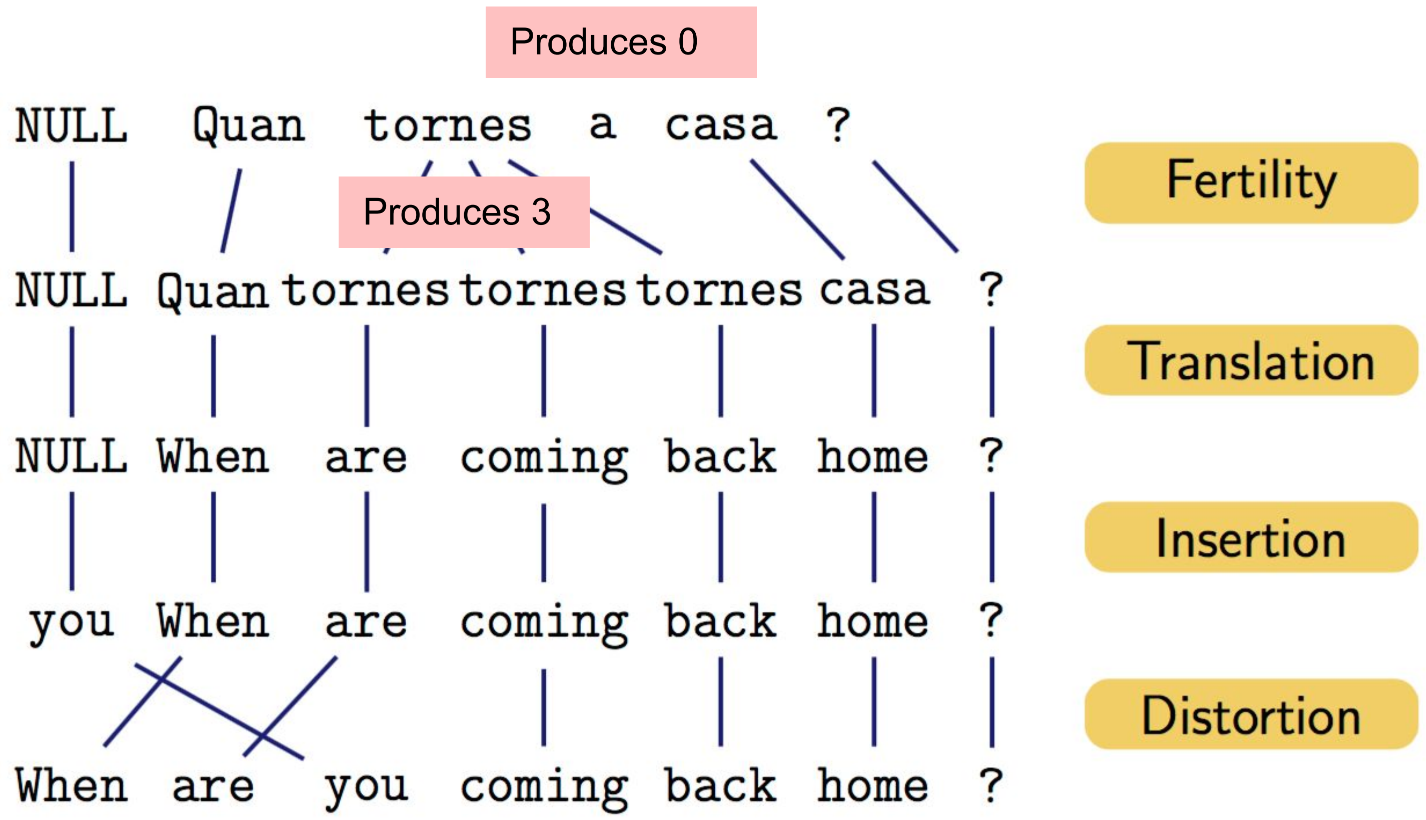
IBM Translation Models

- Characterise $P(E|F)$ with 4 parameters:
  1. **Lexical translation**, t
     e.g. t$(x|y)$: probability that x translates to y
  2. **Fertility**, n
     e.g. $n(3|x)$: probability that x produces 3 words
  3. **Distortion**, d
     e.g. $d(j|i,m,n)$: prob. that the j^{th} word generates the i^{th} word. m and n are the length of the source and target sentences
  4. **Insertion**, p_1
     e.g. $p_1(x)$: probability that x is generated from $NULL$

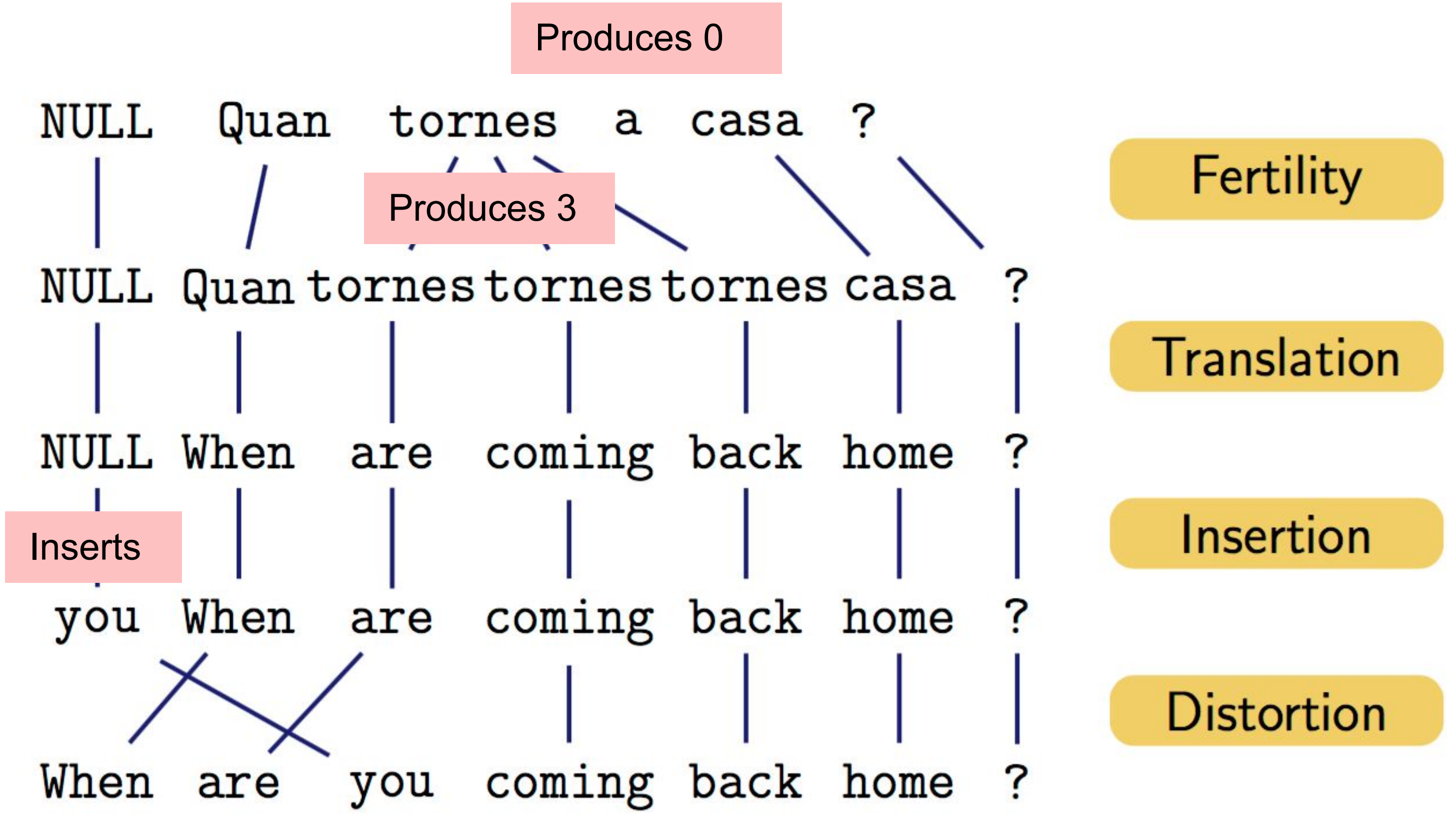
IBM Translation Model Example



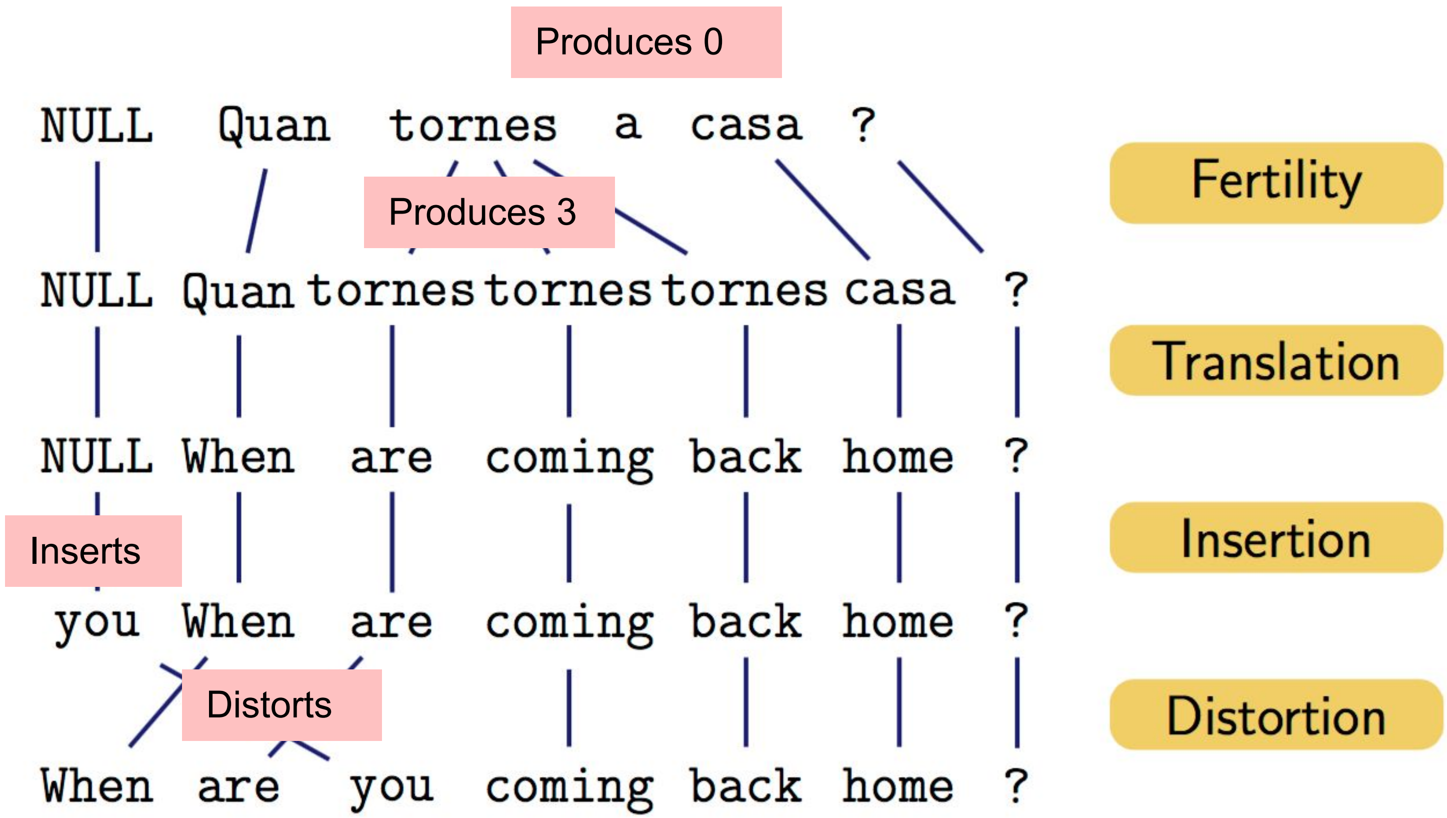
IBM Translation Model Example



IBM Translation Model Example



IBM Translation Model Example



IBM SMT Translation Models

- Models are typically built progressively, starting from the simplest to the most complex
  - IBM1 – lexical transitions only
  - IBM2 – lexicon plus absolute position
  - HMM – lexicon plus relative position
  - IBM3 – plus fertilities
  - IBM4 – inverted relative position alignment
  - IBM5 – non-deficient version of model 4

IBM1 Translation Model

$$P(F, A|E) = \frac{\epsilon}{(1+l)^m} \prod_{i=1}^{m} t(f_i \,|e_{A_i})$$

where l is the number of words in E sentence (target)

m is the number of words in F sentence (source)

ϵ is the normalizing constant

IBM1 Translation Model

$$P(F,A|E) = \frac{\epsilon}{(1+l)^m} \prod_{i=1}^{m} t(f_i \,|e_{A_i})$$

where l is the number of words in E sentence (target)

m is the number of words in F sentence (source)

ϵ is the normalizing constant

e.g. *null* The quick fox jumps over the lazy dog $l = 8$

Le renard rapide saut par - dessus le chien parasseux $m = 10$

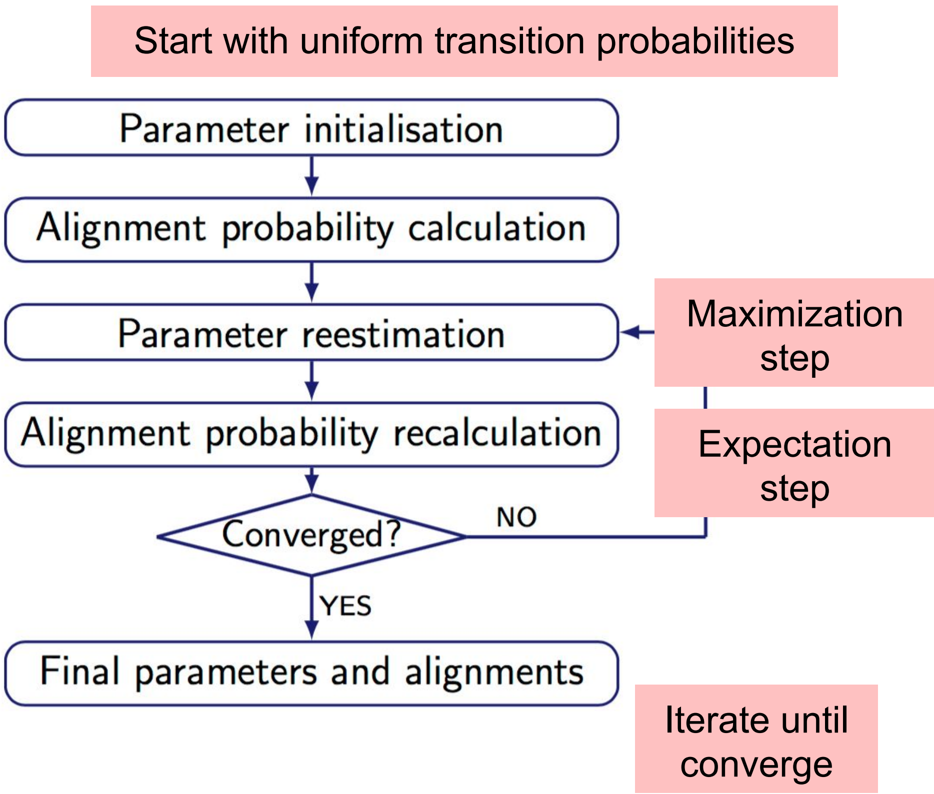
$$P(F,A|E) = \frac{\epsilon}{(1+8)^{10}} [t(le|the) \cdot$$
$$t(reneard|fox) \cdot$$
$$\dots$$
$$t(parasseux|lazy)]$$

Training Translation Models

- How can t, n, d and p_1 be estimated?

