**Natural Language Processing**

**with Deep Learning**

**CS224N/Ling284**



Machine Translation,

Sequence-to-sequence and Attention

**Abigail See, Matthew Lamm**

# Overview

• Introduce a new task: Machine Translation

## is a major use-case of

* Introduce a new neural architecture: sequence-to-sequence

**is improved by**

* Introduce a new neural technique: attention

**Section 1: Pre-Neural Machine Translation**

# Machine Translation

**Machine Translation (MT)** is the task of translating a sentence *x* from one language (the source language) to a sentence *y* in another language (the target language).

*x: L'homme est né libre, et partout il est dans les fers*

*y: Man is born free, but everywhere he is in chains*

*-* Rousseau

# 1950s: Early Machine Translation

Machine Translation research began in the early 1950s.

* Russian → English

(motivated by the Cold War!)

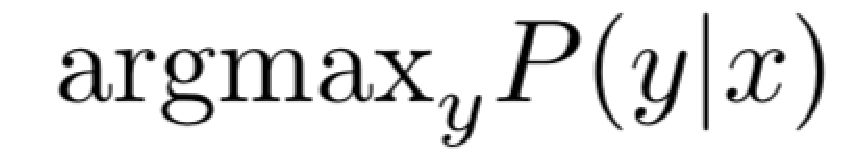
**1 minute video showing 1954 MT:** <https://youtu.be/K-HfpsHPmvw>

* Systems were mostly rule-based, using a bilingual dictionary to map Russian words to their English counterparts

# 1990s-2010s: Statistical Machine Translation

* Core idea: Learn a probabilistic model from data
* Suppose we’re translating French → English.
* We want to find best English sentence *y,* given French sentence

*x*



* Use Bayes Rule to break this down into two components to be learnt separately:



**Translation Model**

Models how words and phrases

should be translated (

*fidelity*

)

.

Learnt from parallel data.

**Language Model**

Models how to write

good English (

*fluency*

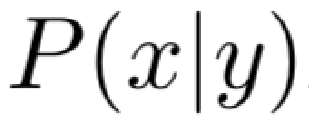
)

.

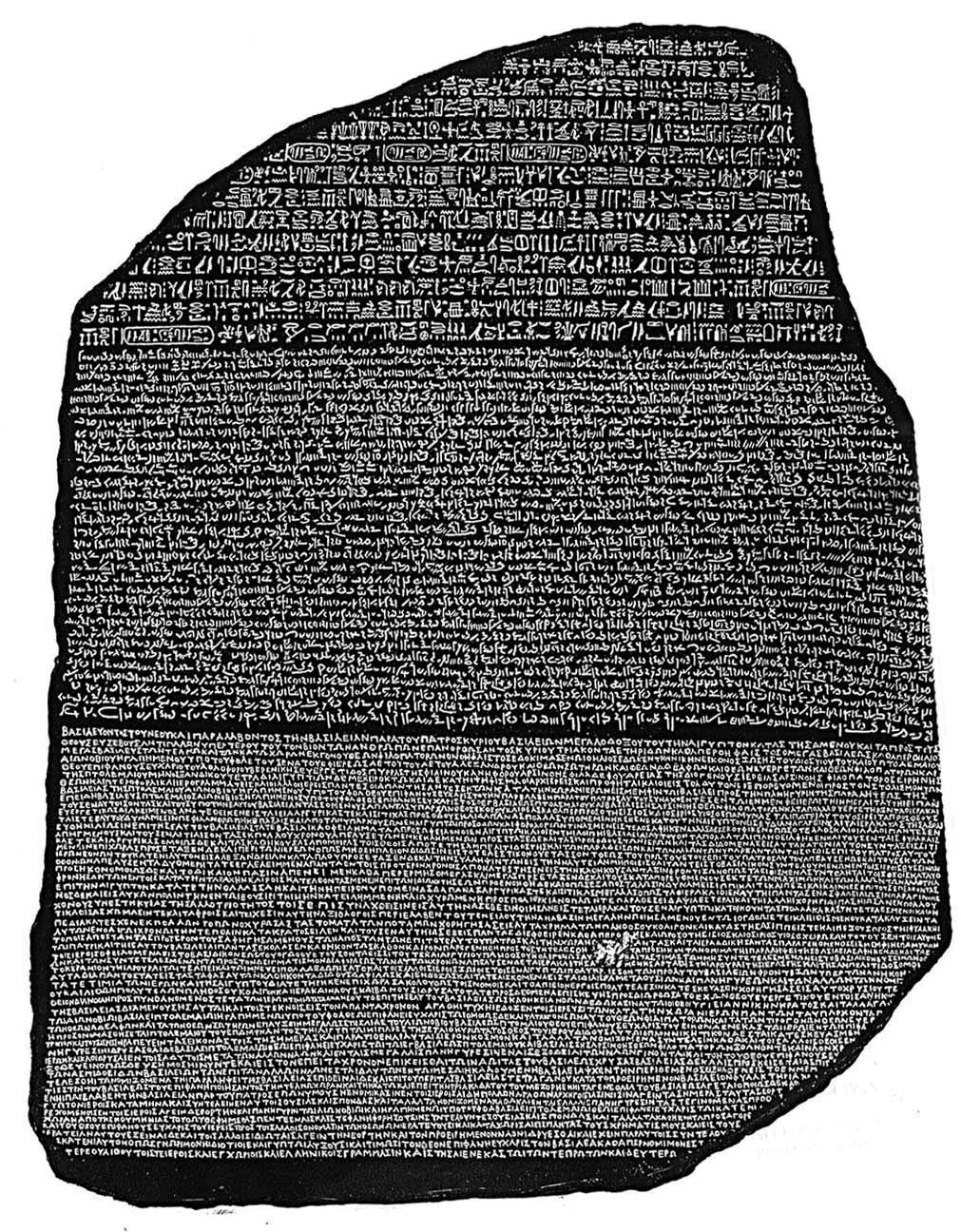
Learnt from monolingual

data.

# 1990s-2010s: Statistical Machine Translation

* Question: How to learn translation model ?
* First, need large amount of parallel data

(e.g. pairs of human-translated French/English sentences)



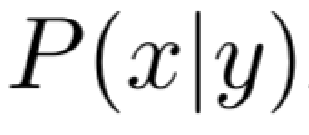
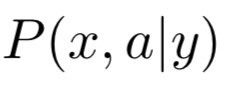
Ancient Egyptian

Demotic

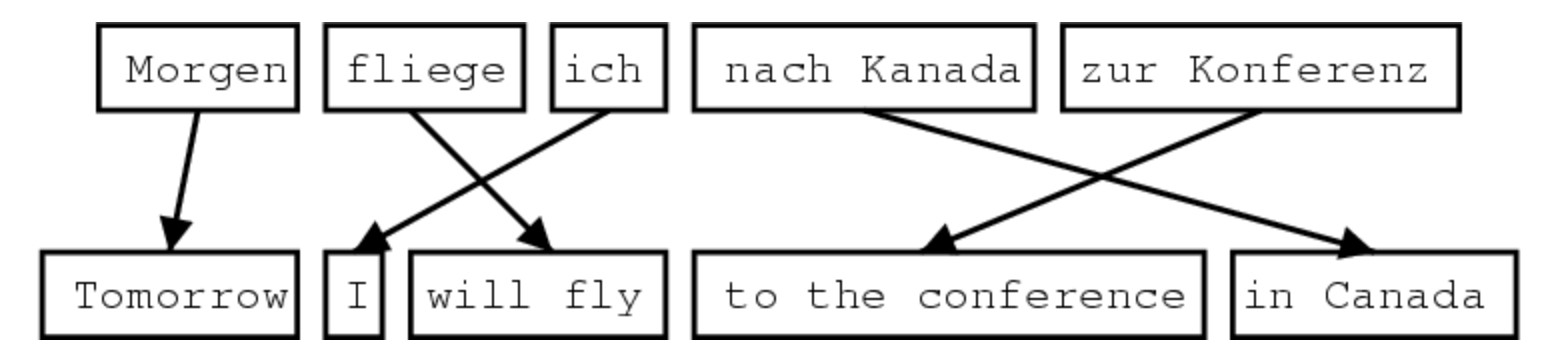
Ancient Greek

**The Rosetta Stone**

# Learning alignment for SMT

* Question: How to learn translation model from the parallel corpus?
* Break it down further: Introduce latent a variable into the model:

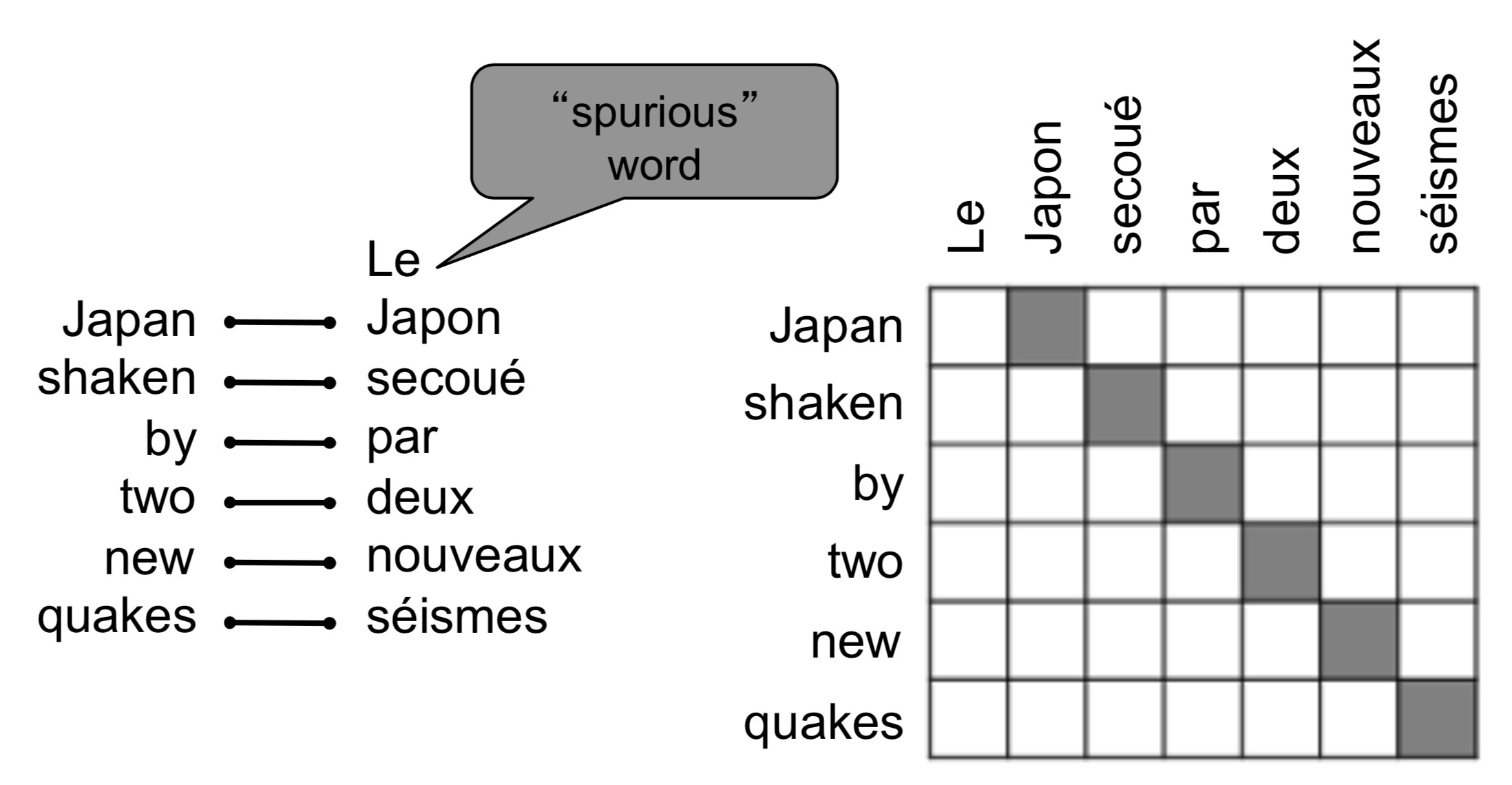
where *a* is the alignment, i.e. word-level correspondence between source sentence *x* and target sentence *y*