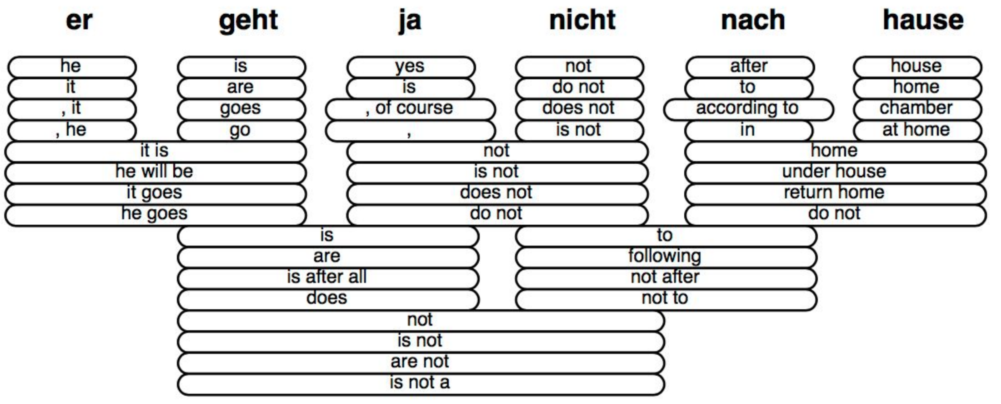
Training Translation Models

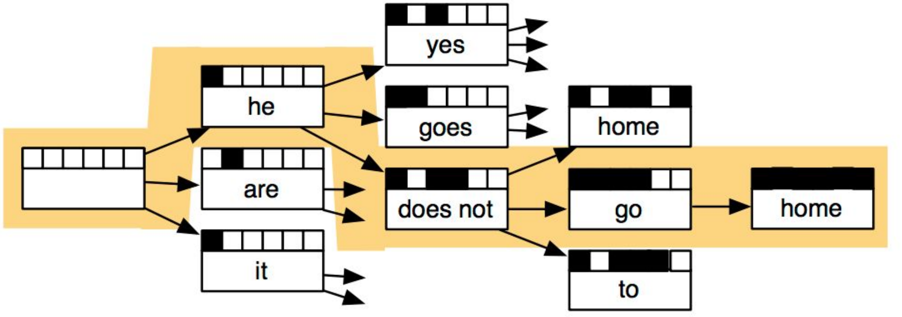
- How can t, n, d and p_1 be estimated?
- Use Expectation-Maximization (EM) method to iteratively infer the parameters



Decoder – Search problem

- Finds a list of possible target translation candidates, with the highest probabilities
- Exhaustive search is infeasible
- Typically, search heuristics are used instead
  e.g. beam search
  greedy decoding





Other things not discussed

- Phrase-based alignment
- Sentence alignment
- Syntactic-based or grammar-based models
- Hierarchical phrase-based translation
- …

Shortcomings of SMT

- Models can grow to be overly complex
- Requires a lot of hand-designed feature engineering in order to be effective
- Difficult to manage out-of-vocabulary words
- Components are independently trained; no synergy between models vs. end-to-end training
- Memory-intensive, as decoder relies on huge lookup tables → not mobile-tech friendly

➔ Let's look into how Neural Machine Translation (NMT) models solves the above issues

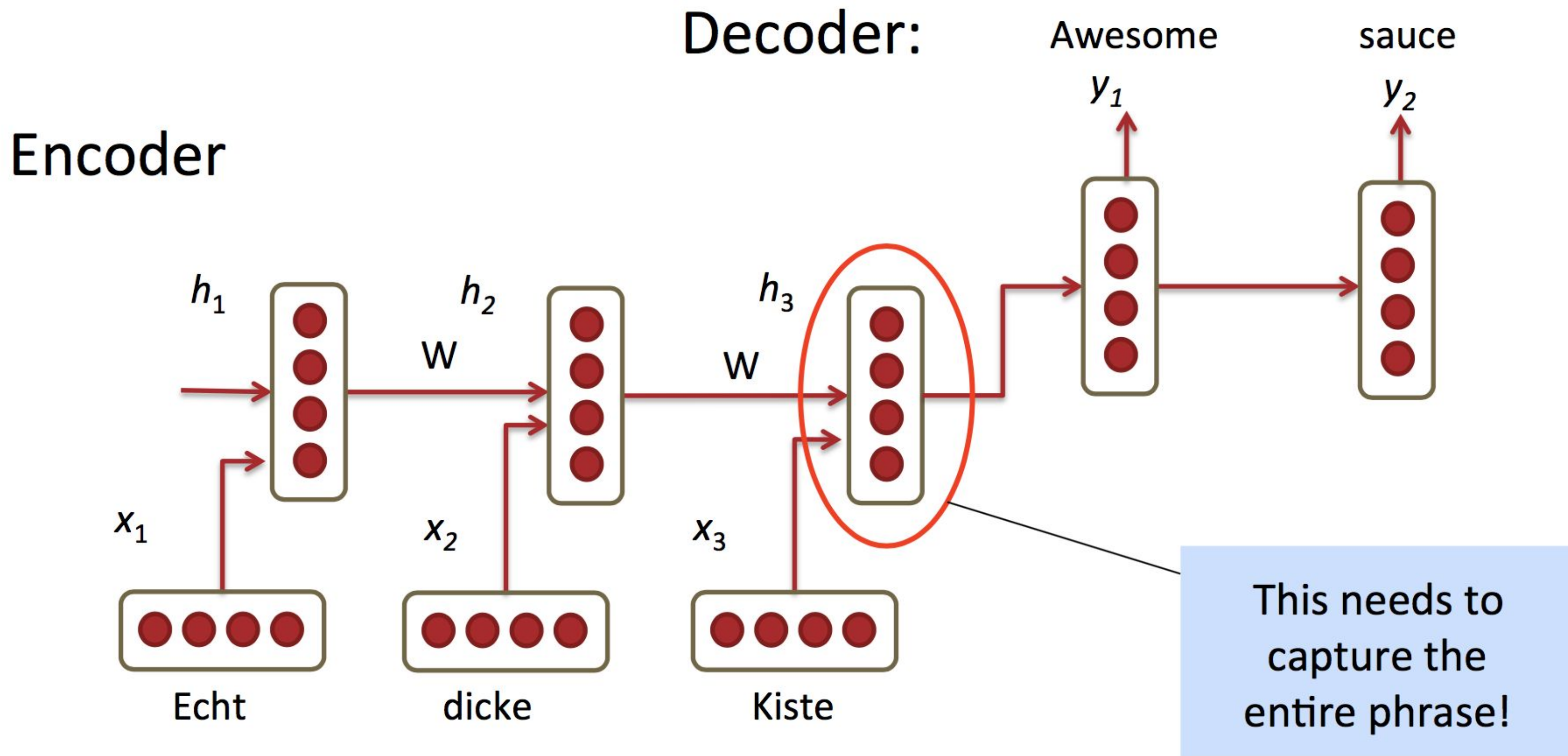
Neural machine translation and sequence-to-sequence architecture

What is Neural Machine Translation?

- Neural Machine Translation (NMT) is a way to do Machine Translation with a single neural network.

- The neural network architecture is called sequence-to-sequence (aka seq2seq) and it involves two RNNs.

The sequence-to-sequence model



Simplest Model

Encoder: $h_t = \phi(h_{t-1}, x_t) = f\left(W^{(hh)} h_{t-1} + W^{(hx)} x_t\right)$

Decoder: $h_t = \phi(h_{t-1}) = f\left(W^{(hh)} h_{t-1}\right)$

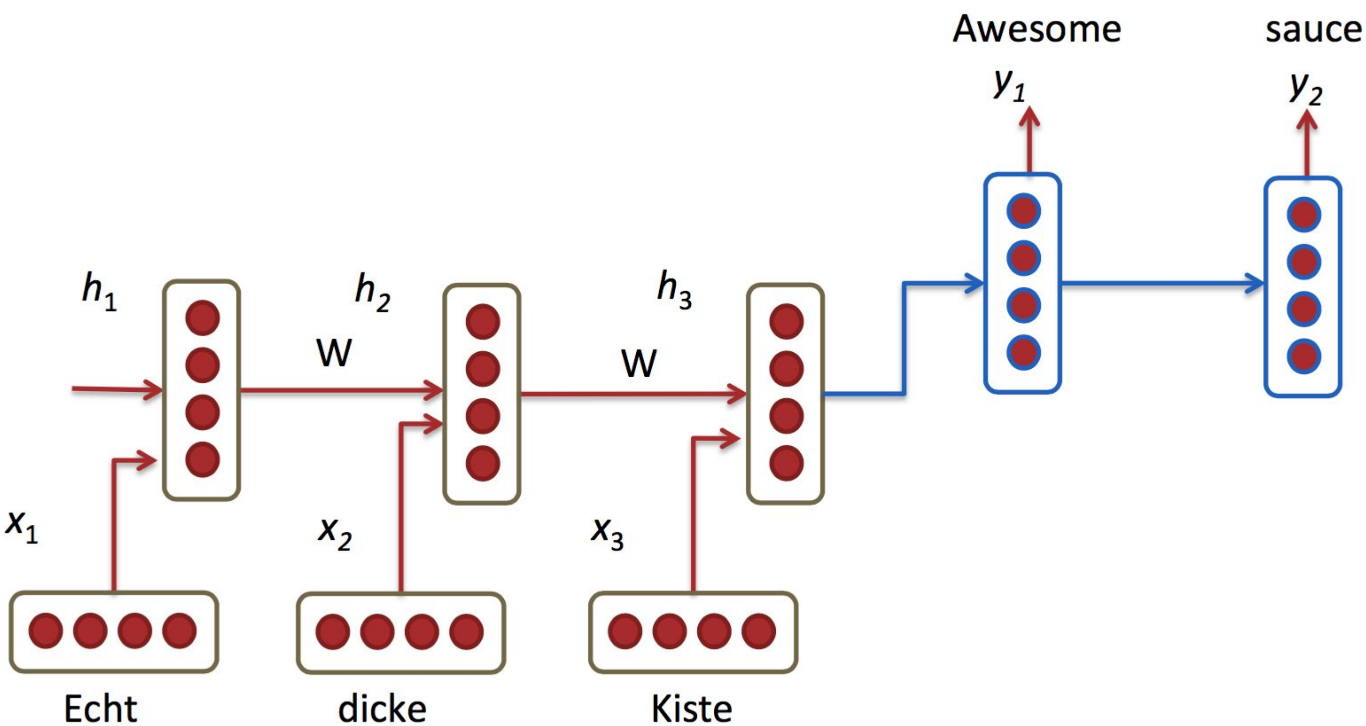
$$y_t = softmax\left(W^{(S)} h_t\right)$$

Minimize cross entropy error for all target words conditioned on source words

$$\max_\theta \frac{1}{N} \sum_{n=1}^{N} \log p_\theta(y^{(n)} | x^{(n)})$$

Model Extensions

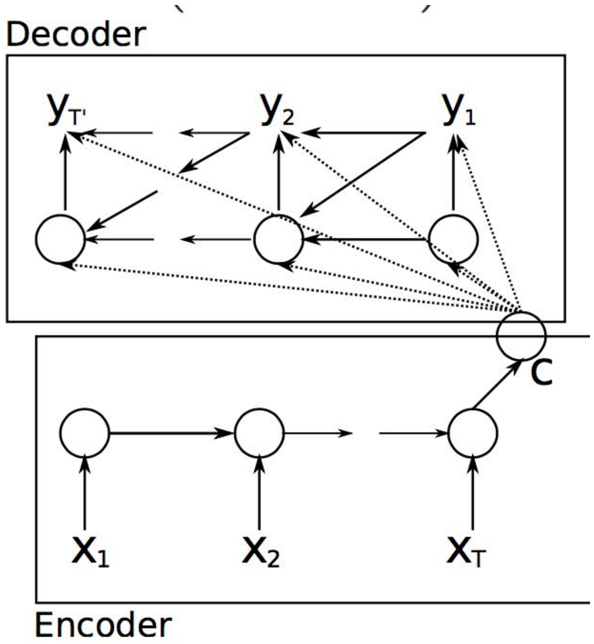
1. Train different RNN weights for encoding and decoding



Model Extensions

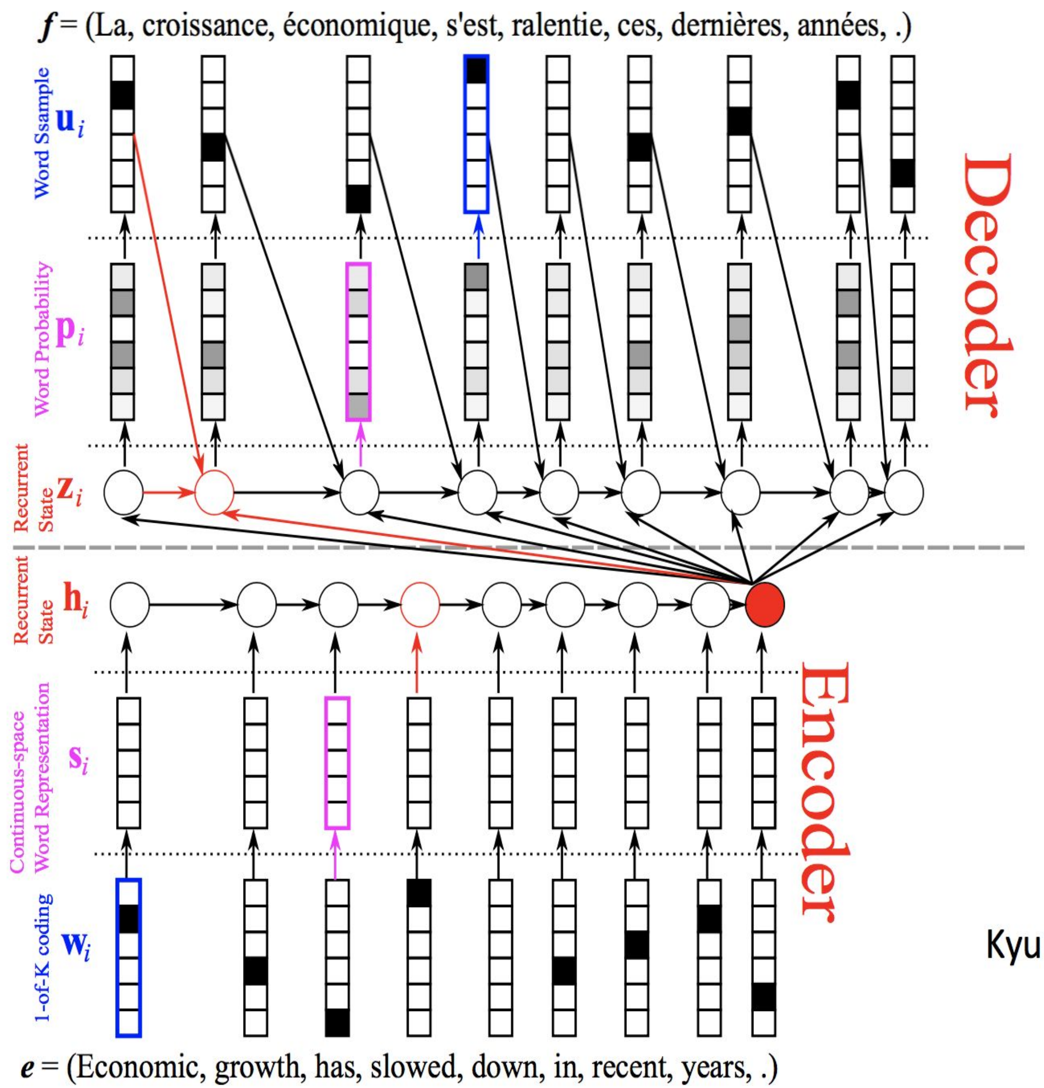
2. Compute every hidden state in decoder from
    - Previous hidden state (standard)
    - Last hidden vector of encoder $c=h_T$
    - Previous predicted output word y_{t-1}

$$h_{D,t} = \phi_D(h_{t-1}, c, y_{t-1})$$



Cho et al. 2014

Model Extensions

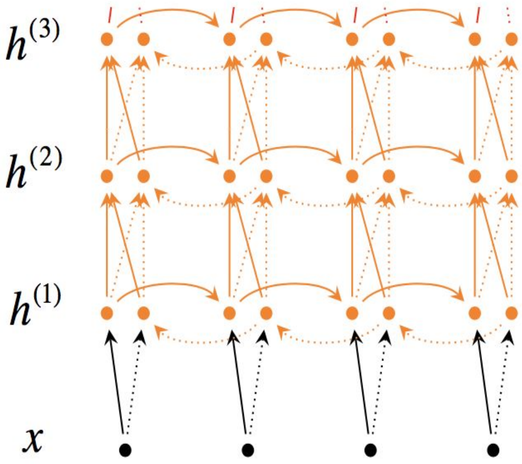


Kyunghyun Cho et al. 2014

Model Extensions

3. Train stacked/deep RNNs with multiple layers

4. Potentially train bidirectional encoder



5. Train input sequence in reverse order for simple optimization problem: Instead of A B C → X Y, train with C B A → X Y

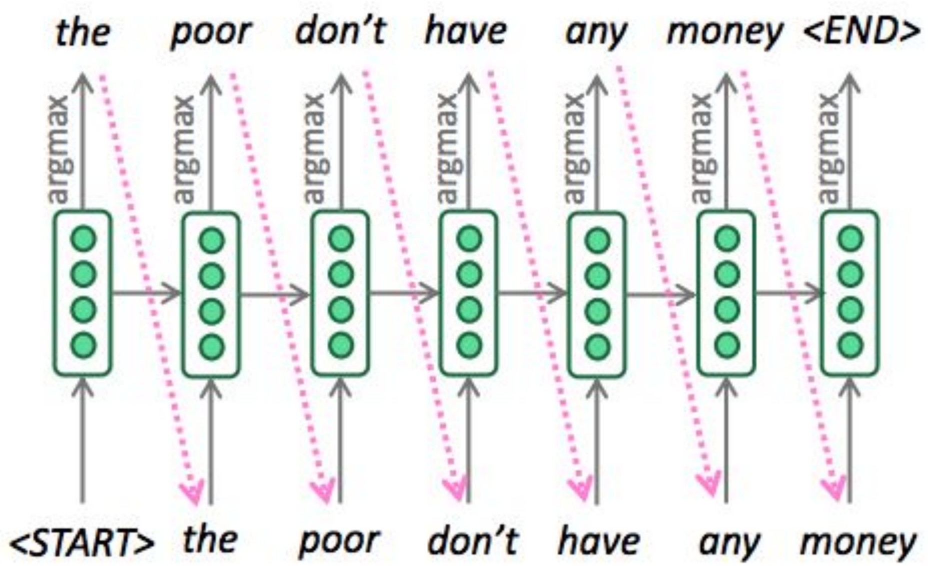
Model Extensions

6.Use more complex hidden units GRU or LSTM.

- Main ideas:
  - keep around memories to capture long distance dependencies
  - allow error messages to flow at different strengths depending on the inputs

Greedy Decoding

- Target sentence is generated by taking argmax on each step of the decoder.