**What is alignment?**

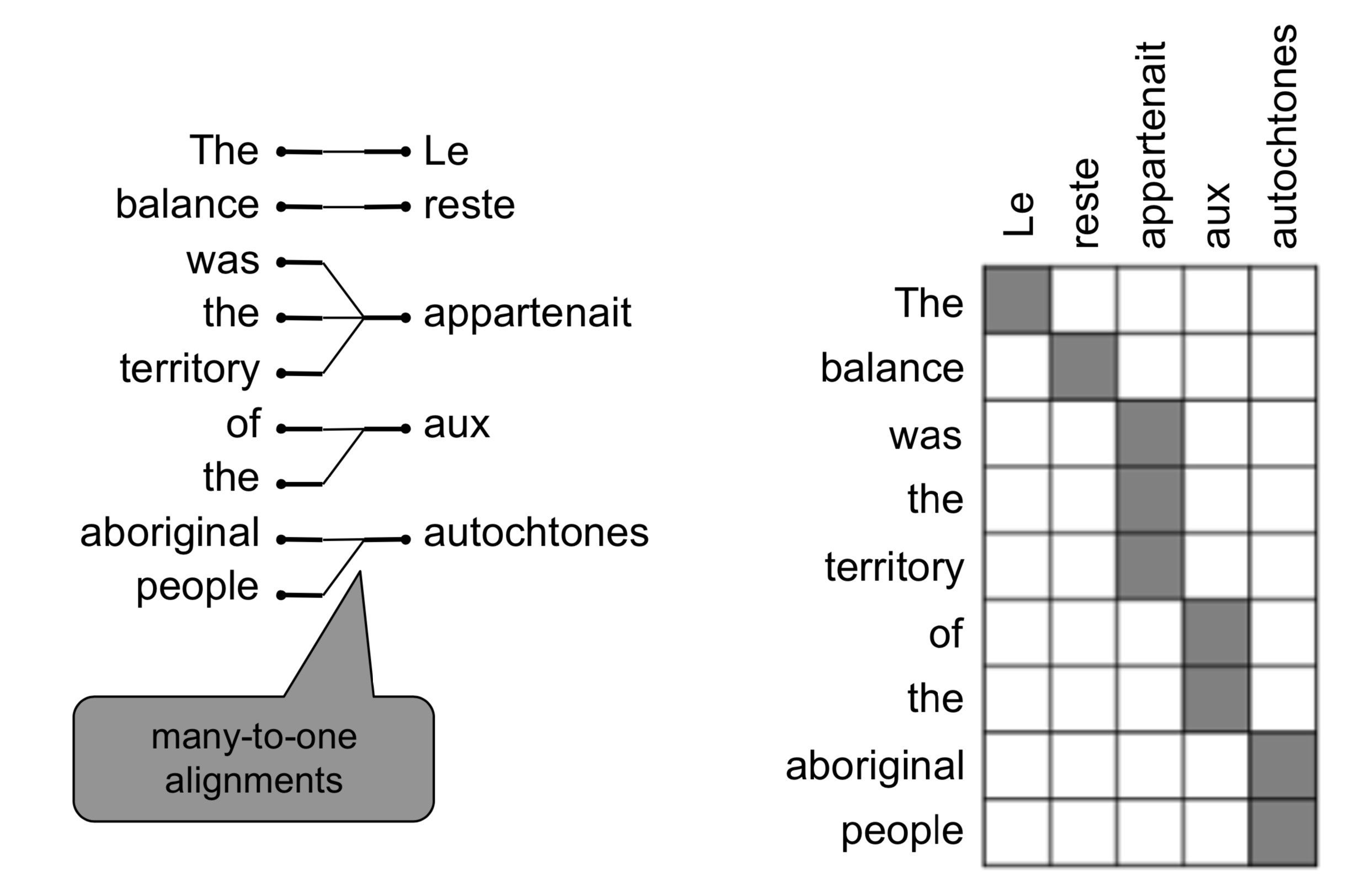
Alignment is the correspondence between particular words in the translated sentence pair.

* Typological differences between languages lead to complicated alignments! • Note: Some words have no counterpart



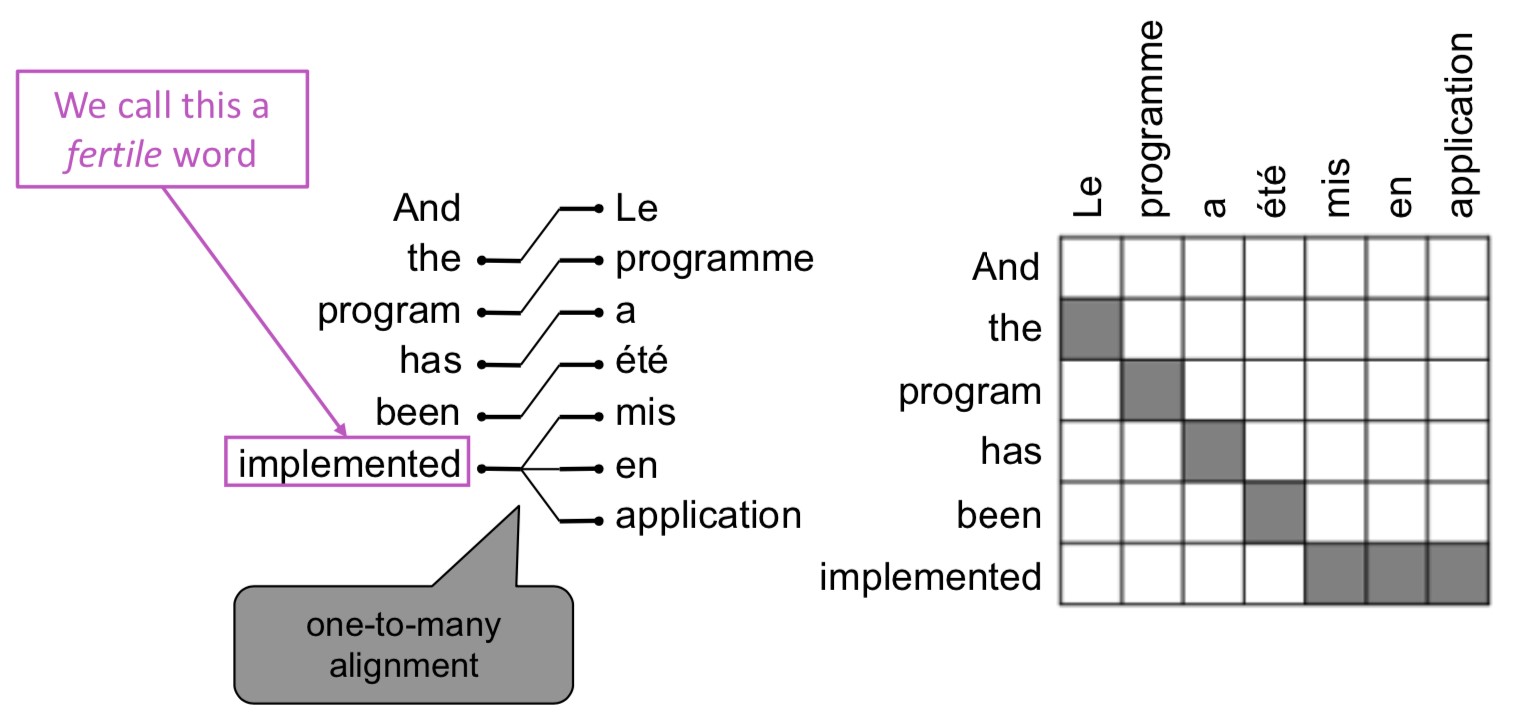
**Examples from:** “The Mathematics of Statistical Machine Translation: Parameter Estimation", Brown et al, 1993. [http://www.aclweb.org/ anthology/J93-2003](http://www.aclweb.org/anthology/J93-2003)

Alignment can be many-to-one



**Examples from:** “The Mathematics of Statistical Machine Translation: Parameter Estimation", Brown et al, 1993. [http://www.aclweb.org/ anthology/J93-2003](http://www.aclweb.org/anthology/J93-2003)

Alignment can be one-to-many



**Examples from:** “The Mathematics of Statistical Machine Translation: Parameter Estimation", Brown et al, 1993. [http://www.aclweb.org/ anthology/J93-2003](http://www.aclweb.org/anthology/J93-2003)

Some words are very fertile!

he hit me with a pie

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

il a m’ entarté

il

a

m’

entarté

he

hit

me

with

a

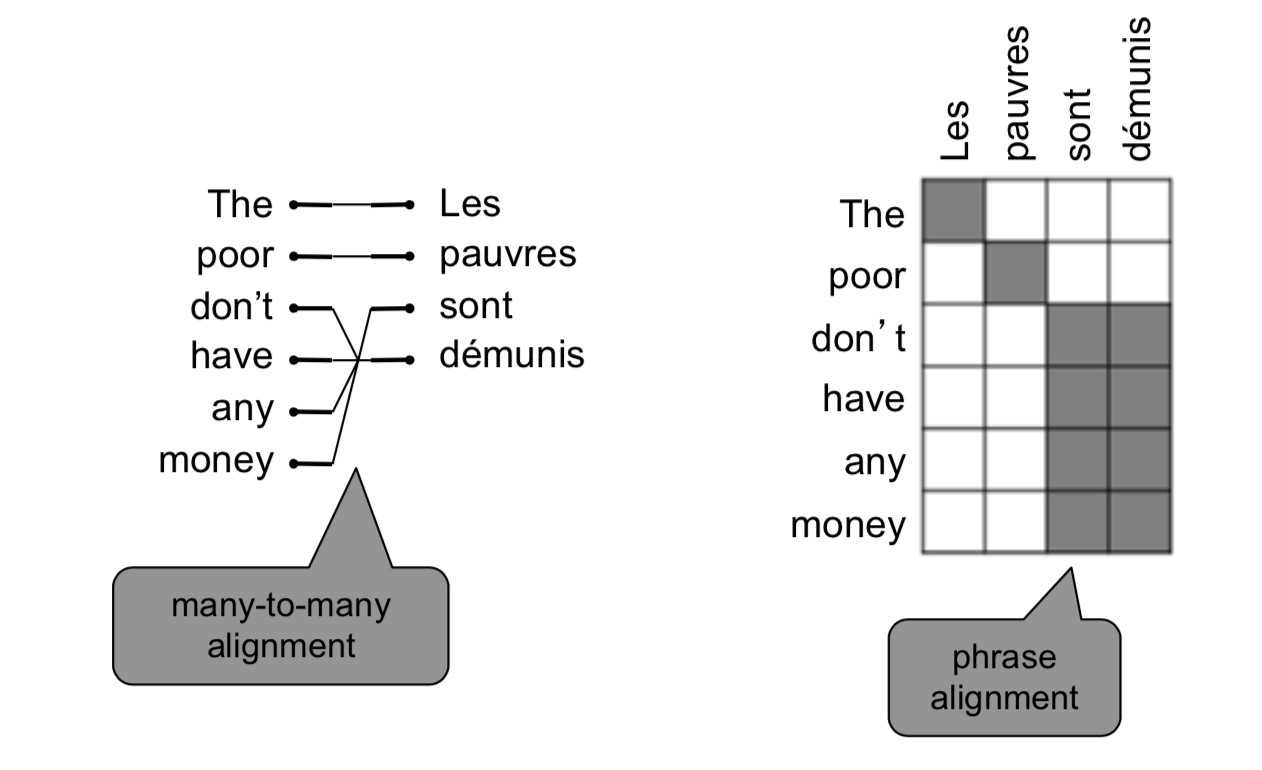
pie

This word has no single-

word equivalent in

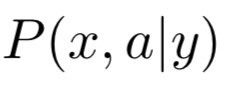
English

Alignment can be many-to-many (phrase-level)



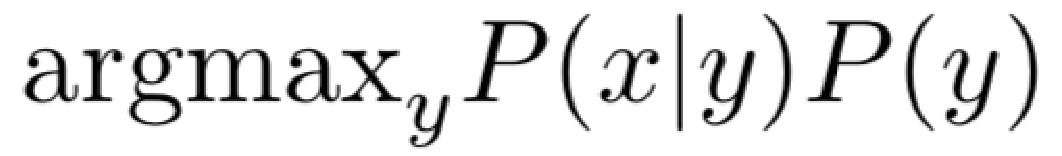
**Examples from:** “The Mathematics of Statistical Machine Translation: Parameter Estimation", Brown et al, 1993. [http://www.aclweb.org/ anthology/J93-2003](http://www.aclweb.org/anthology/J93-2003)

# Learning alignment for SMT

* We learn as a combination of many factors, including:
* Probability of particular words aligning (also depends on position in sent)
* Probability of particular words having particular fertility (number of corresponding words)
* etc.
* Alignments *a* are **latent variables**: They aren’t explicitly specified in the data!
* Require the use of special learning aglos (like ExpectationMaximization) for learning the parameters of distributions with latent variables (CS 228)

# Decoding for SMT

Language Model **Question:**

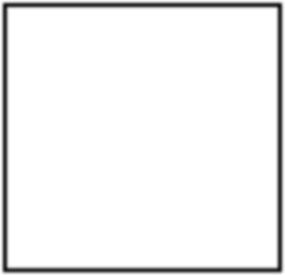
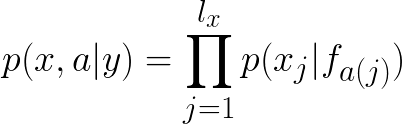
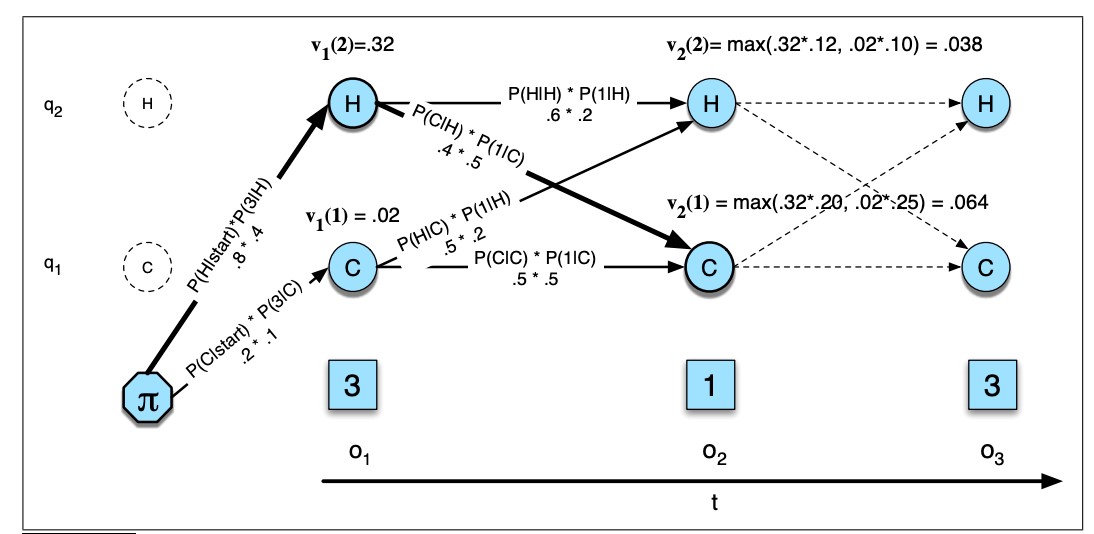


How to compute Translation this argmax? Model

* We could enumerate every possible *y* and calculate the probability? → Too expensive!
* Answer: Impose strong independence assumptions in model, use dynamic programming for globally optimal solutions (e.g. Viterbi algorithm).
* This process is called *decoding*

## Viterbi: Decoding with Dynamic Programming

• Impose strong independence assumptions in model:



**Source:** “Speech and Language Processing", Chapter A, Jurafsky and Martin, 2019.  

# 1990s-2010s: Statistical Machine Translation

* SMT was a huge research field
* The best systems were extremely complex
* Hundreds of important details we haven’t mentioned here
* Systems had many separately-designed subcomponents
* Lots of feature engineering
* Need to design features to capture particular language phenomena
* Require compiling and maintaining extra resources
* Like tables of equivalent phrases
* Lots of human effort to maintain
* Repeated effort for each language pair!

**Section 2: Neural Machine Translation 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-tosequence (aka seq2seq) and it involves *two* RNNs.

# Neural Machine Translation (NMT)

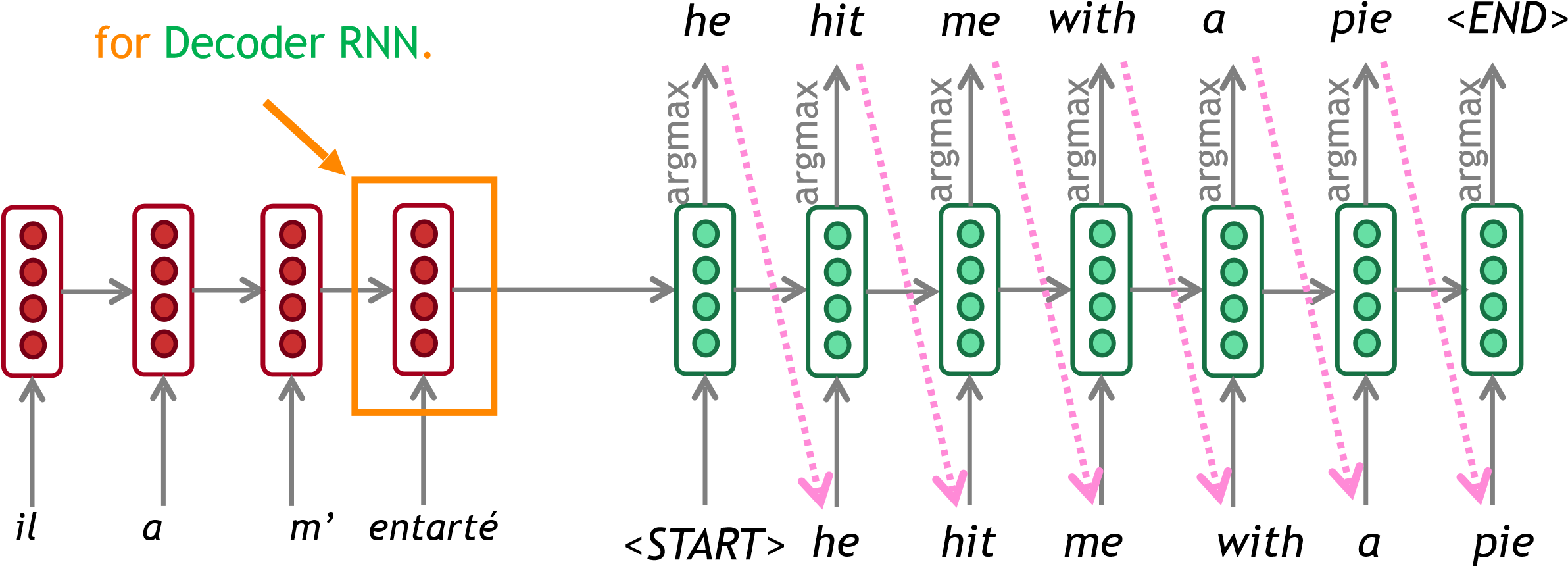
The sequence-to-sequence model

Target sentence (output)

Encoding of the source sentence.  Provides initial hidden state

Decoder RNN

Encoder RNN



|  |
| --- |
| Decoder RNN is a Language Model that generates target sentence, *conditioned on encoding*. |
| Note: This diagram shows **test time** behavior: decoder output is fed in as next step’s input |

|  |
| --- |
| Encoder RNN produces an encoding of the source sentence. |

Source sentence (input)

**Sequence-to-sequence is versatile!**

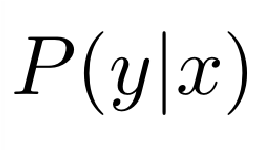
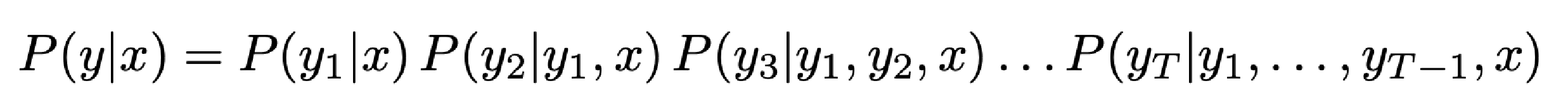
* Sequence-to-sequence is useful for *more than just MT*
* Many NLP tasks can be phrased as sequence-to-sequence:
* Summarization (long text → short text)
* Dialogue (previous utterances → next utterance)
* Parsing (input text → output parse as sequence)
* Code generation (natural language → Python code)

# Neural Machine Translation (NMT)

* The sequence-to-sequence model is an example of a  **Conditional Language Model**.
* **Language Model** because the decoder is predicting the  next word of the target sentence *y*
* **Conditional** because its predictions are *also* conditioned on the source sentence *x*

•

NMT directly calculates :