Greedy Decoding Problems

- 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

Beam Search Decoding

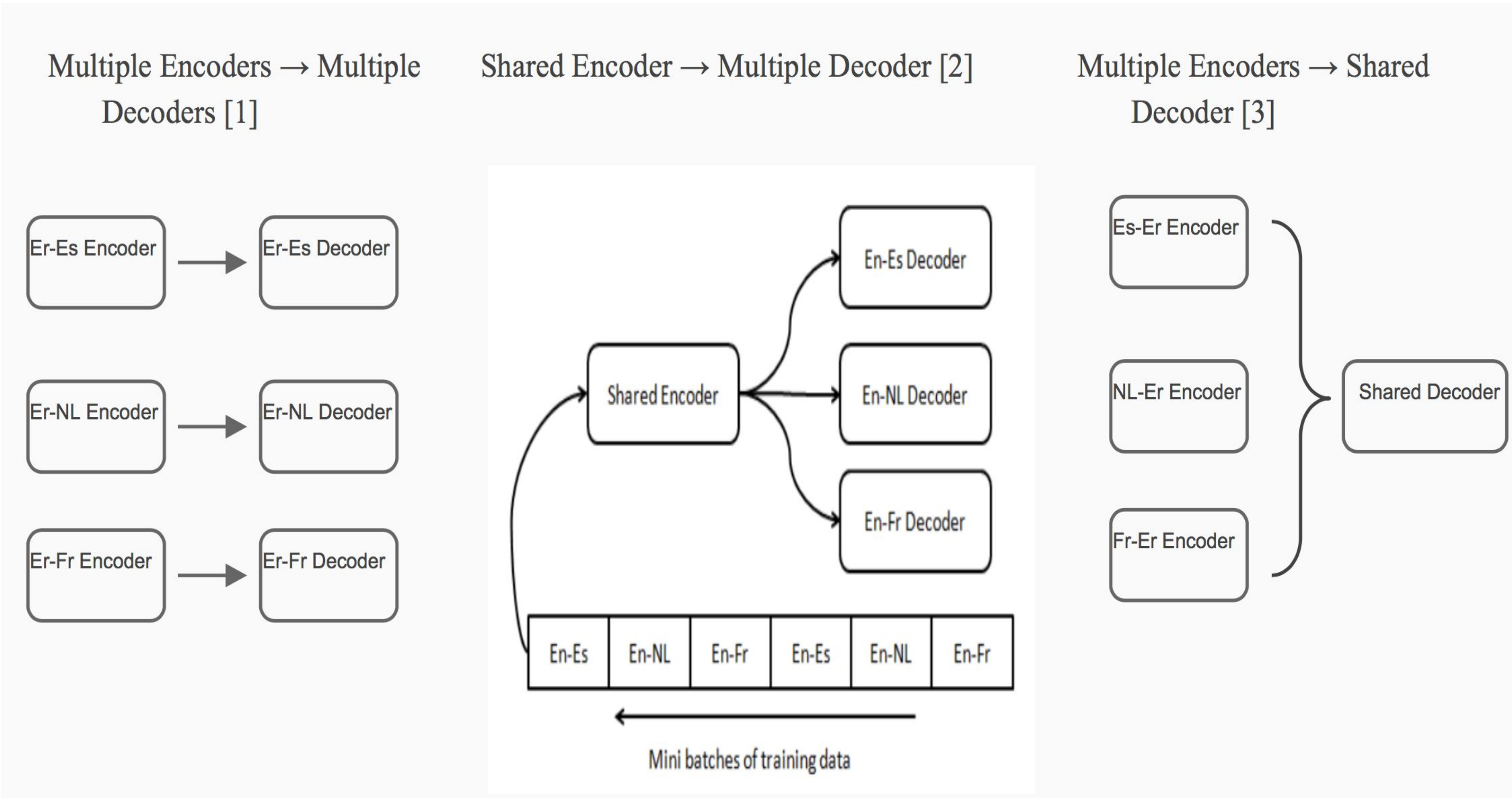
- Ideally we want to find *y* that maximizes

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

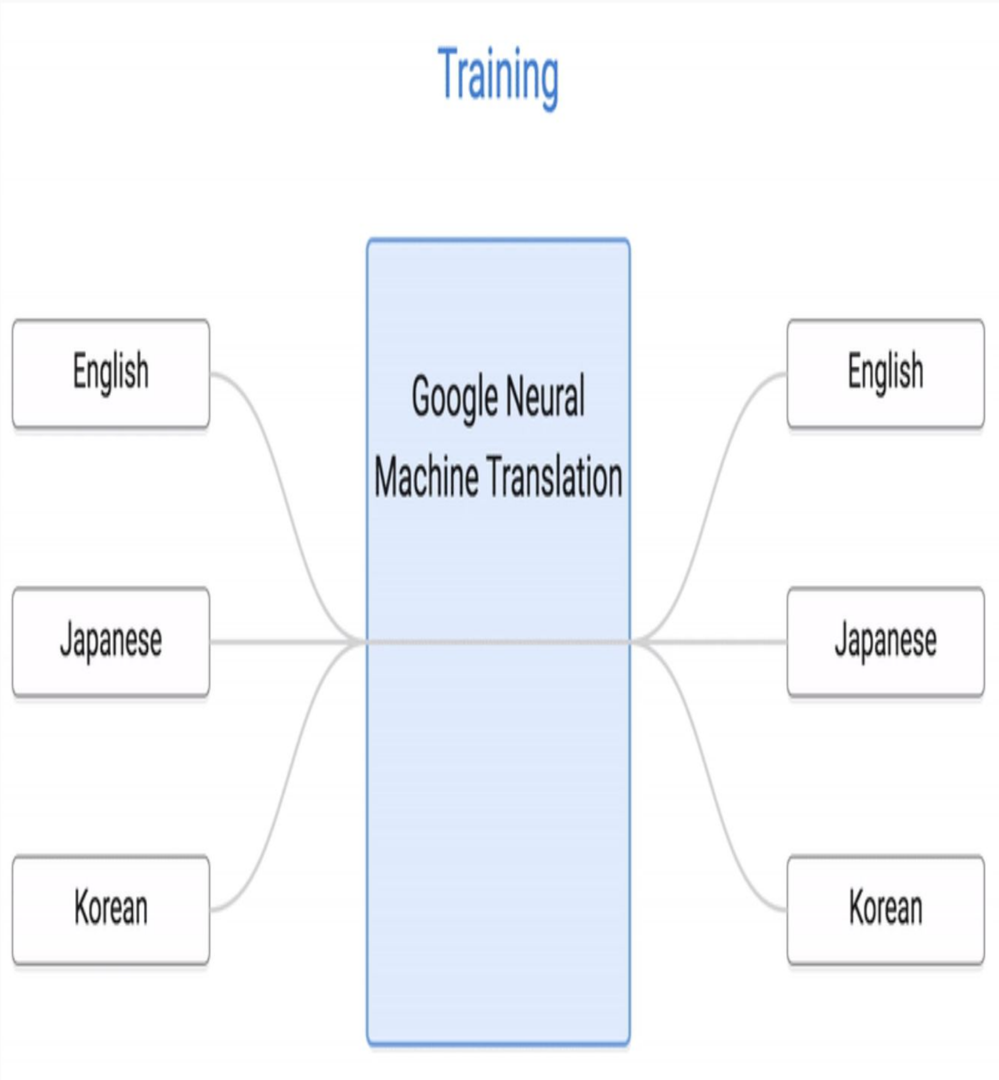
- We could try enumerating all *y* → too expensive!
  - Complexity $O(V^T)$ where *V* is vocab size and *T* is target sequence length

- **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!

Multilingual NMT

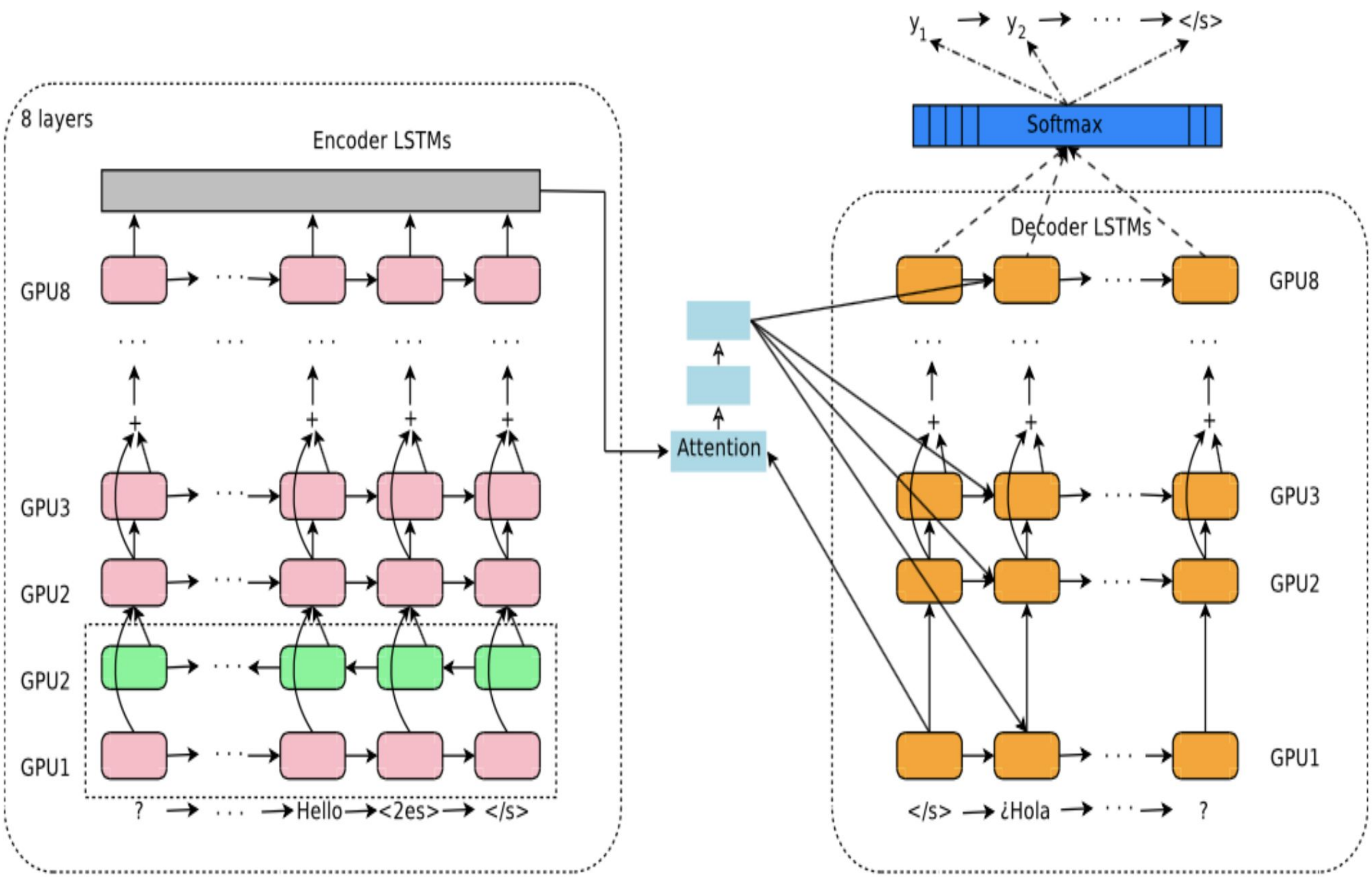


Google’s Multilingual NMT



- **Simplicity**: single model
- **Low-resource language improvements**
- **Zero-shot translation**

Google’s Multilingual NMT Architecture



Four big wins of NMT

1. **End-to-end training**
   All parameters are simultaneously optimized to minimize a loss function on the network's output

2. **Distributed representations share strength**
   Better exploitation of word and phrase similarities

3. **Better exploitation of context**
   NMT can use a much bigger context – both source and partial target text – to translate more accurately

4. **More fluent text generation**
   Deep learning text generation is much higher quality

So is Machine Translation solved?

- **Nope!**
- Many difficulties remain:
  - Out-of-vocabulary words
  - Domain mismatch between train and test data
  - Maintaining context over longer text
  - Low-resource language pairs

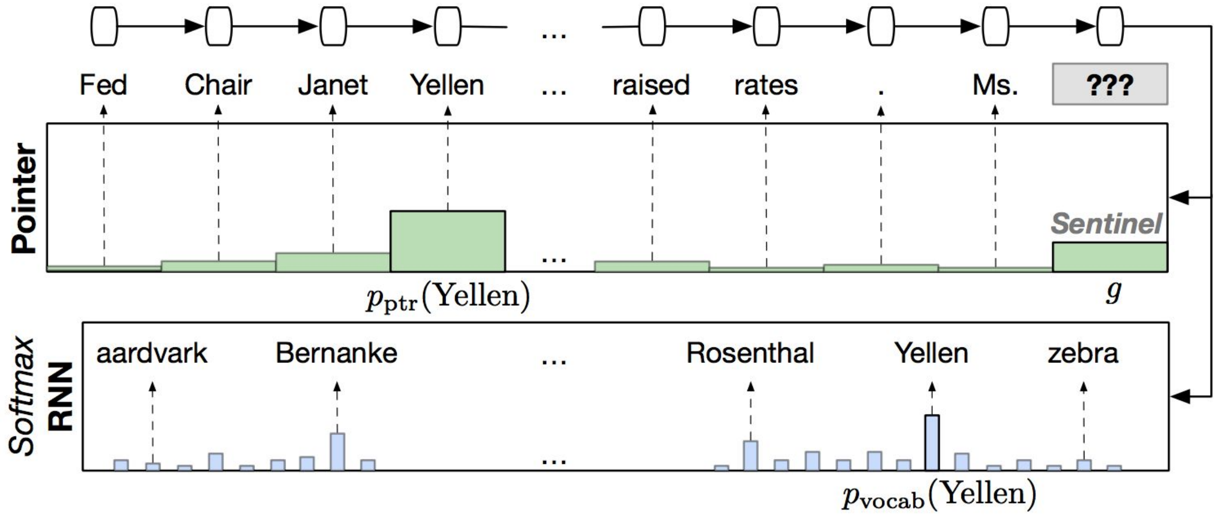
Zero Shot Word Prediction

- Answers can only be predicted if they were seen during training and part of the softmax

- But it’s natural to learn new words in an active conversation and systems should be able to pick them up

Predicting Unseen Words

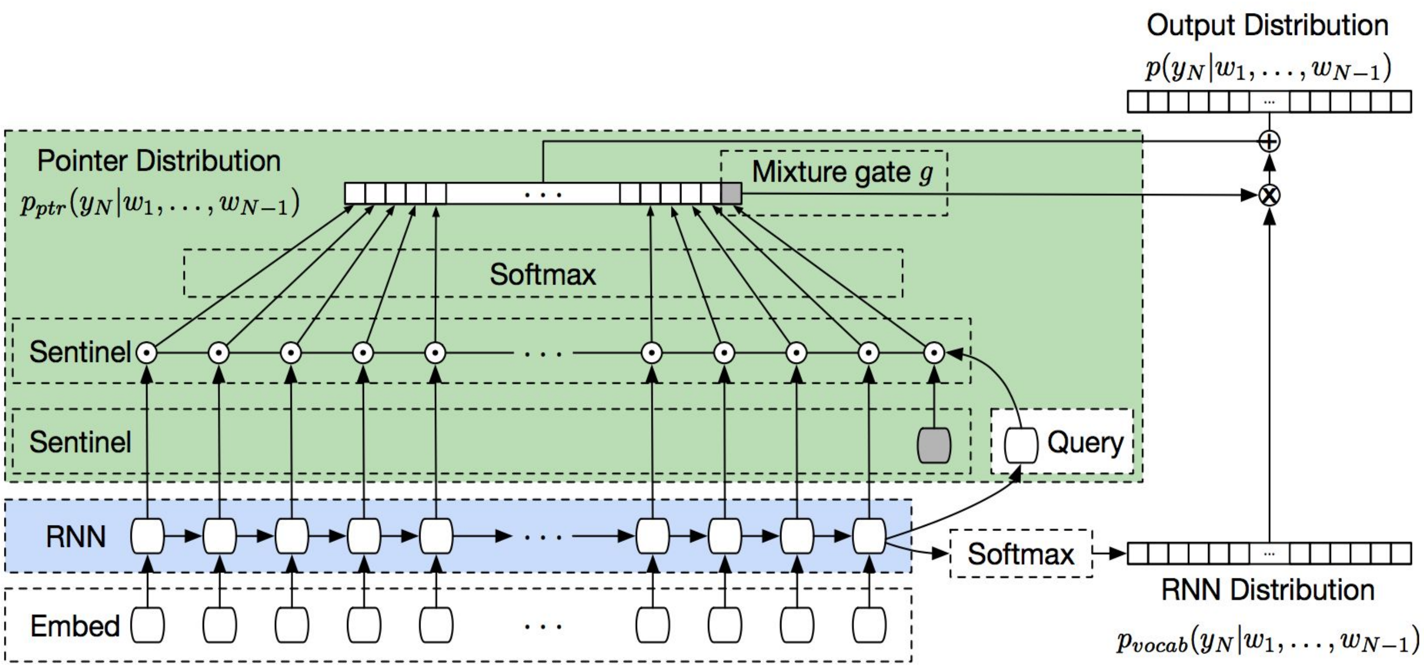
- Idea: Mixture Model of softmax and pointers:



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

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

Pointer Details



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

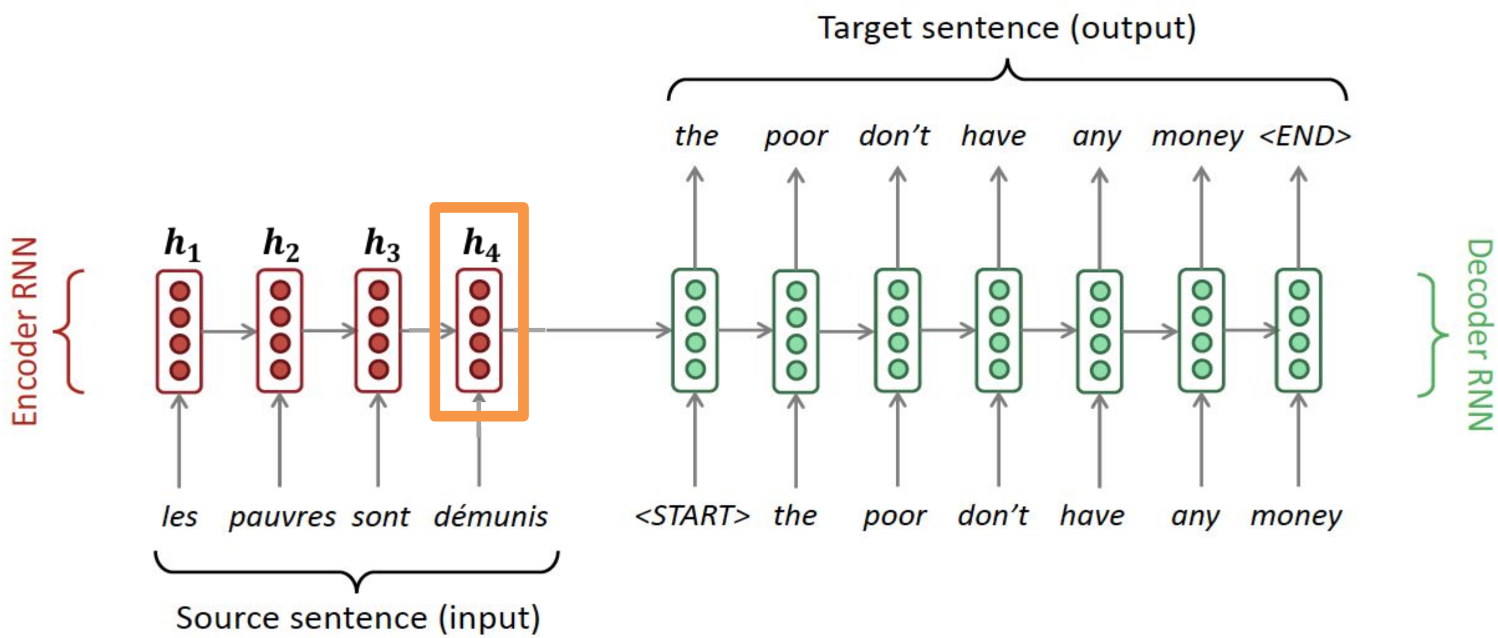
$$z_i = q^T h_i, \qquad p_{\text{ptr}}(w) = \sum_{i \in I(w,x)} a_i,$$

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

Attention please!

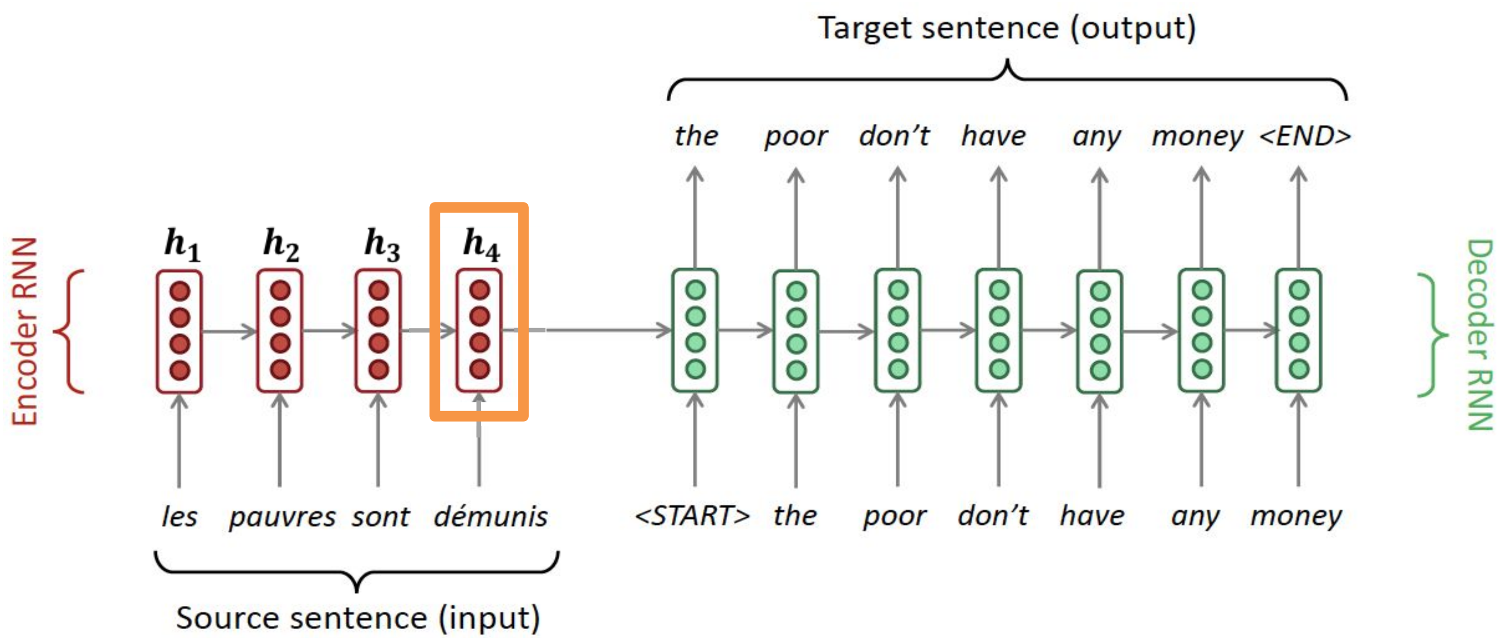
Seq2Seq: the bottleneck problem

- Encoding of source sentence: a fixed length vector $\boldsymbol{h_4}$



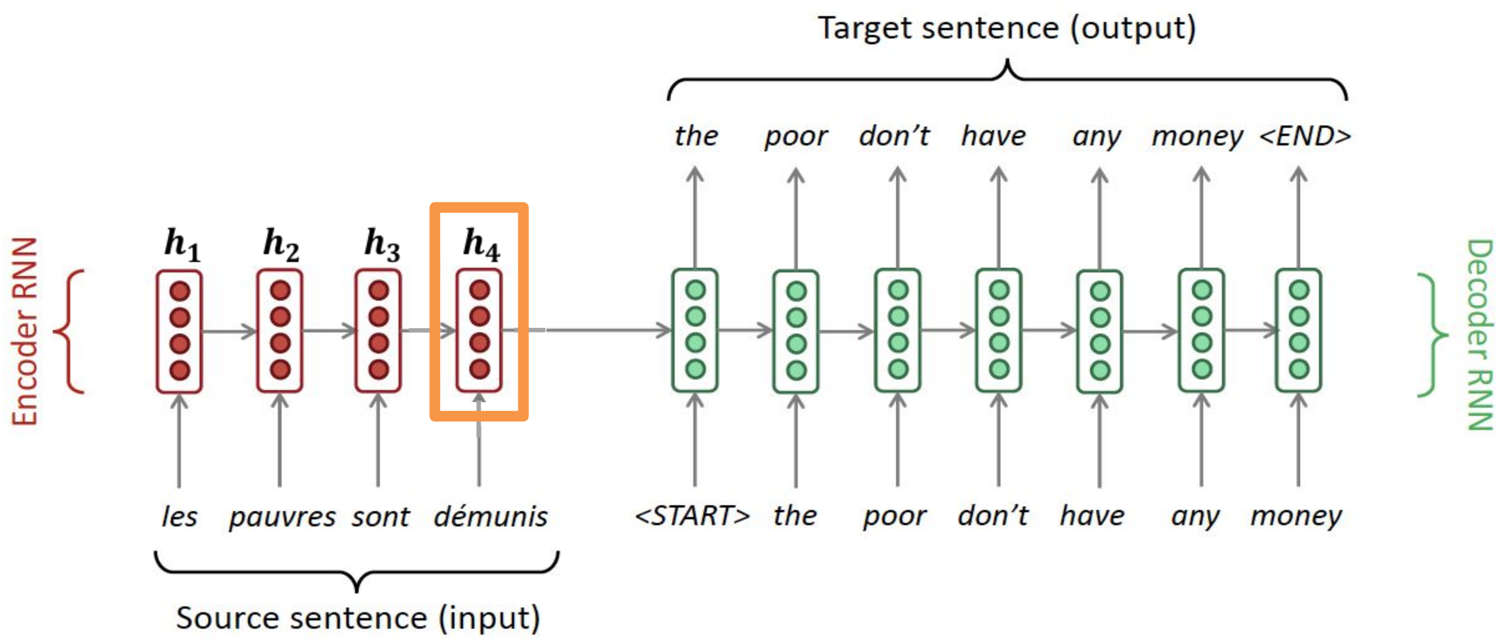
Seq2Seq: the bottleneck problem

- Encoding of source sentence: a fixed length vector $\boldsymbol{h_4}$
- Need to capture all necessary information of source sentence



Seq2Seq: the bottleneck problem

- Encoding of source sentence: a fixed length vector $\boldsymbol{h_4}$
- Need to capture all necessary information of source sentence
- Information bottleneck, especially when source sentence is long



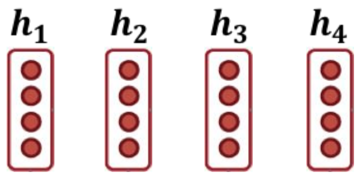
Attention: thinking process

How to solve the bottleneck problem?

Attention: thinking process

How to solve the bottleneck problem?

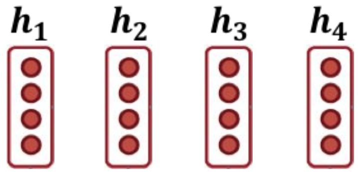
Instead of only using only $\boldsymbol{h}_4$, let's use all encoder hidden states!



Attention: thinking process

How to solve the bottleneck problem?

Instead of only using only $\boldsymbol{h_4}$, let's use all encoder hidden states!

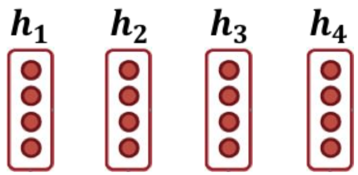


How do we deal with variable length input sequence?

Attention: thinking process

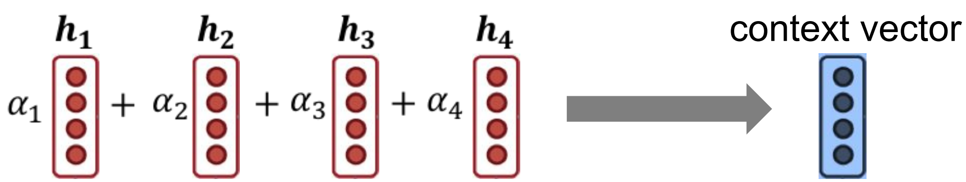
How to solve the bottleneck problem?

Instead of only using only $\boldsymbol{h}_4$, let's use all encoder hidden states!



How do we deal with variable length input sequence?

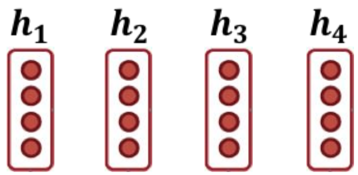
Let's do a weighted sum of all encoder hidden states!



Attention: thinking process

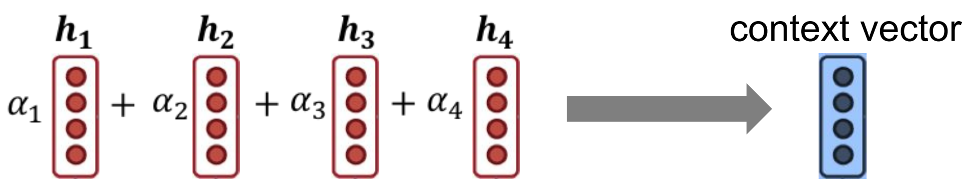
How to solve the bottleneck problem?

Instead of only using only $\boldsymbol{h_4}$, let's use all encoder hidden states!



How do we deal with variable length input sequence?

Let's do a weighted sum of all encoder hidden states!

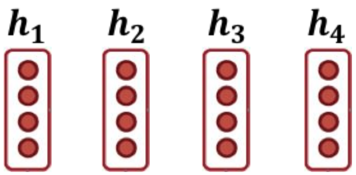


How do we get the weights α_i?

Attention: thinking process

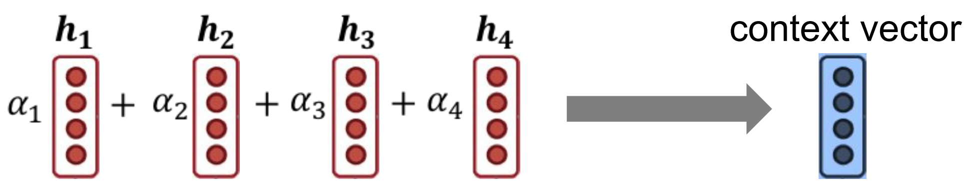
How to solve the bottleneck problem?

Instead of only using only $\boldsymbol{h}_4$, let's use all encoder hidden states!



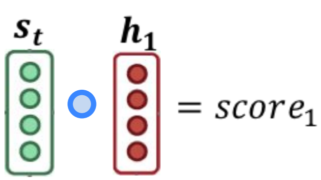
How do we deal with variable length input sequence?

Let's do a weighted sum of all encoder hidden states!



How do we get the weights α_i?

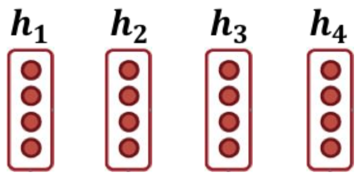
Step 1: dot product



Attention: thinking process

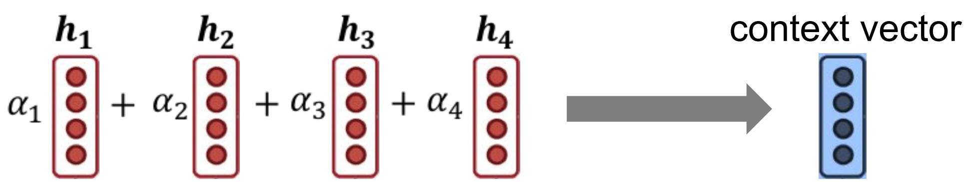
How to solve the bottleneck problem?

Instead of only using only $\boldsymbol{h}_4$, let's use all encoder hidden states!



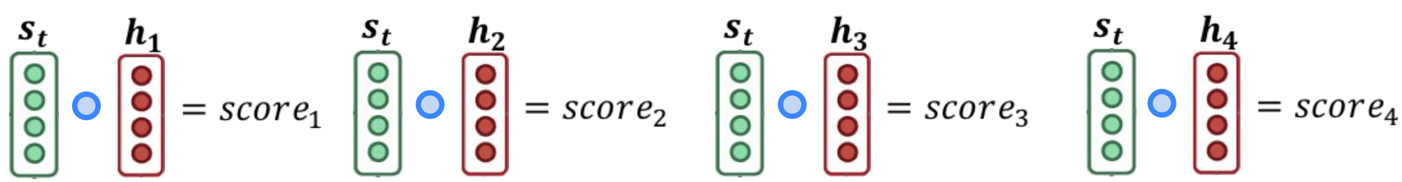
How do we deal with variable length input sequence?

Let's do a weighted sum of all encoder hidden states!



How do we get the weights α_i?

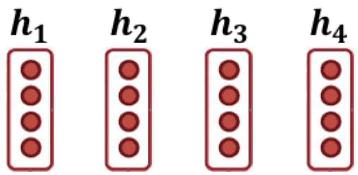
Step 1: dot product



Attention: thinking process

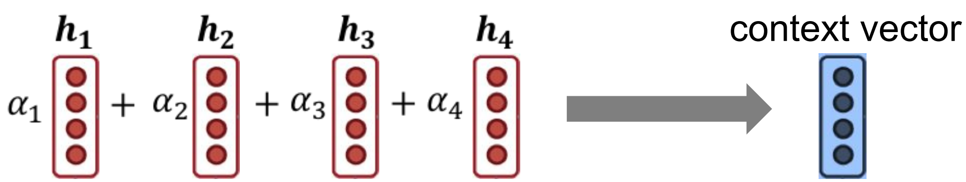
How to solve the bottleneck problem?

Instead of only using only $\boldsymbol{h_4}$, let's use all encoder hidden states!



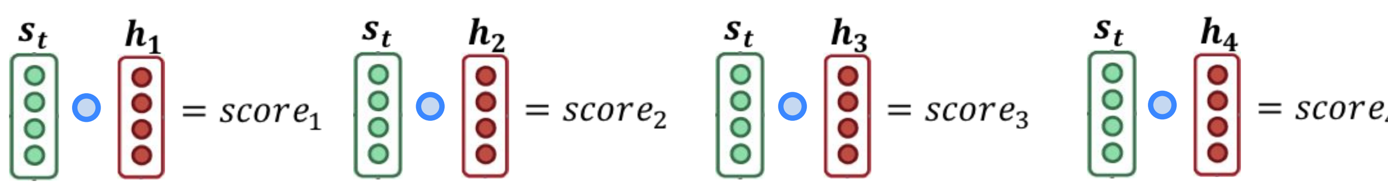
How do we deal with variable length input sequence?

Let's do a weighted sum of all encoder hidden states!



How do we get the weights α_i?

Step 1: dot product



Step 2: softmax $\alpha_i = \frac{\exp(score_i)}{\sum_{j=1}^{4} score_j}$

Attention: thinking process

How to solve the bottleneck problem?

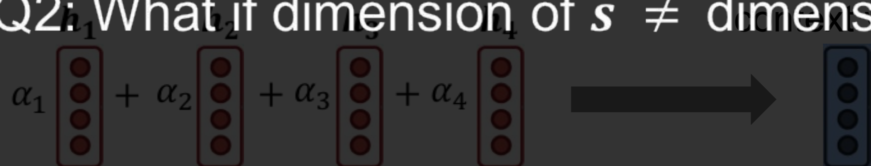
Instead of only using only $\boldsymbol{h_4}$, let's use all encoder hidden states!

Q1: Why dot product?

How do we deal with variable length input sequence?

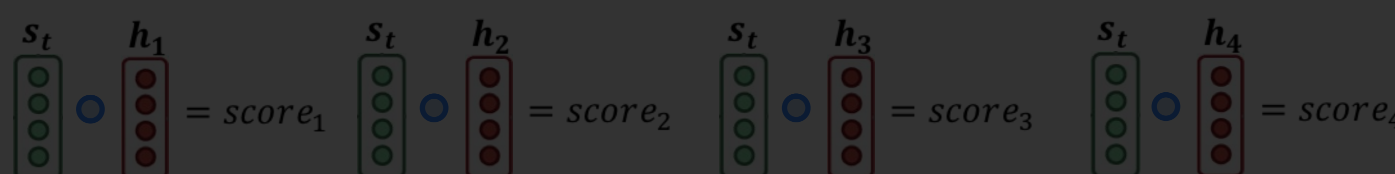
Let's do a weighted sum of all encoder hidden states!

Q2: What if dimension of $\boldsymbol{s}$ $\neq$ dimension of $\boldsymbol{h}$?