Probability of next target word, given

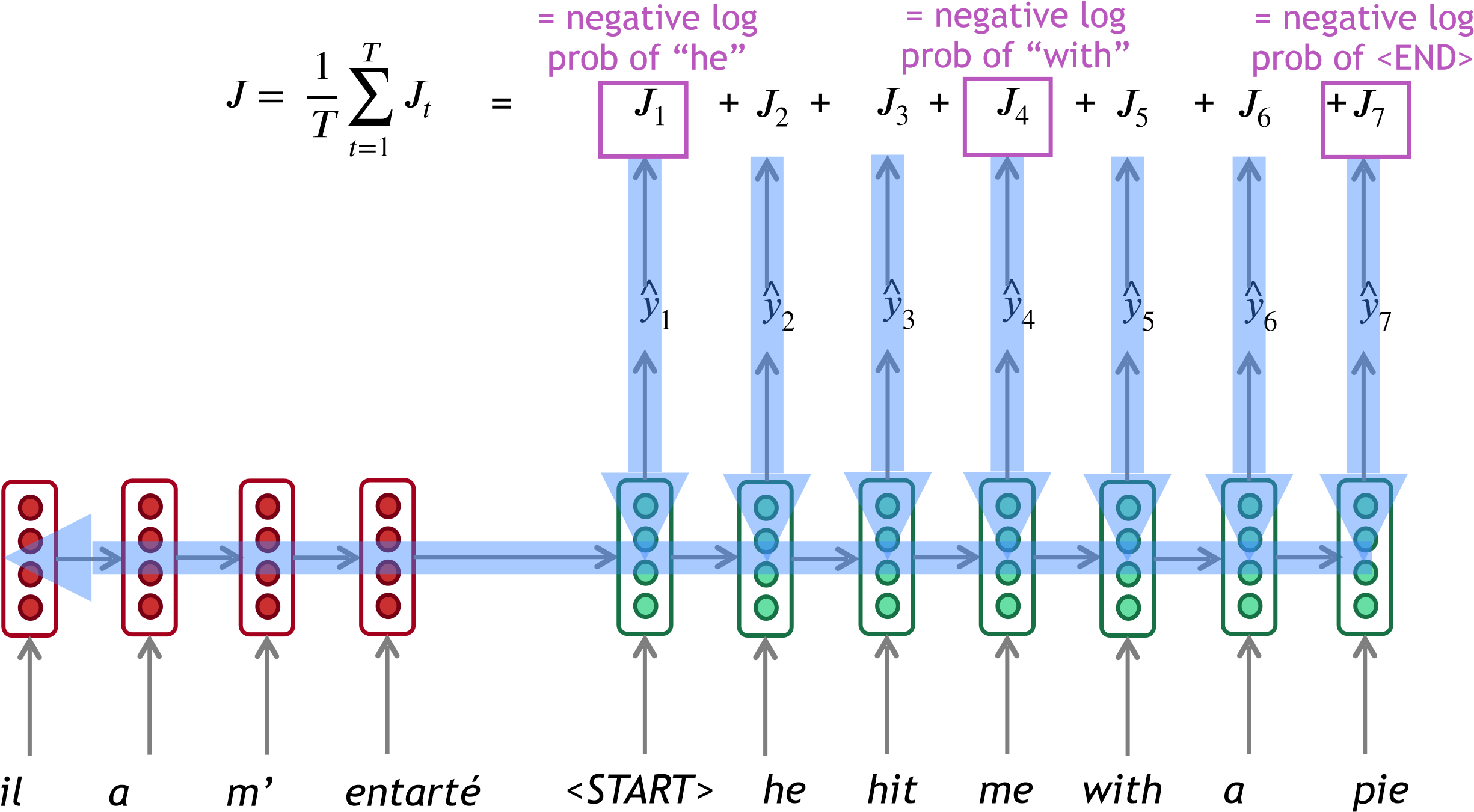
target words so far and source

sentence *x*

* **Question**: How to train a NMT system?
* **Answer**: Get a big parallel corpus…

## Training a Neural Machine Translation system

Encoder RNN



Decoder RNN

Source sentence (from corpus) Target sentence (from corpus)

Seq2seq is optimized as a **single system.**

Backpropagation operates “*end-to-end”*.

# Greedy decoding

* We saw how to generate (or “decode”) the target sentence by taking argmax on each step of the decoder

*he hit me with a pie <END>*

argmax

argmax

argmax

argmax

argmax

argmax

argmax

*<START> he hit me with a*

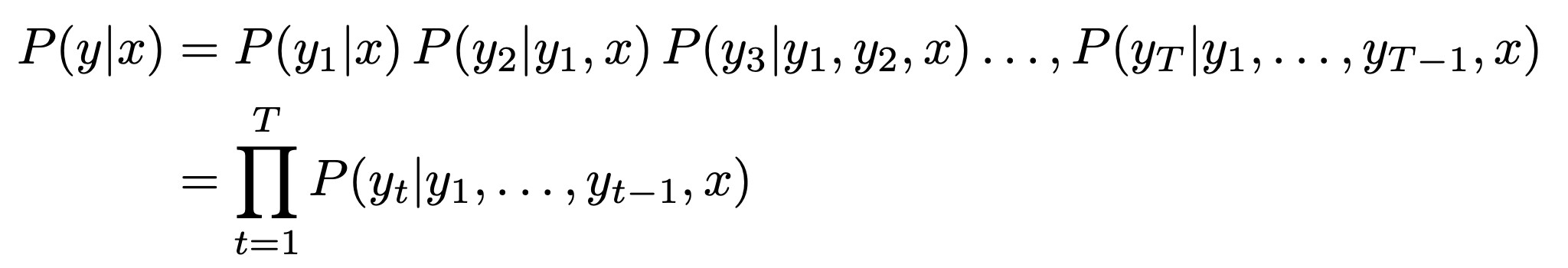
* This is greedy decoding (take most probable word on each *pie* step)
* **Problems with this method?**

# Problems with 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 to fix this?

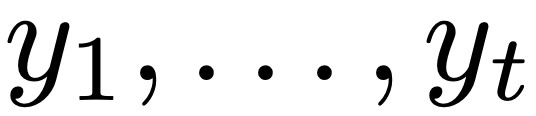
# Exhaustive search decoding

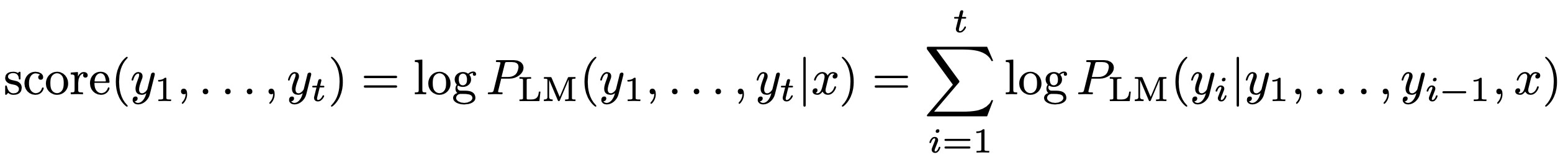
* Ideally we want to find a (length *T*) translation *y* that maximizes



* We could try computing all possible sequences *y*
* This means that on each step *t* of the decoder, we’re tracking Vt possible partial translations, where *V* is vocab size
* This O(VT) complexity is far too expensive!

# Beam search decoding

* Core idea: On each step of decoder, keep track of the *k* most probable partial translations (which we call *hypotheses*)
* *k* is the beam size (in practice around 5 to 10)
* A hypothesis has a score which is its log probability:



* Scores are all negative, and higher score is better
* We search for high-scoring hypotheses, tracking top *k* on each step
* Beam search is not guaranteed to find optimal solution
* But much more efficient than exhaustive search!

*<START>*

Calculate prob  dist of next word -0.7 = log PLM(*he*|*<START>*)

*<*

*START*

*>*

*he*

*I*

-0.9= log PLM(*I*|*<START>*)

Take top *k* words  and compute scores -1.7 = log PLM(*hit*|*<START> he*) + -0.7

-2.9= log PLM(*struck*|*<START> he*) + -0.7

-1.6= log PLM(*was*|*<START> I*) + -0.9

*hit*

*struck*

*was*

*got*

*<*

*START*

*>*

*he*

*I*

-0.7

-0.9

-1.8

= log PLM(*got*|*<START> I*) + -0.9

For each of the *k* hypotheses, find  top *k* next words and calculate scores Of these *k*2 hypotheses, just keep *k* with highest scores -2.8 = log PLM(*a*|*<START> he hit*) + -1.7

*hit*

*struck*

*was*

*got*

*<*

*START*

*>*

*he*

*I*

-0.7

-0.9

-1.6

-1.8

-1.7

-2.9

= log PLM(*me*|*<START> he hit*) + -1.7

-2.9 = log PLM(*hit*|*<START> I was*) + -1.6

*hit*

*struck*

*was*

*got*

*a*

*me*

*hit*

*struck*

*<*

*START*

*>*

*he*

*I*

-0.7

-0.9

-1.6

-1.8

-1.7

-2.9

-2.5

-3.8= log PLM(*struck*|*<START> I was*) + -1.6

For each of the *k* hypotheses, find  top *k* next words and calculate scores Of these *k*2 hypotheses, just keep *k* with highest scores

*hit*

*struck*

*was*

*got*

*a*

*me*

*hit*

*struck*

*<*

*>*

*START*

*he*

*I*

-0.7

-0.9

-1.6

-1.8

-1.7

-2.9

-2.5

-2.8

-3.8

-2.9

For each of the *k* hypotheses, find  top *k* next words and calculate scores just keep *k* with highest scores

*hit*

*struck*

*was*

*got*

*a*

*me*

*hit*

*struck*

*tart*

*pie*

*with*

*on*

*>*

*START*

*<*

*he*

*I*

-0.7

-0.9

-1.6

-1.7

-2.9

-2.5

-2.8

-3.8

-2.9

-3.5

-3.3

-4.0

-3.4

*hit*

*struck*

*was*

*got*

*a*

*me*

*hit*

*struck*

*tart*

*pie*

*with*

*on*

*>*

*START*

*<*

*he*

*I*

-0.7

-0.9

-1.6

-1.7

-2.9

-2.5

-2.8

-3.8

-2.9

-3.5

-3.3

-4.0

-3.4

Of these

*k*

2

hypotheses,

*in*

*hit*

*struck*

*was*

*got*

*a*

*me*

*hit*

*struck*

*tart*

*pie*

*with*

*on*

*with*

*a*

*one*

*START*

*>*

*<*

*he*

*I*

-0.7

-0.9

-1.6

-1.7

-2.9

-2.5

-2.8

-3.8

-2.9

-3.5

-3.3

-4.0

-3.4

-3.7

-4.3

-4.5

-4.8

For each of the

*k*

hypotheses, find

top *k* next words and calculate scores

*in*

*hit*

*struck*

*was*

*got*

*a*

*me*

*hit*

*struck*

*tart*

*pie*

*with*

*on*

*with*

*a*

*one*

*START*

*>*

*<*

*he*

*I*

-0.7

-0.9

-1.6

-1.7

-2.9

-2.5

-2.8

-3.8

-2.9

-3.5

-3.3

-4.0

-3.4

-3.7

-4.3

-4.5

-4.8

Of these

*k*

2

hypotheses,

just keep *k* with highest scores

*in*

*hit*

*struck*

*was*

*got*

*a*

*me*

*hit*

*struck*

*tart*

*pie*

*with*

*on*

*with*

*a*

*one*

*pie*

*tart*

*pie*

*tart*

*>*

*START*

*<*

*he*

*I*

-0.7

-0.9

-1.6

-1.7

-2.9

-2.5

-2.8

-3.8

-2.9

-3.5

-3.3

-4.0

-3.4

-3.7

-4.3

-4.5

-4.8

-4.3

-4.6

-5.0

-5.3

For each of the

*k*

hypotheses, find

top *k* next words and calculate scores

*in*

*hit*

*struck*

*was*

*got*

*a*

*me*

*hit*

*struck*

*tart*

*pie*

*with*

*on*

*with*

*a*

*one*

*pie*

*tart*

*pie*

*tart*

*START*

*>*

*<*

*he*

*I*

-0.7

-0.9

-1.6

-1.7

-2.9

-2.5

-2.8

-3.8

-2.9

-3.5

-3.3

-4.0

-3.4

-3.7

-4.3

-4.5

-4.8

-4.3

-4.6

-5.0

-5.3

This is the top-scoring hypothesis!