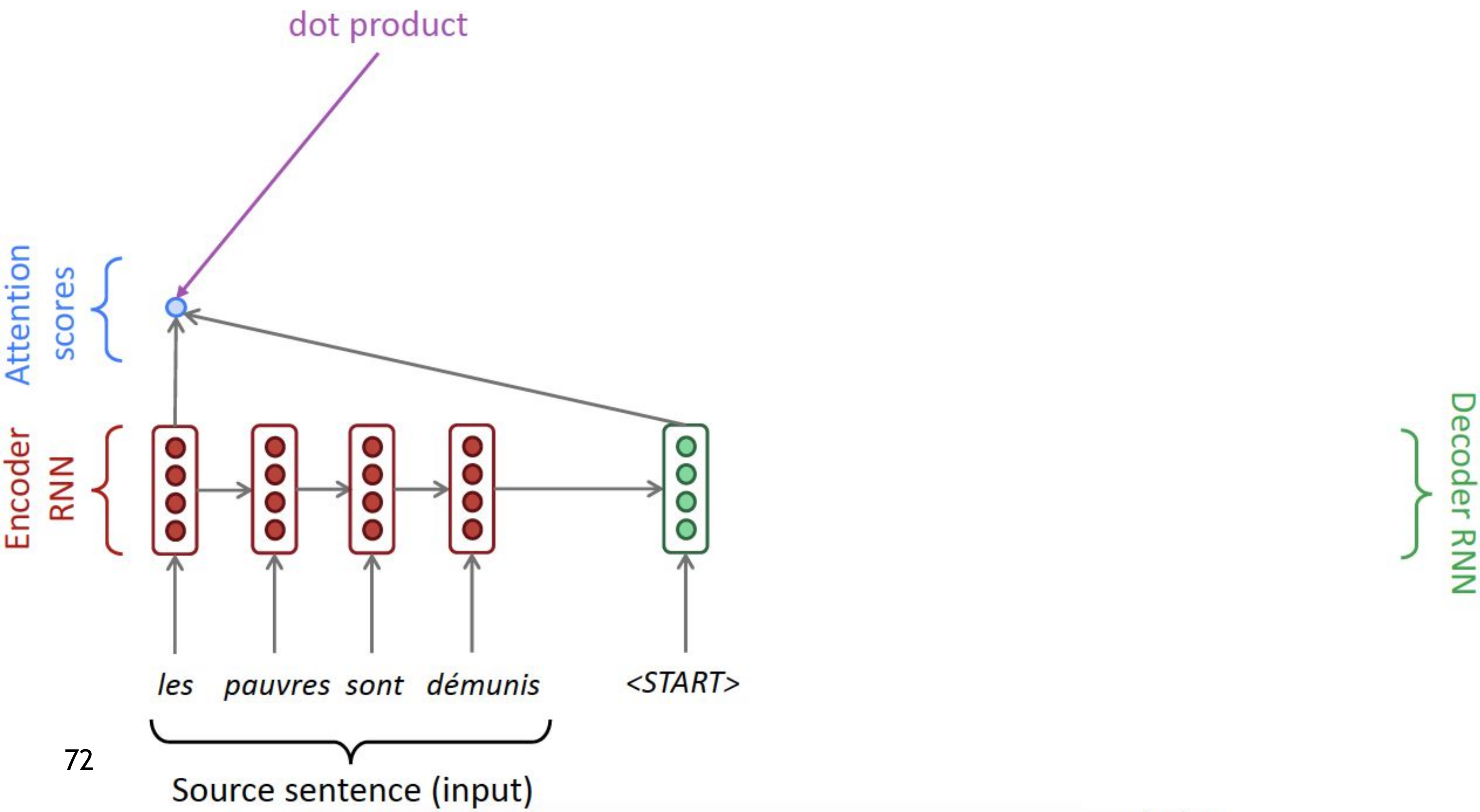
How do we get the weights α_i?