*in*

*hit*

*struck*

*was*

*got*

*a*

*me*

*hit*

*struck*

*tart*

*pie*

*with*

*on*

*with*

*a*

*one*

*pie*

*tart*

*pie*

*tart*

*START*

*>*

*<*

*he*

*I*

-0.7

-0.9

-1.6

-1.7

-2.9

-2.5

-2.8

-3.8

-2.9

-3.5

-3.3

-4.0

-3.4

-3.7

-4.3

-4.5

-4.8

-4.3

-4.6

-5.0

-5.3

Backtrack to obtain the full hypothesis

# Beam search decoding: stopping criterion

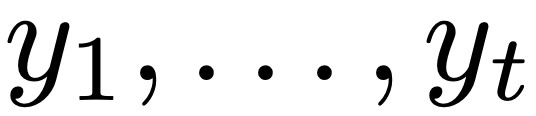
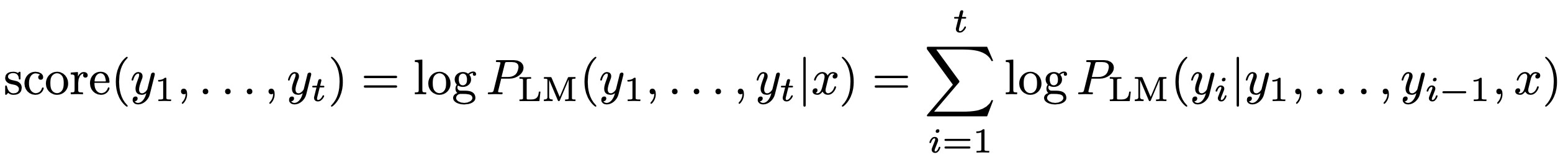
* In greedy decoding, usually we decode until the model produces a <END> token
* For example: *<START> he hit me with a pie <END>*
* In beam search decoding, different hypotheses may produce <END> tokens on different timesteps
* When a hypothesis produces <END>, that hypothesis is complete.
* Place it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

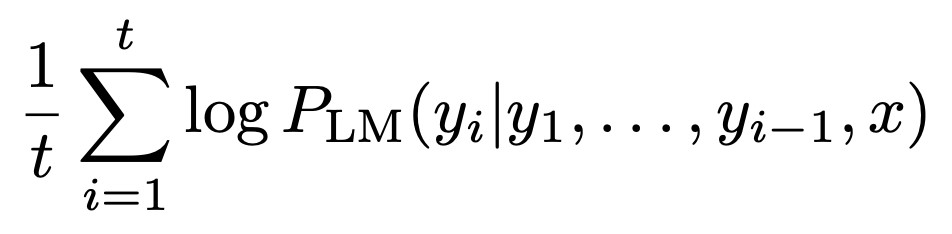
# Beam search decoding: finishing up

* We have our list of completed hypotheses.
* How to select top one with highest score?

•

Each hypothesis on our list has a score



* Problem with this: longer hypotheses have lower scores
* Fix: Normalize by length. Use this to select top one instead: 

**Advantages of NMT**

Compared to SMT, NMT has many advantages:

* Better performance
* More fluent
* Better use of context
* Better use of phrase similarities
* A single neural network to be optimized end-to-end
* No subcomponents to be individually optimized
* Requires much less human engineering effort
* No feature engineering
* Same method for all language pairs

**Disadvantages of NMT?**

Compared to SMT:

* NMT is less interpretable
* Hard to debug
* NMT is difficult to control
* For example, can’t easily specify rules or guidelines for translation
* Safety concerns!

**How do we evaluate Machine Translation?**

|  |
| --- |
| You’ll see BLEU in detail in Assignment 4! |

**BLEU** (**B**i**l**ingual **E**valuation **U**nderstudy)

* BLEU compares the machine-written translation to one or several human-written translation(s), and computes a similarity score based on:
* *n*-gram precision (usually for 1, 2, 3 and 4-grams)
* Plus a penalty for too-short system translations
* BLEU is useful but imperfect
* There are many valid ways to translate a sentence
* So a good translation can get a poor BLEU score because it has low *n*-gram overlap with the human translation ☹

**Source:** ”BLEU: a Method for Automatic Evaluation of Machine Translation", Papineni et al, 2002. <http://aclweb.org/anthology/P02-1040>

# MT progress over time

[Edinburgh En-De WMT newstest2013 Cased BLEU; NMT 2015 from U. Montréal]

0

6.8

13.5

20.3

27

2013

2014

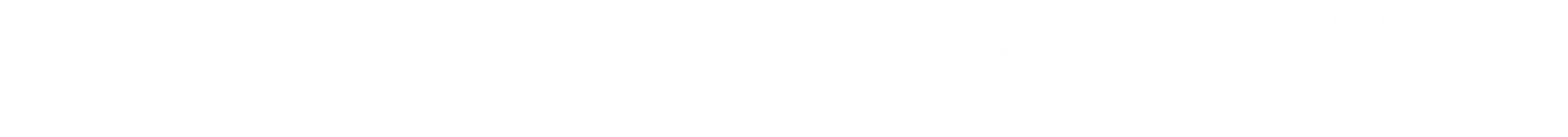
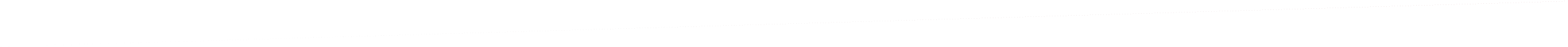
2015

2016

Phrase-based SMT

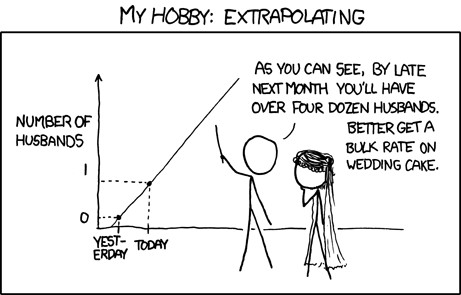
Syntax-based SMT

Neural MT



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

# MT progress over time



48

**NMT: the biggest success story of NLP Deep Learning**

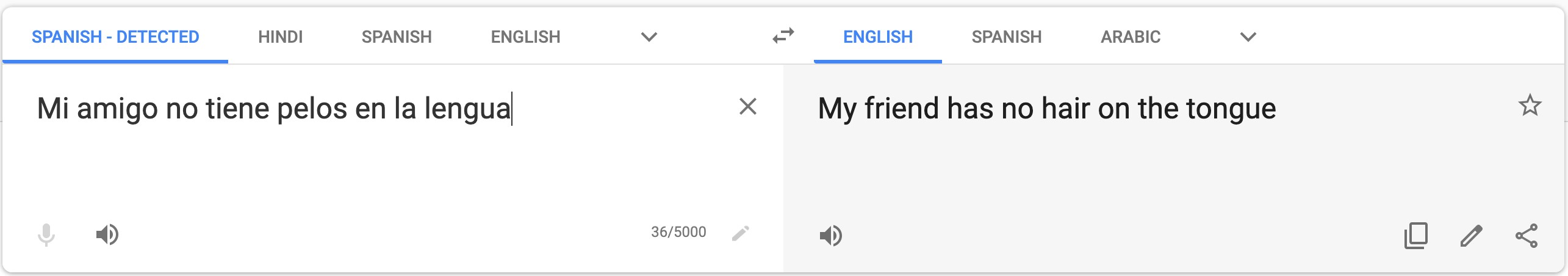
Neural Machine Translation went from a fringe research activity in **2014** to the leading standard method in **2016**

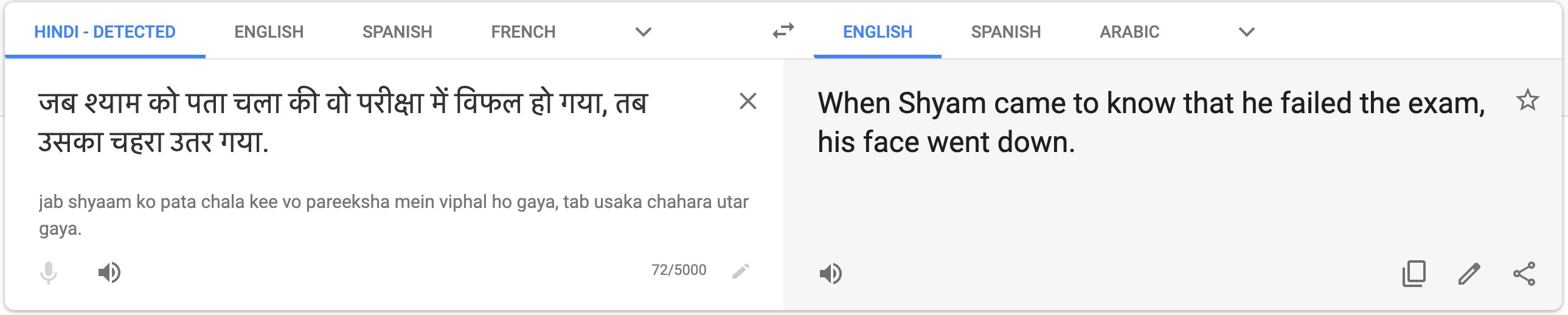
* **2014**: First seq2seq paper published
* **2016**: Google Translate switches from SMT to NMT
* This is amazing!
* **SMT** systems, built by hundreds of engineers over many years, outperformed by NMT systems trained by a handful of engineers in a few months
* **Nope!**
* Many difficulties remain:
* Out-of-vocabulary words
* Domain mismatch between train and test data
* Maintaining context over longer text
* Low-resource language pairs

**Further reading:** “*Has AI surpassed humans at translation? Not even close!”*

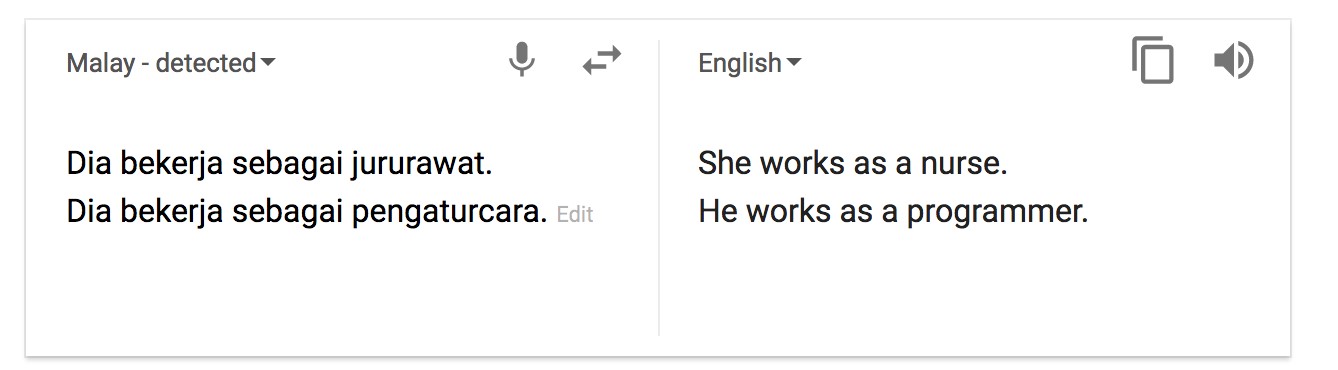
<https://www.skynettoday.com/editorials/state_of_nmt>

* **Nope!**
* Using common sense is still hard
* Idioms are difficult to translate





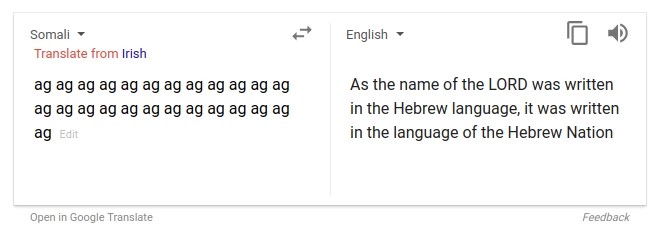
* **Nope!**
* NMT picks up biases in training data



Didn’t specify gender

**Source:** [https://hackernoon.com/bias-sexist-or-this-is-the-way-it-should-bece1f7c8c683c](https://hackernoon.com/bias-sexist-or-this-is-the-way-it-should-be-ce1f7c8c683c)

* Nope!
* Uninterpretable systems do strange things



**Picture source**: [https://www.vice.com/en\_uk/article/j5npeg/why-is-google-translate-spitting-out-sinisterreligious-prophecies](https://www.vice.com/en_uk/article/j5npeg/why-is-google-translate-spitting-out-sinister-religious-prophecies)

**Explanation**: <https://www.skynettoday.com/briefs/google-nmt-prophecies>

# NMT research continues

NMT is the **flagship task** for NLP Deep Learning

* NMT research has pioneered many of the recent innovations of NLP Deep Learning
* In **2019**: NMT research continues to thrive
* Researchers have found *many, many* improvements to the

“vanilla” seq2seq NMT system we’ve presented today

* But one improvement is so integral that it is the new vanilla…

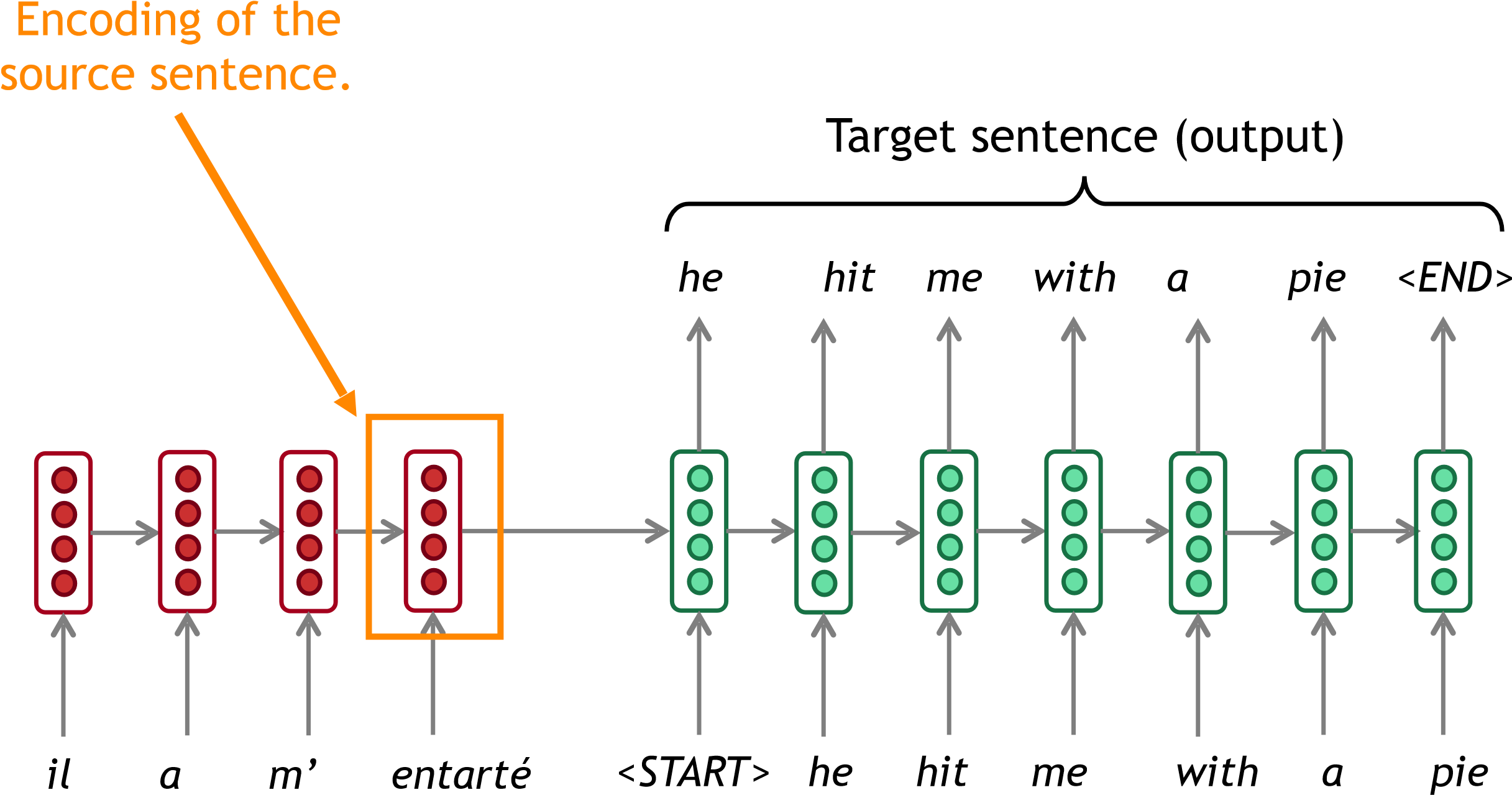
# ATTENTION

## Section 3: Attention

### Sequence-to-sequence: the bottleneck problem

Decoder RNN

Encoder RNN



Source sentence (input)

**Problems with this architecture?**

|  |  |
| --- | --- |
| Encoding of the  source sentence.  This needs to capture *all information* about the source sentence.  Information bottleneck! | Target sentence (output)  *he hit me with a pie* |

### Sequence-to-sequence: the bottleneck problem

Decoder RNN

Encoder RNN

*il a m’ <START> he hit me with a pie*

*entarté*

Source sentence (input)

## Attention

* **Attention** provides a solution to the bottleneck problem.
* Core idea: on each step of the decoder, use *direct connection to the encoder* to *focus on a particular part* of the source sequence



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

Encoder

RNN

Attention

scores

dot product

Decoder RNN

Encoder

RNN

*il a m’ entarté*

Attention

scores

dot product

Decoder RNN

Encoder

RNN

Attention

scores

dot product

Decoder RNN

Encoder

RNN

Attention

scores

dot product

Decoder RNN

Encoder

RNN

Attention

scores

On this decoder timestep,

we’re mostly focusing on the

first encoder hidden state

(

*”he”*

)