Step 1: dot product

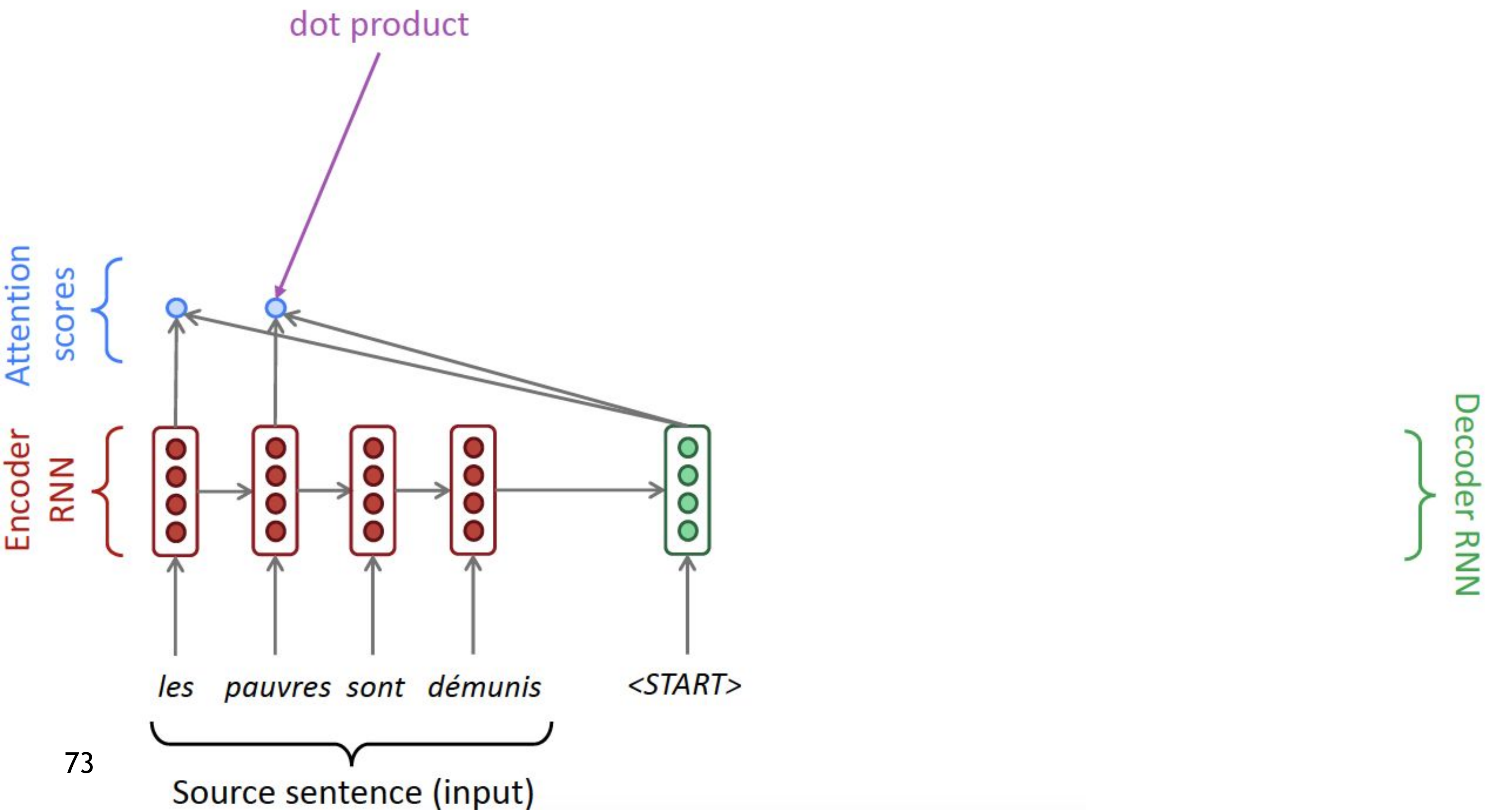


Step 2: softmax $\alpha_i = \frac{\exp(score_i)}{\sum_{j=1}^{4} score_j}$

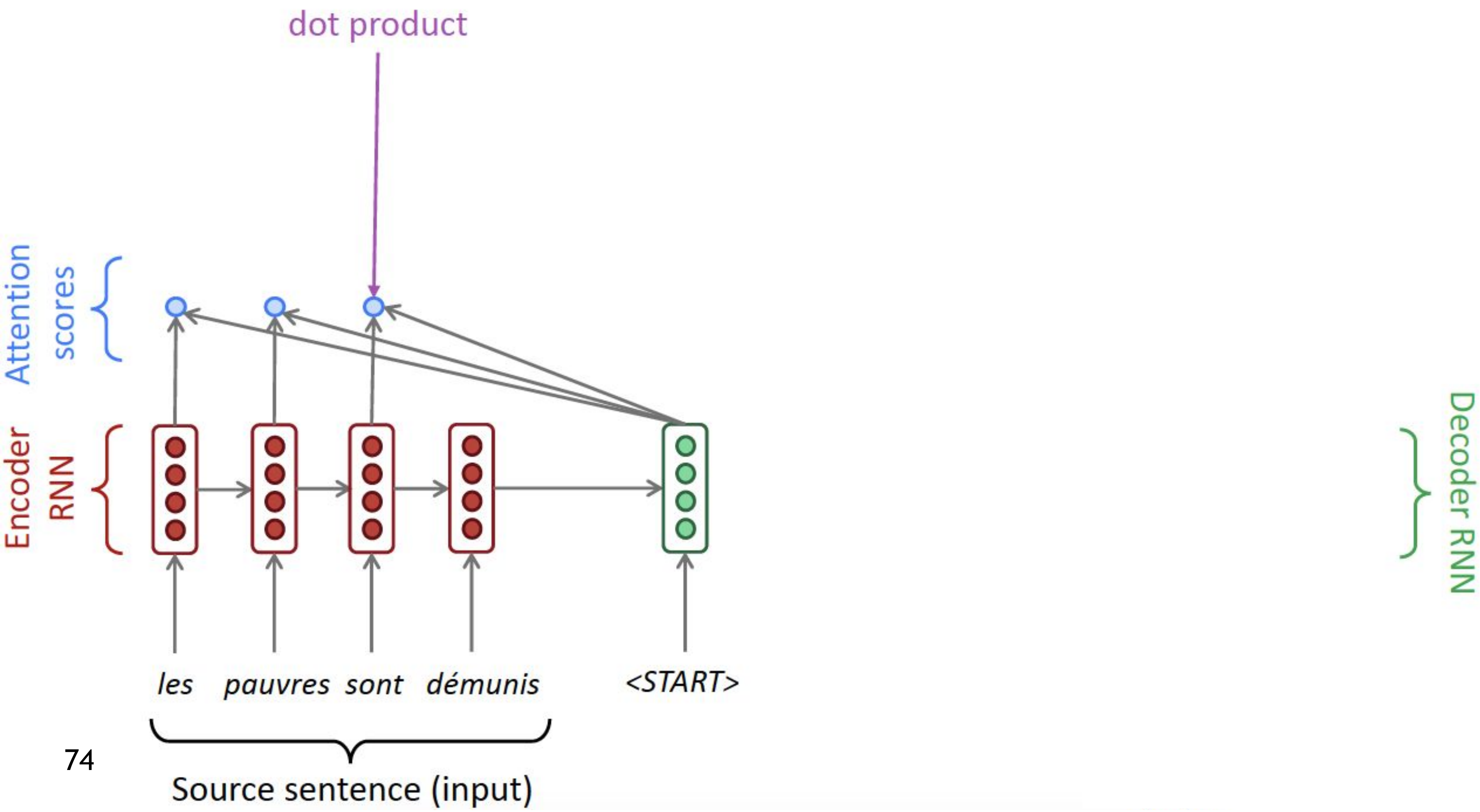
Seq2Seq with Attention



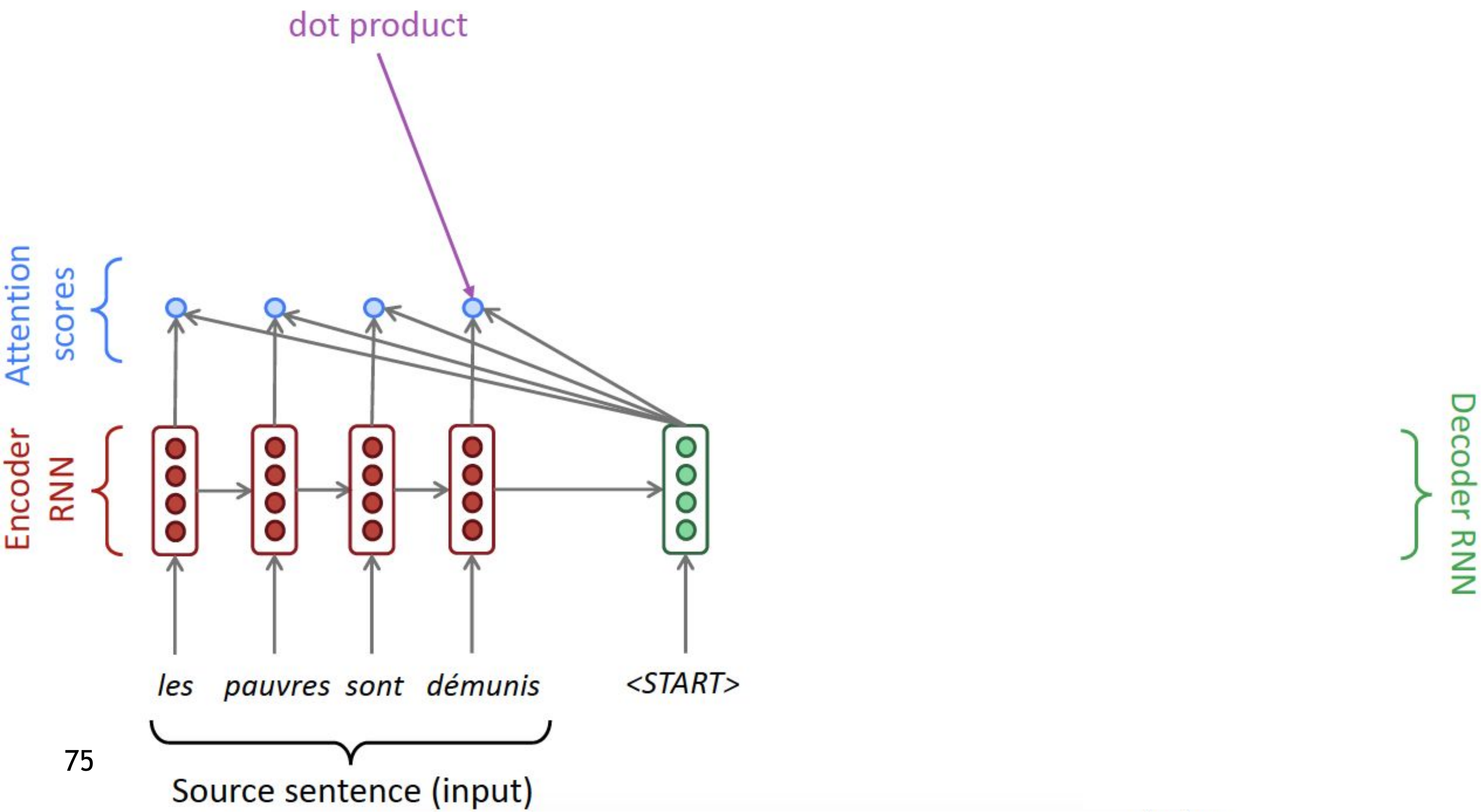
Seq2Seq with Attention



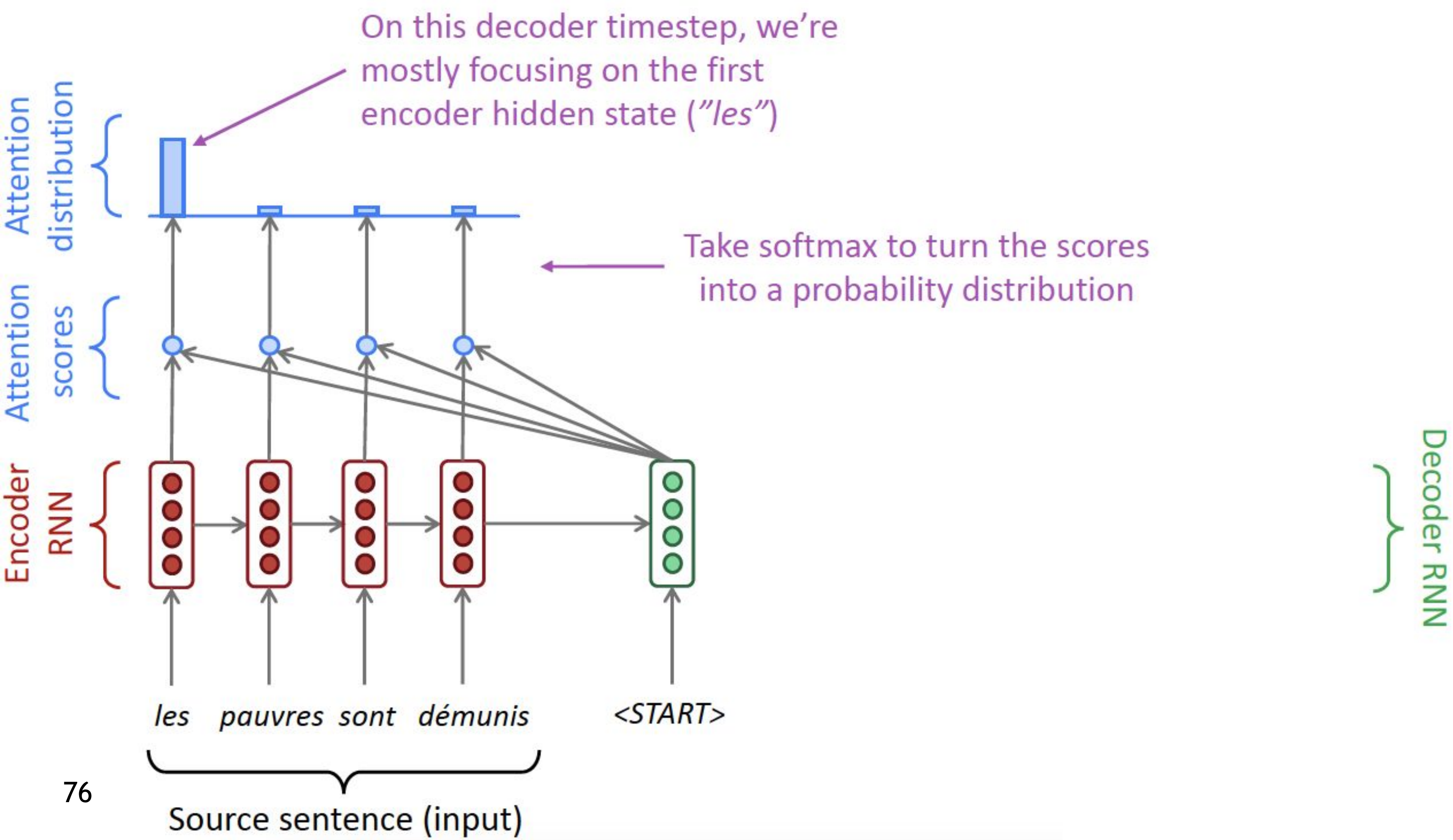
Seq2Seq with Attention



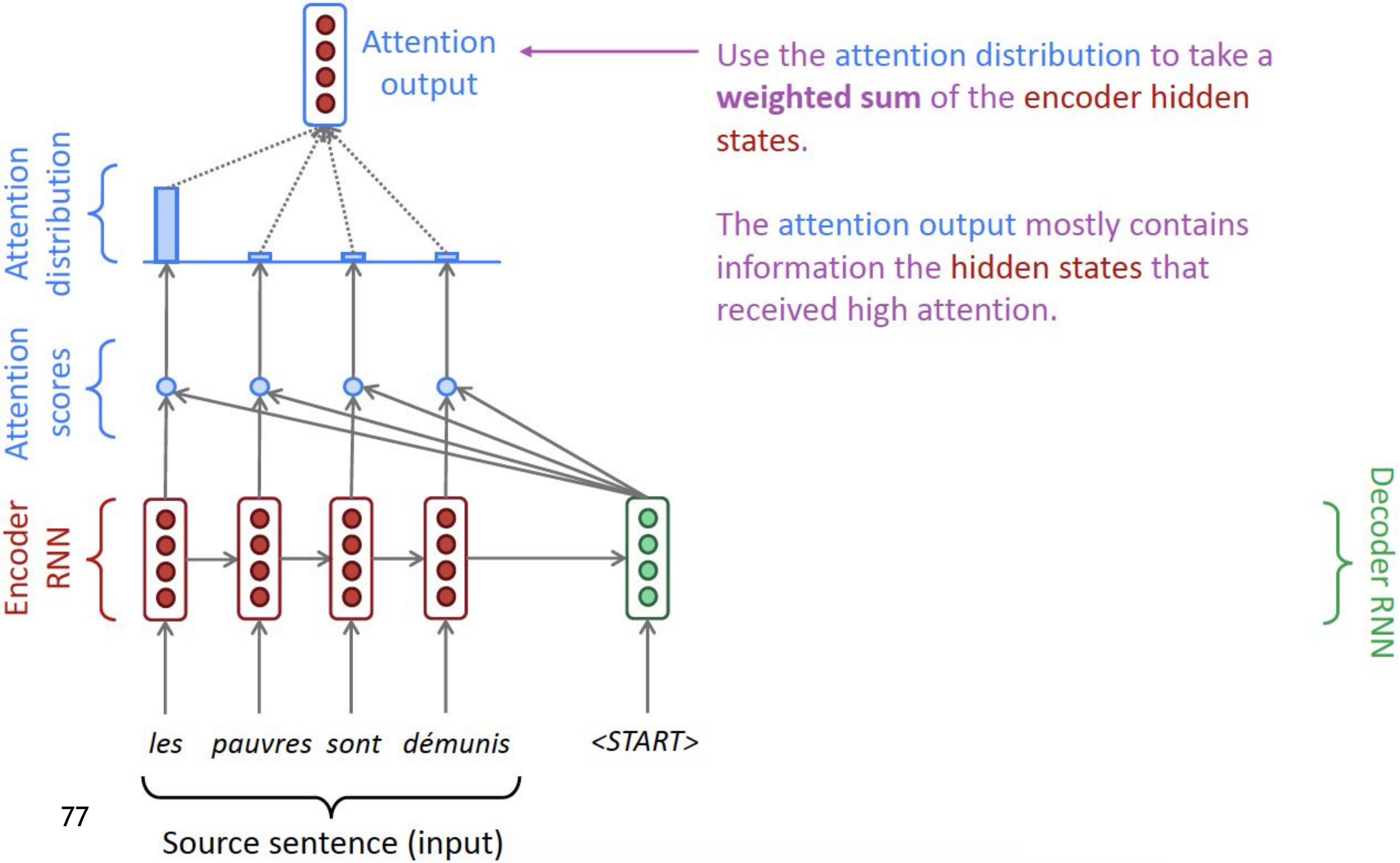
Seq2Seq with Attention



Seq2Seq with Attention



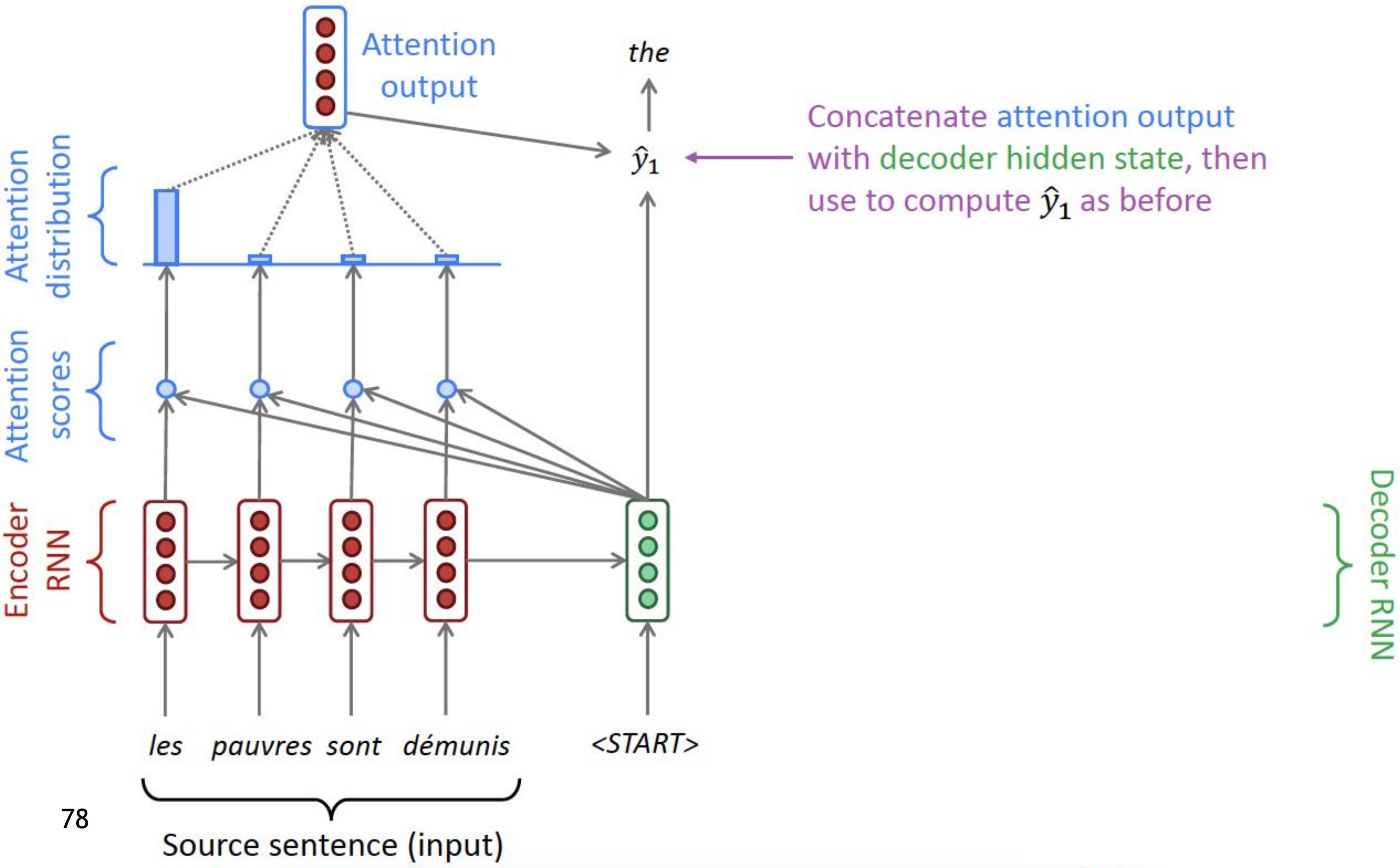
Seq2Seq with Attention



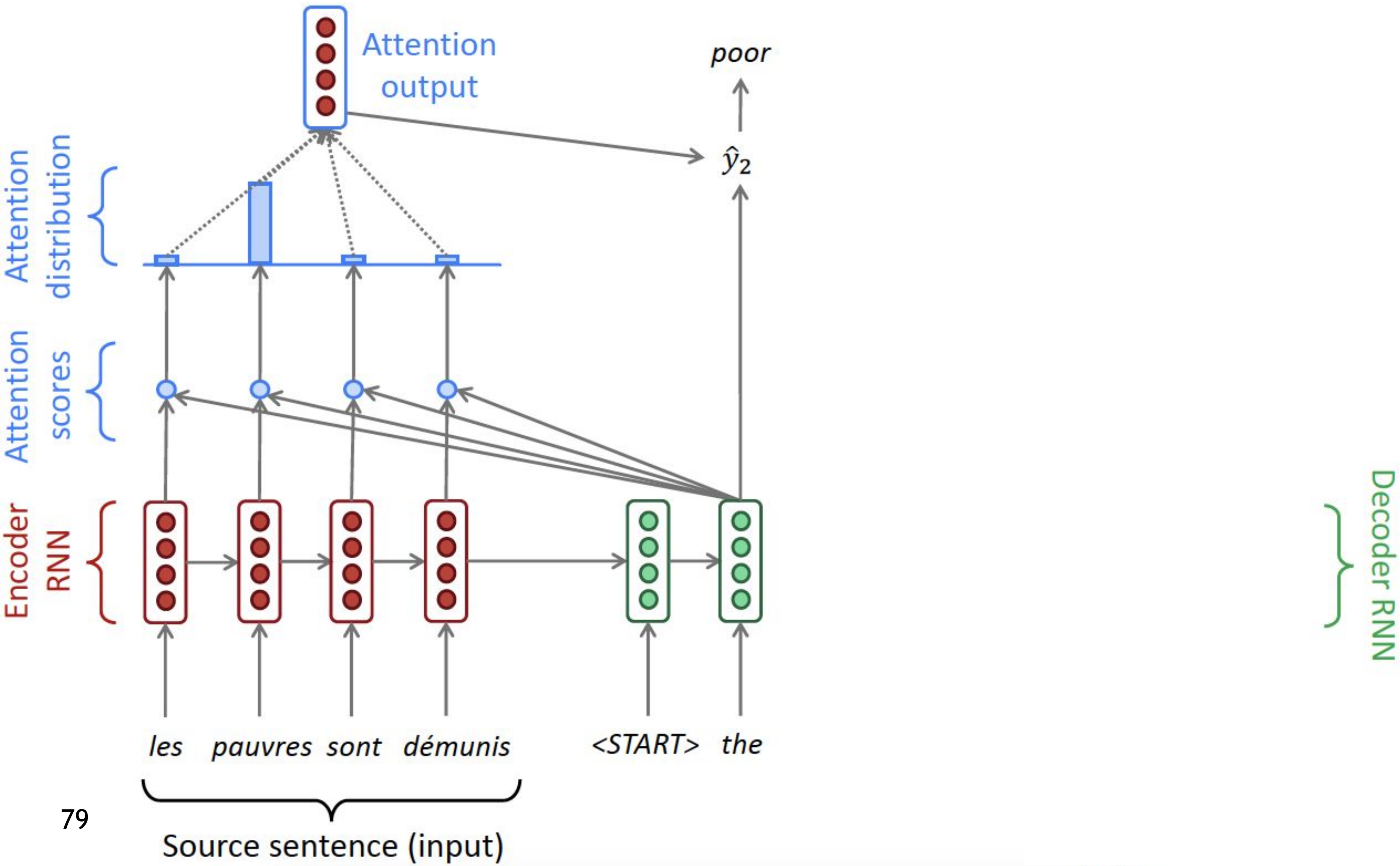
Use the attention distribution to take a **weighted sum** of the encoder hidden states.

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

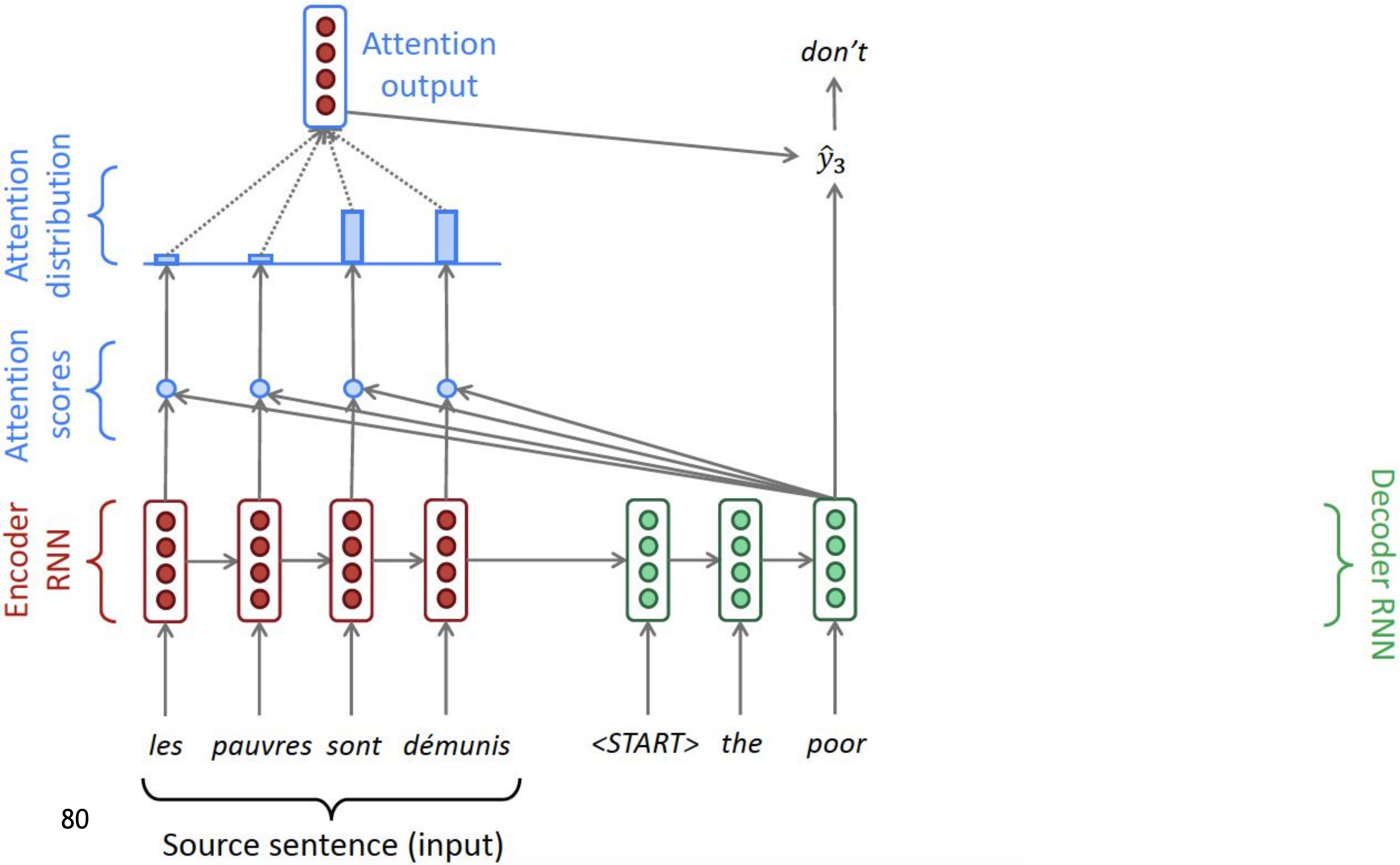
Seq2Seq with Attention



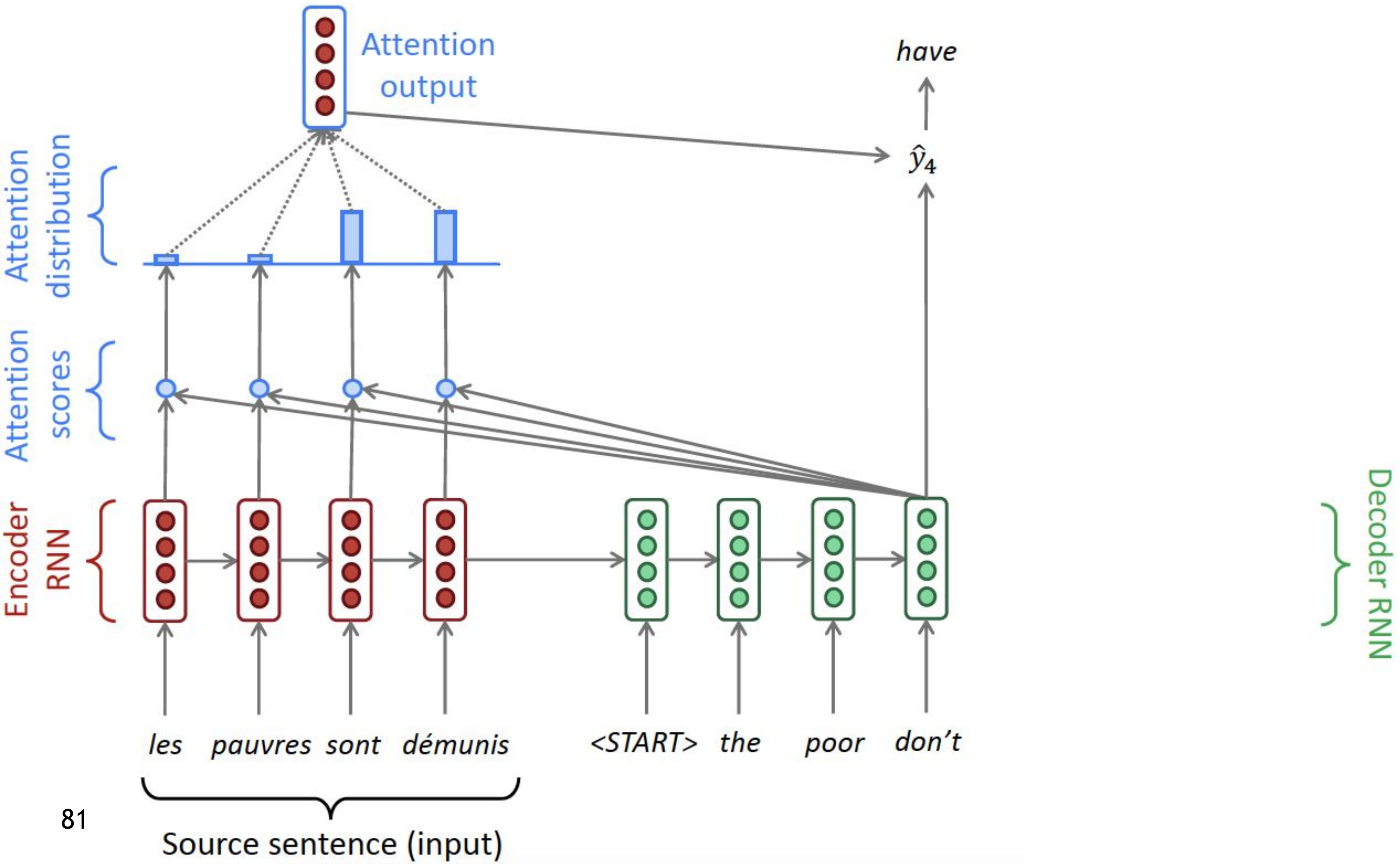
Seq2Seq with Attention



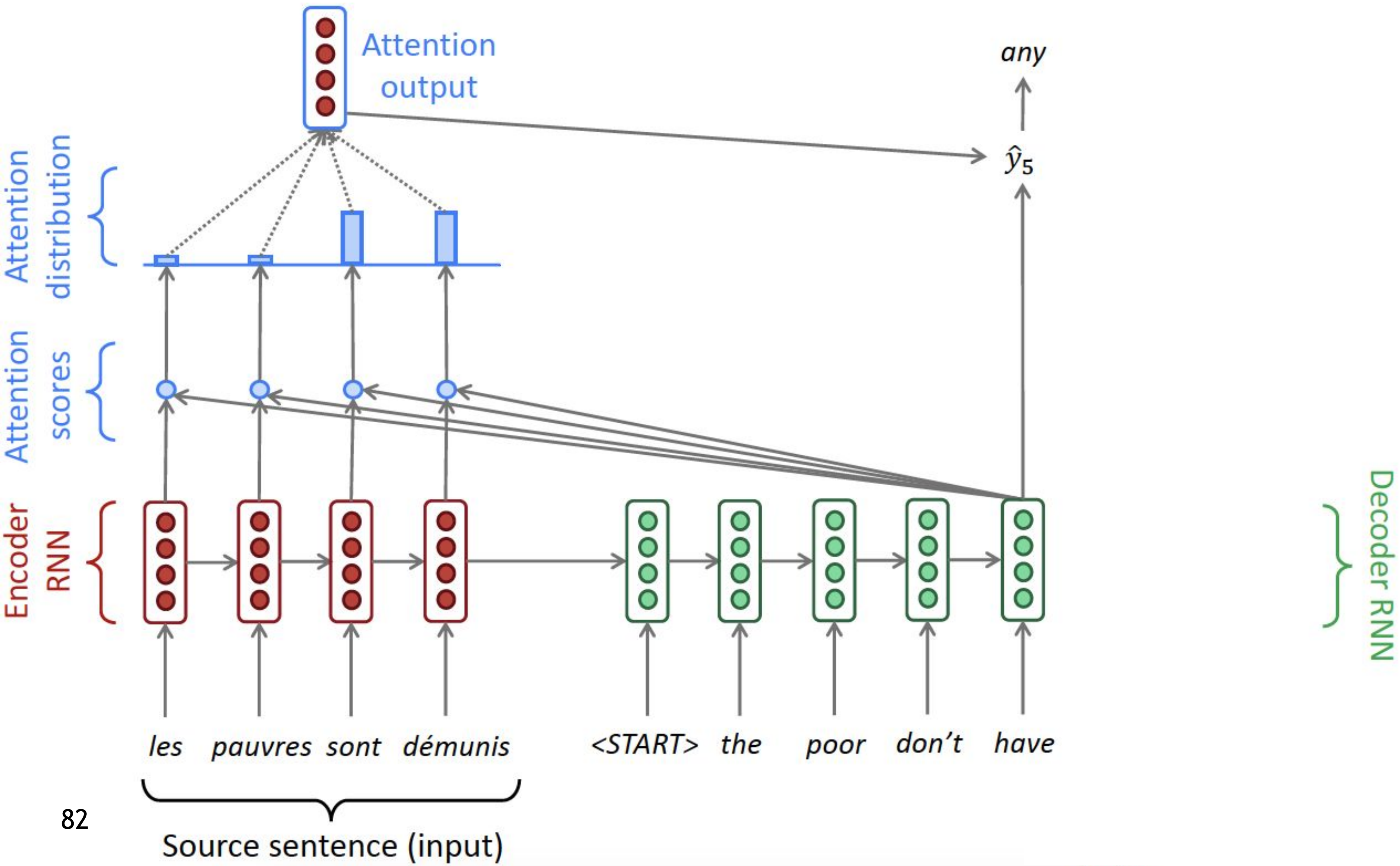
Seq2Seq with Attention



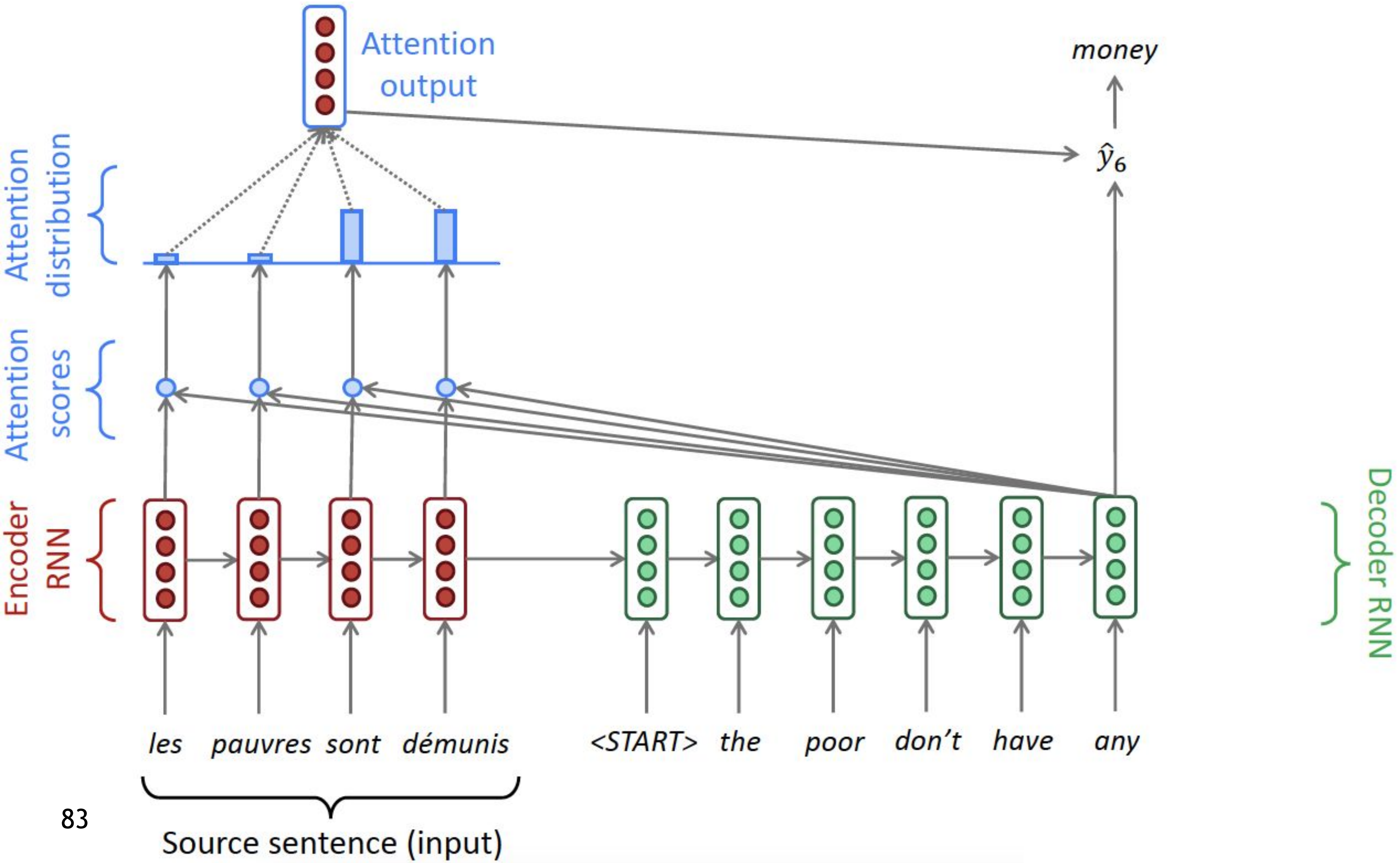
Seq2Seq with Attention



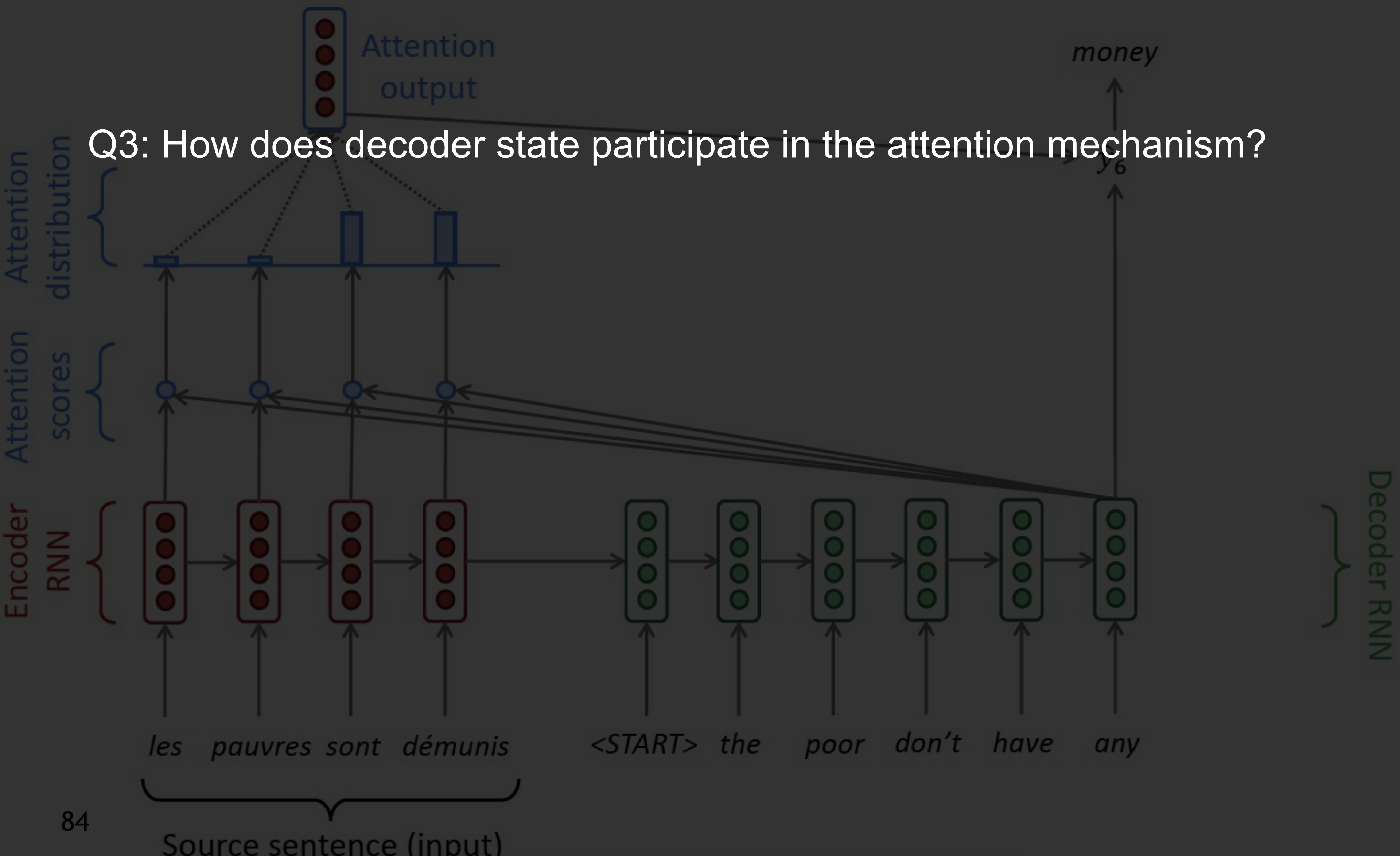
Seq2Seq with Attention



Seq2Seq with Attention



Seq2Seq with Attention



Q3: How does decoder state participate in the attention mechanism?

Attention: in equations

- We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$
- On timestep *t*, we have decoder hidden state $s_t \in \mathbb{R}^h$
- We get the attention scores $\boldsymbol{e}^t$ 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 take softmax to get the attention distribution α^t for this step (this is a probability distribution and sums to 1)

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

- We use α^t to take a weighted sum of the encoder hidden states to get the attention output $\boldsymbol{a}_t$

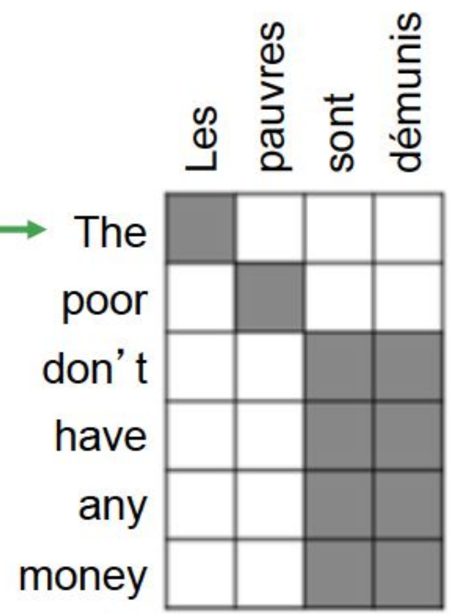
$$\boldsymbol{a}_t = \sum_{i=1}^N \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

- Finally we concatenate the attention output $\boldsymbol{a}_t$ with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

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

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



Seq2Seq Applications

Seq2seq and attention applications

- Summarization
- Dialogue systems
- Speech recognition
- Image captioning

Text summarization

“*Automatic summarization is the process of shortening a text document with software, in order to create a summary with the major points of the original document.”*

volume of **transactions** at the **nigerian stock exchange** has **continued its decline** since last week , a nse official said thursday . the latest statistics showed that a total of ##.### million shares valued at ###.### million naira -lrb- about #.### million us dollars -rrb- were traded on wednesday in #,### deals.	transactions dip at nigerian stock exchange

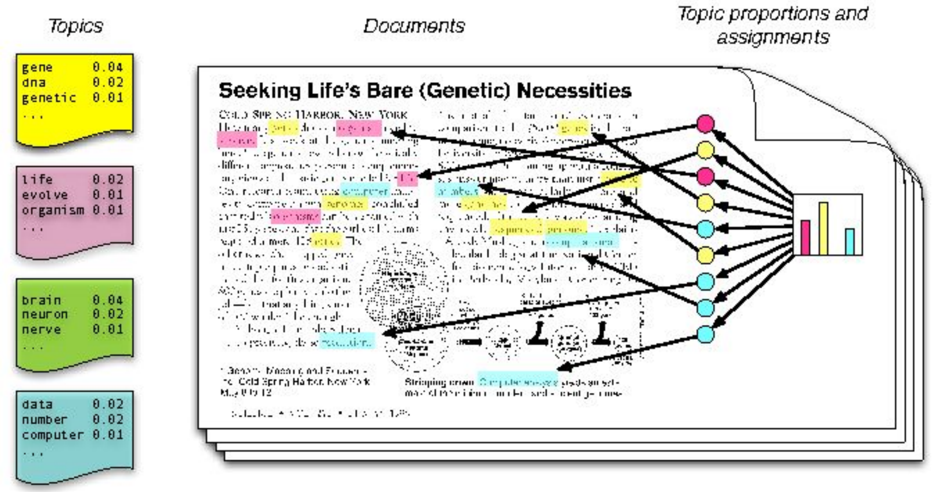
Text summarization (before seq2seq)

Allahyari, Mehdi, et al. "Text summarization techniques: a brief survey."

Extractive Topic Representation Approaches:

- Latent Semantic Analysis
- Bayesian Topic Models
    - Latent Dirichlet Allocation

Latent Dirichlet allocation (LDA)



- Each **topic** is a distribution over words
- Each **document** is a mixture of corpus-wide topics
- Each **word** is drawn from one of those topics

Text summarization (before seq2seq)

Their drawbacks:

- They consider the sentences as independent of each other
- Many of these previous text summarization techniques do not consider the semantics of words.
- The soundness and readability of generated summaries are not satisfactory.

Text summarization (seq2seq)

Nallapati, Ramesh, et al. "Abstractive text summarization using sequence-to-sequence rnns and beyond."

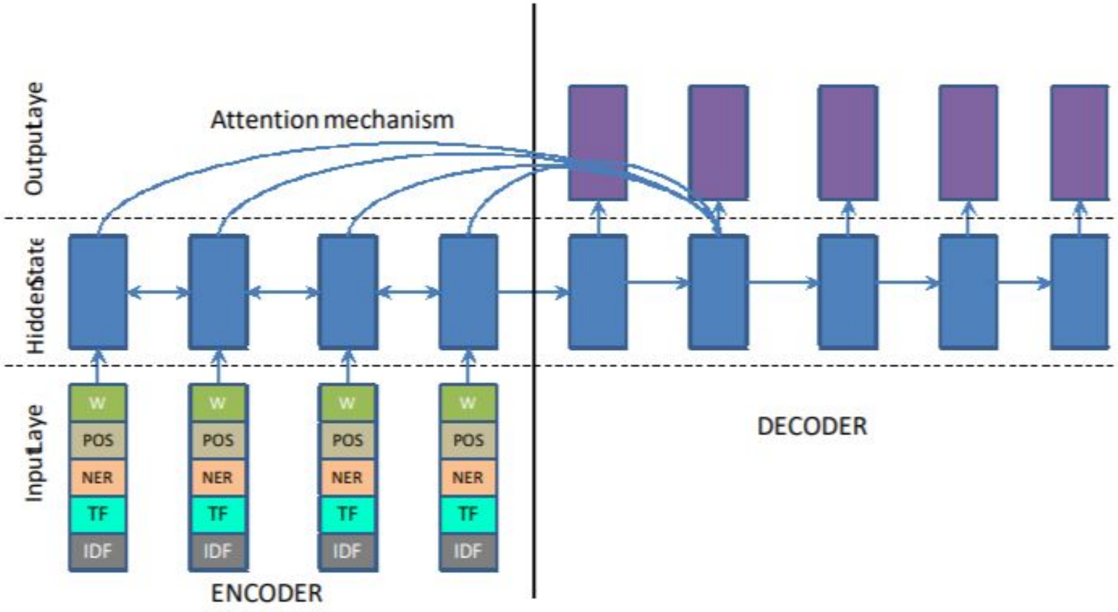
Abstractive Summary - not a mere selection of a few existing passages or sentences extracted from the source

Neural Machine Translation	Abstractive Text Summarization
Output depends on source length	Output is short
Retains the original content	Compresses the original content
Strong notion of one-to-one word level alignment	Less obvious notion of such one-to-one alignment

Text summarization (seq2seq)

Feature-rich encoder:

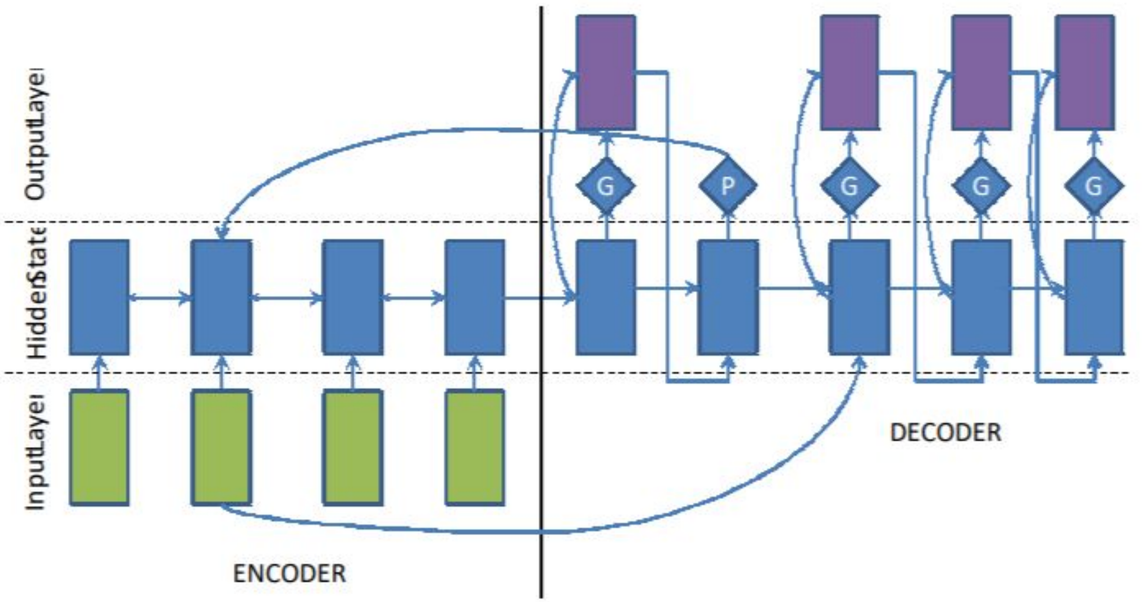
- POS, NER, TF, IDF



Text summarization (seq2seq)

Switch generator/pointer for OOV

- Switch is on → produces a word from its
  target vocabulary
- Switch is off → generates a pointer to a word-position in the Source → copied into the summary

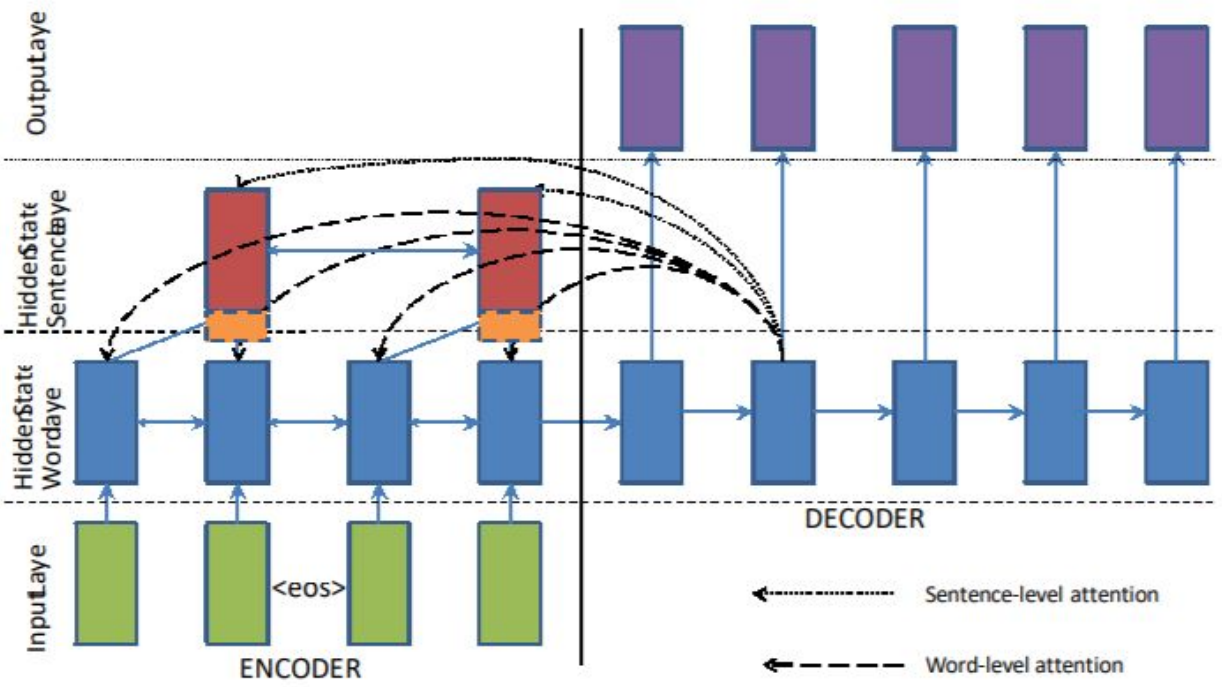


Text summarization (seq2seq)

Identify the key sentences:

- 2 levels of importance → 2 BiRNNs
- Attention mechanisms operate at both levels simultaneously.
- Word-level attention is reweighted:

$$P^a(j) = \frac{P^a_w(j) P^a_s(s(j))}{\sum_{k=1}^{N_d} P^a_w(k) P^a_s(s(k))}$$



Dialogue systems

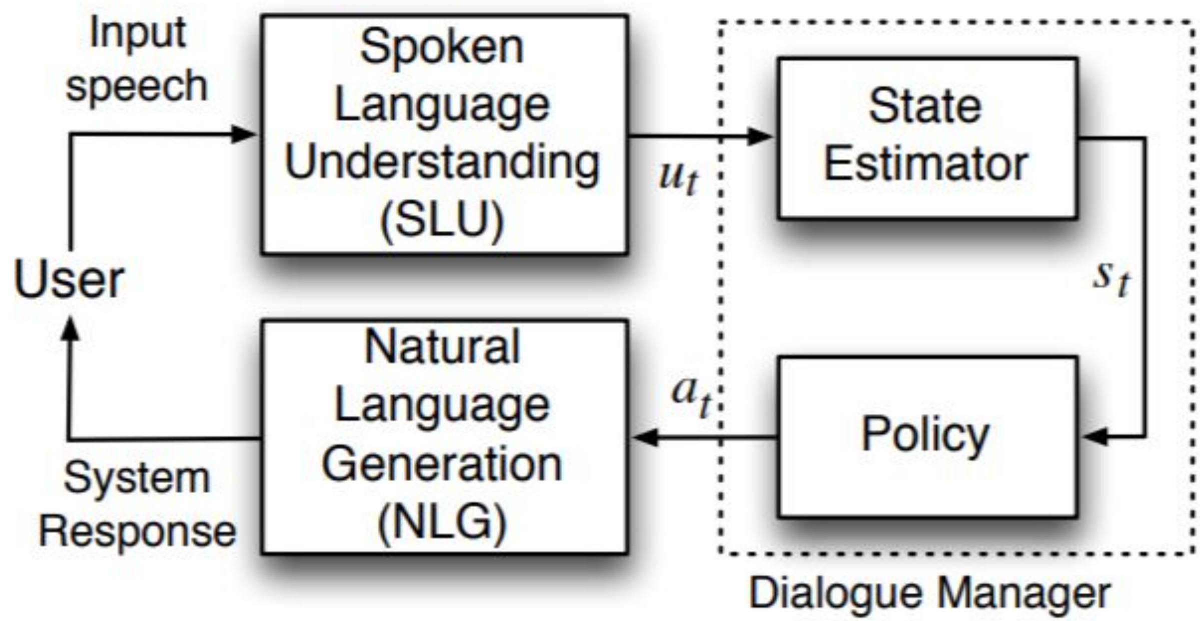
“Dialogue systems, also known as interactive conversational agents, virtual agents and sometimes chatterbots, are used in a wide set of applications ranging from technical support services to language learning tools and entertainment.”

Dialogue systems (before seq2seq)

Young, Steve, et al. "Pomdp-based statistical spoken dialog systems: A review."

Goal-driven system:

- At each *t*, SLU converts input to a semantic representation u_t
- System updates internal state and determines next action a_t, then converted to output speech



Dialogue systems (before seq2seq)

Their drawbacks:

- Use handcrafted features for the state and action space representations
- Require a large corpora of annotated task-specific simulated conversations
  - → Time consuming to deploy

Dialogue systems (seq2seq)

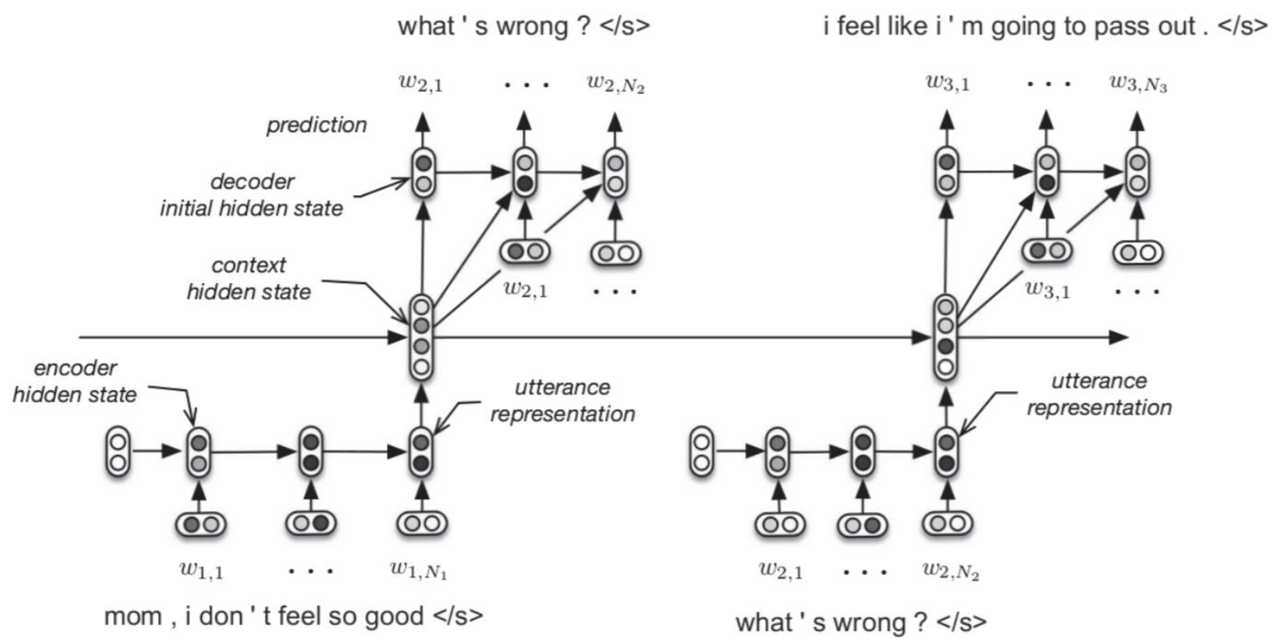
Serban, Iulian Vlad, et al. "Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models."

- End-to-end trainable, non-goal-driven systems based on generative probabilistic models.

Dialogue systems (seq2seq)

Hierarchical Recurrent Encoder - Decoder

- Encoder map each utterance (last token) to an utterance vector
- Higher-level context RNN:
  - $c_n = f(c_{n-1}, u_n)$



Speech recognition

From RNN to Speech Recognition:

- RNN requires pre-segmented input data
  - Word sequences (discrete) vs. audio sequences (continuous)
- RNNs can only be trained to make a series of independent label classifications
  - Network outputs must be post-processed

Speech recognition

Prabhavalkar, Rohit, et al. "A comparison of sequence-to-sequence models for speech recognition."

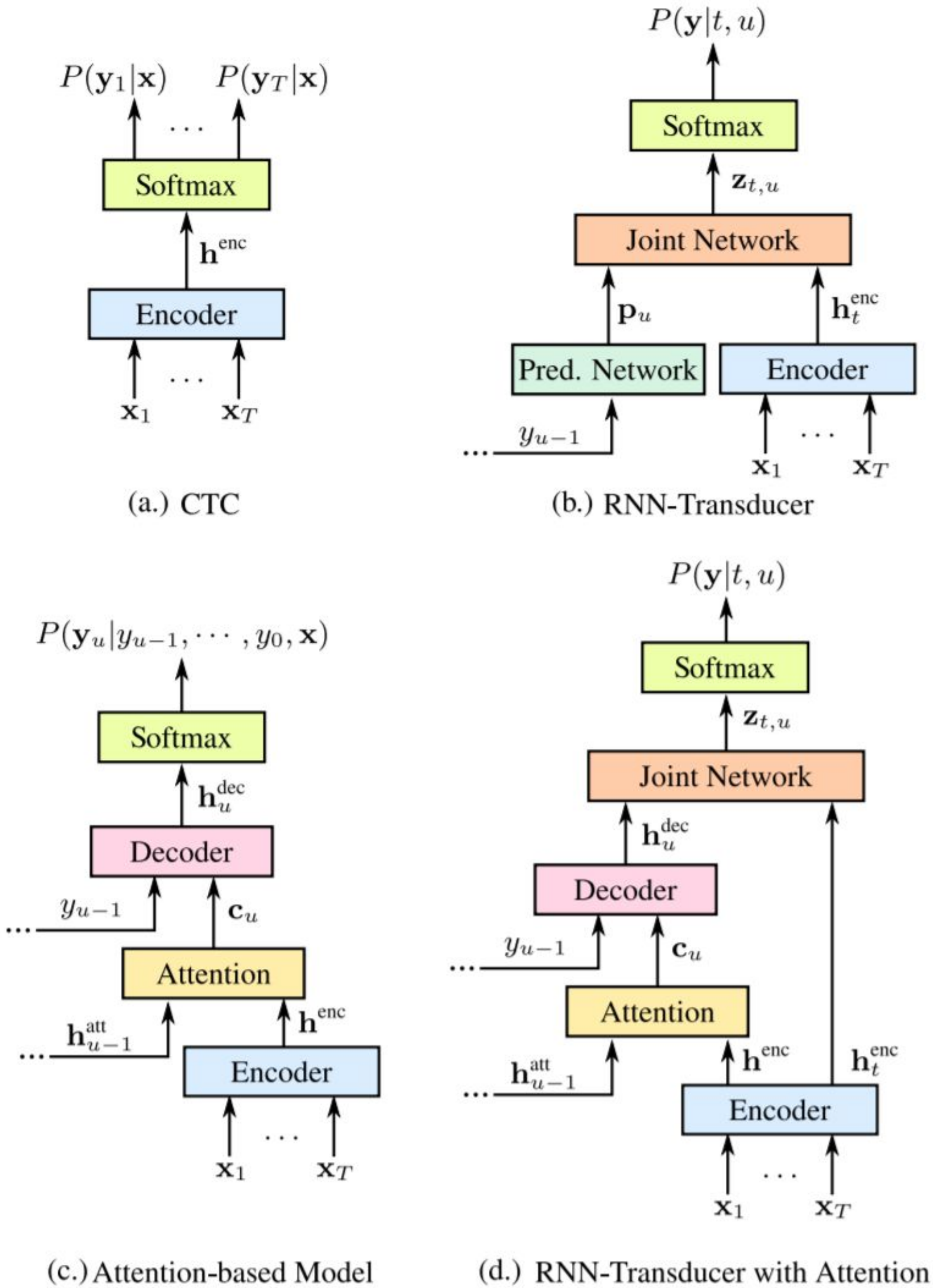


Image captioning

Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention."

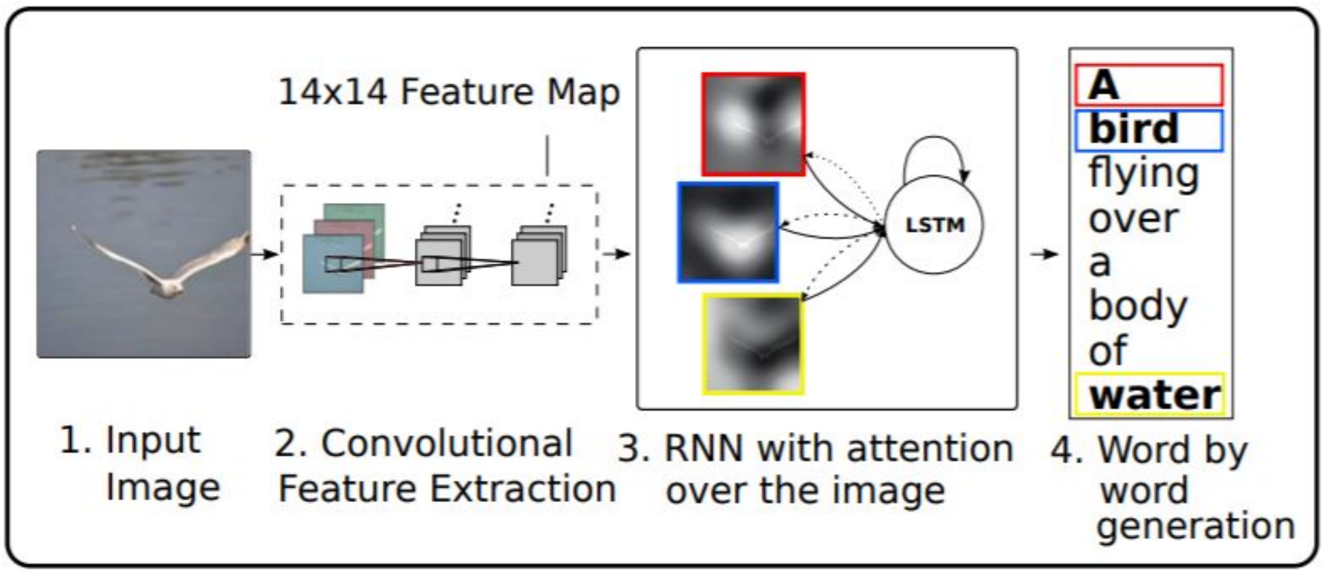
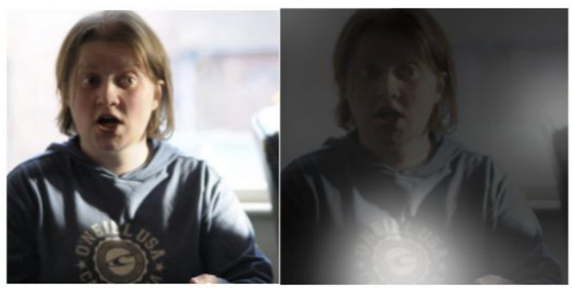


Image captioning

Location variable s_t: where to put attention when generating the t^{th} word

http://kelvinxu.github.io/projects/capgen.html

A large white bird standing in a forest.



A woman holding a clock in her hand.

A man wearing a hat and
a hat on a skateboard.

A person is standing on a beach
with a surfboard.

A woman is sitting at a table
with a large pizza.

A man is talking on his cell phone
while another man watches.

Thank You!