Attention

distribution

Take softmax to turn the

scores into a probability

distribution

Decoder RNN

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

Encoder

RNN

Attention

distribution

Attention

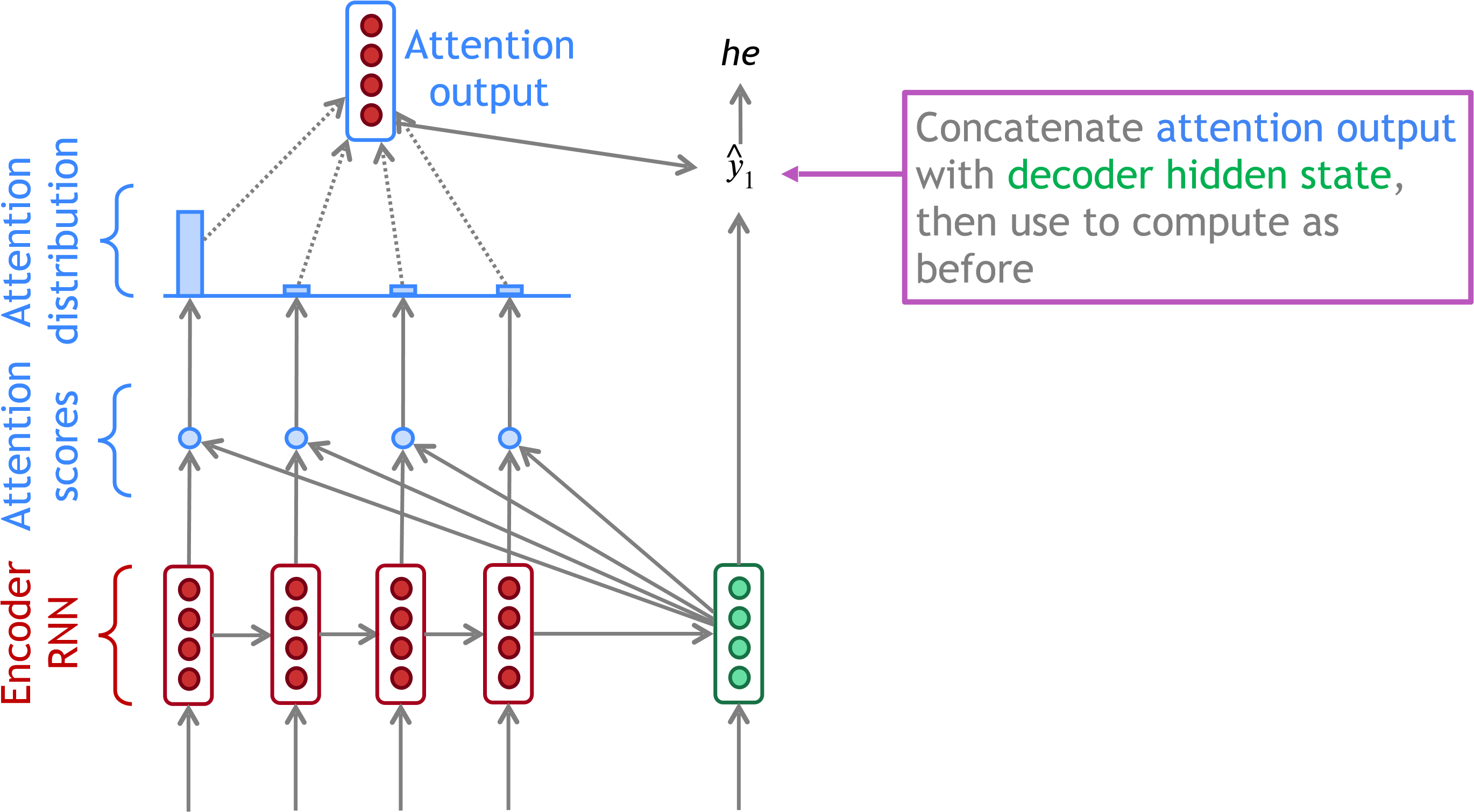
scores

Attention

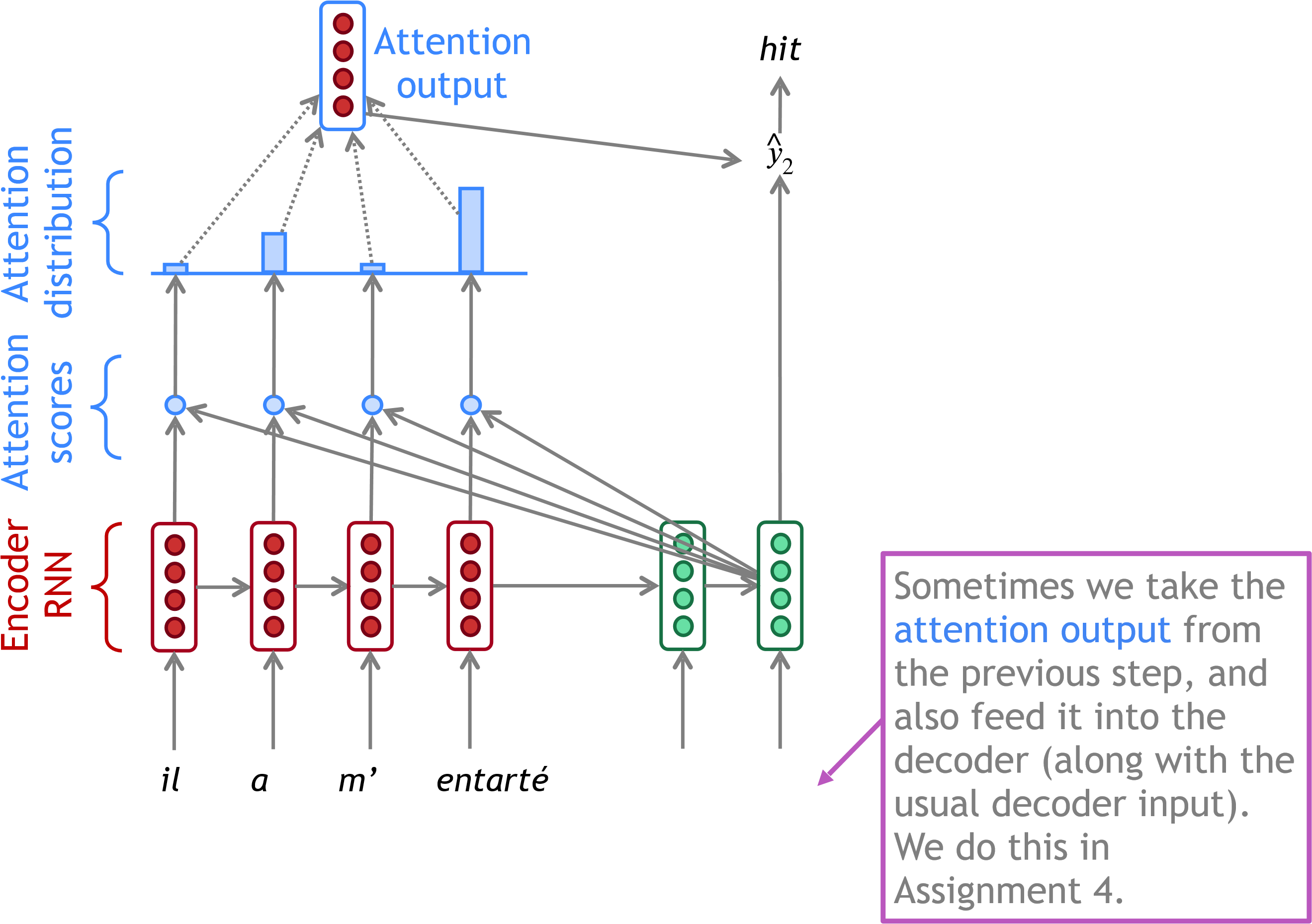
output

Decoder RNN

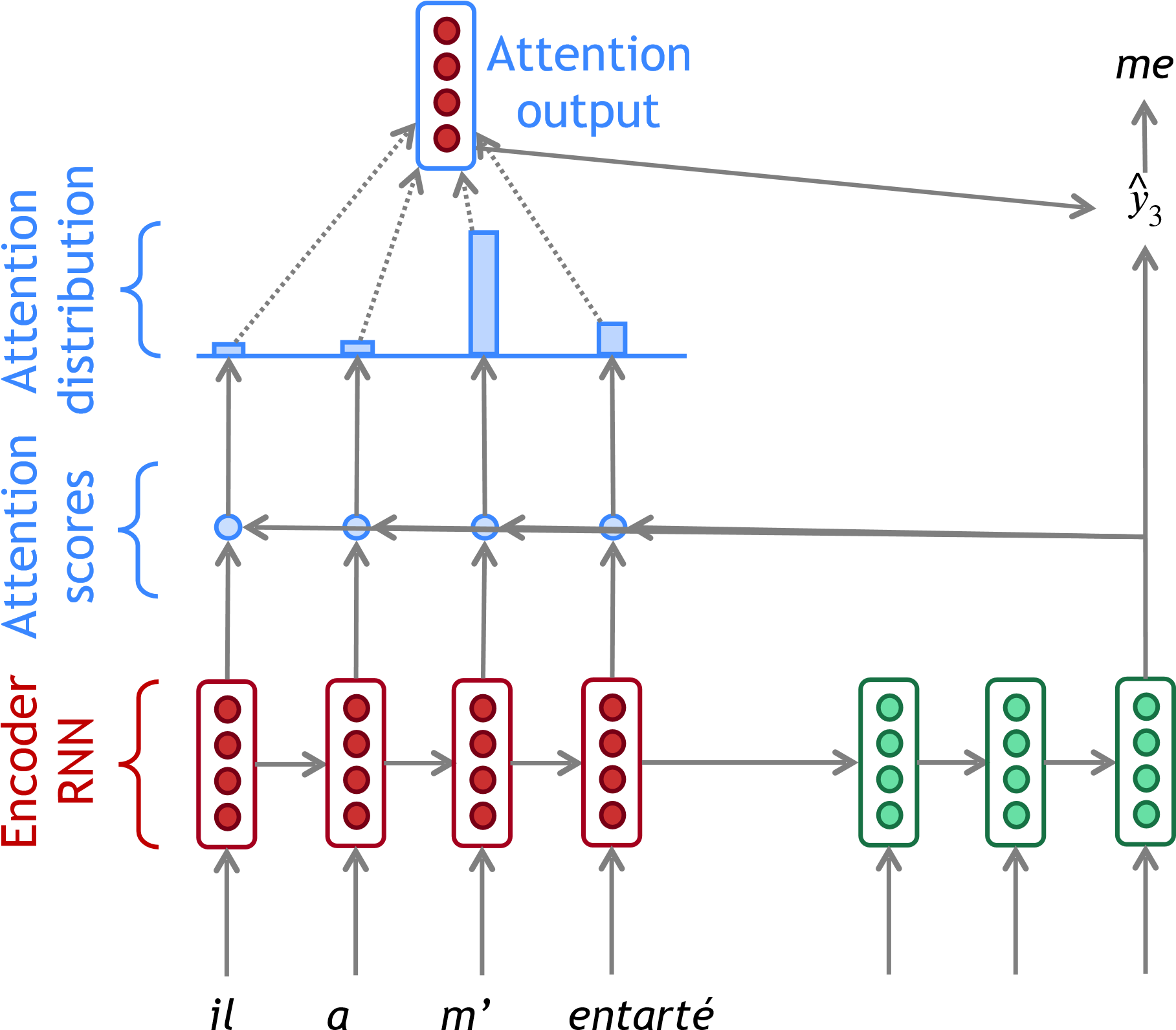
The attention output mostly contains information from the hidden states that received high attention.



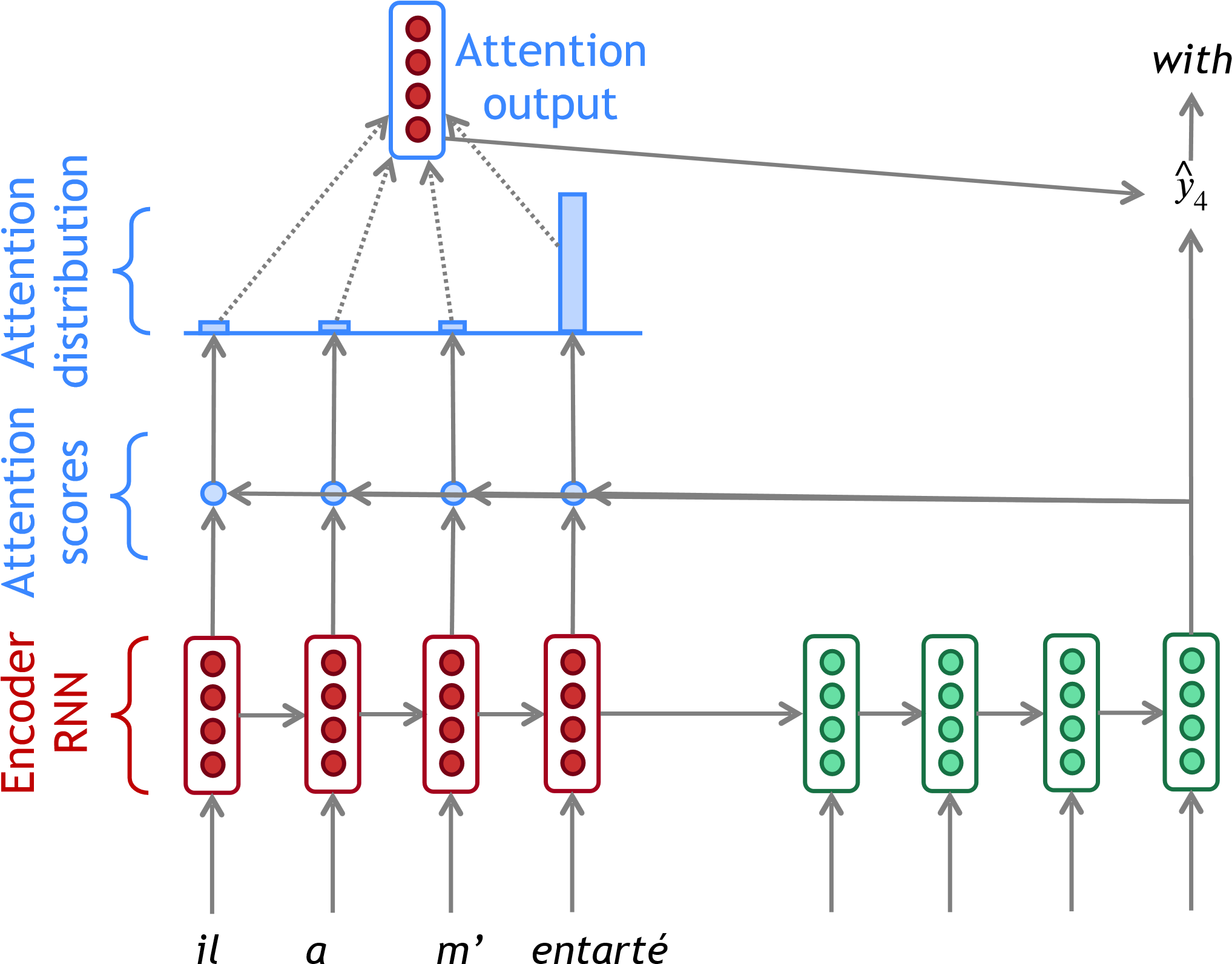
Decoder RNN



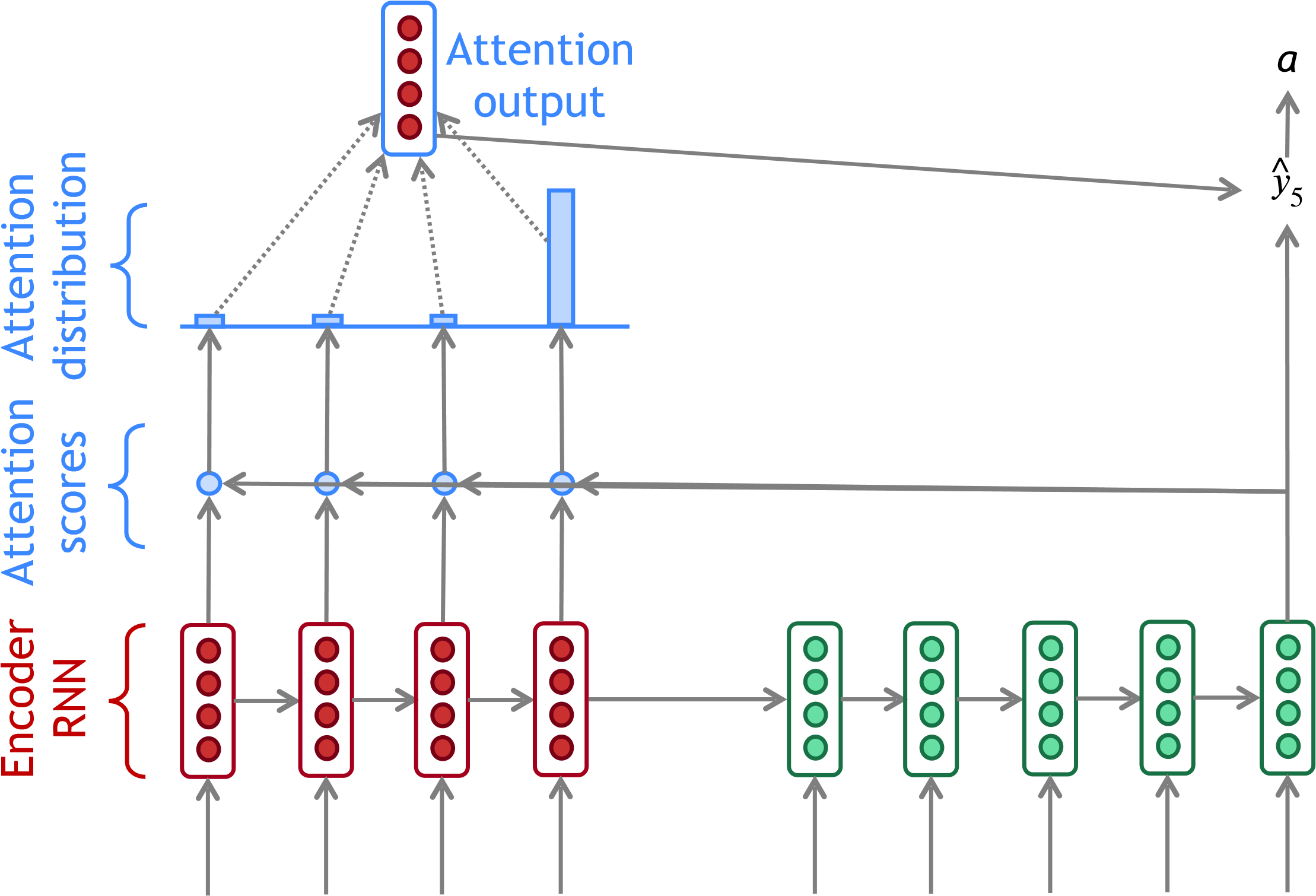
Decoder RNN



Decoder RNN

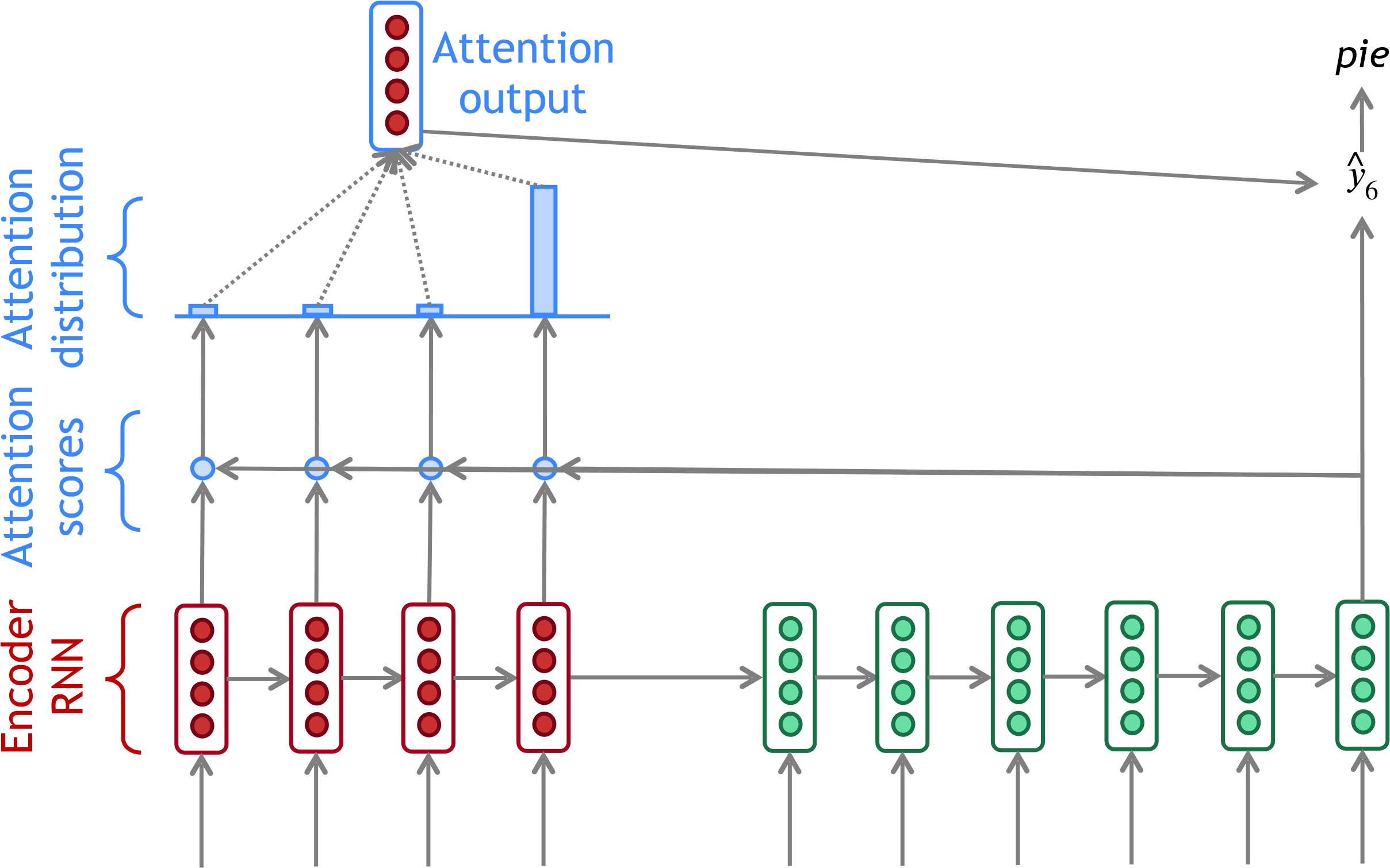


Decoder RNN



*il a m’ entarté with*

Decoder RNN



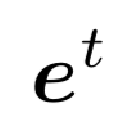
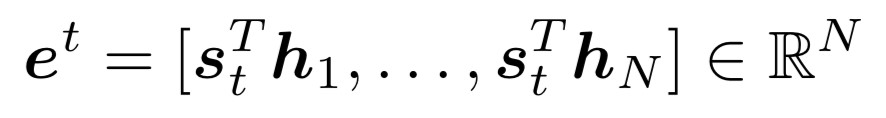
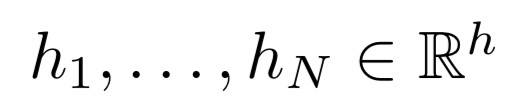
*il a m’ entarté with a*

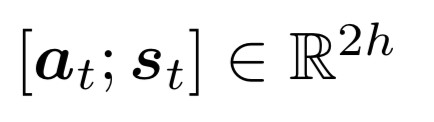
Decoder RNN

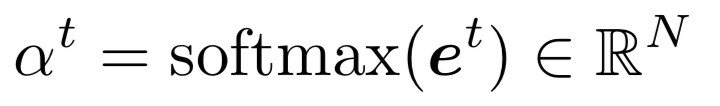
## Attention: in equations

* We have encoder hidden states
* On timestep *t*, we have decoder hidden state
* We get the attention scores for this step:

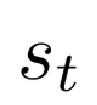
* We take softmax to get the attention distribution for this step

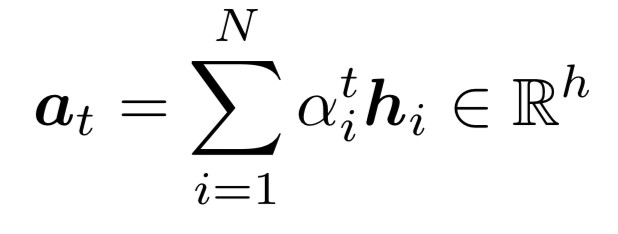


(this is a probability distribution and sums to 1)



* We use to take a weighted sum of the encoder hidden states to get the attention output

* Finally we concatenate the attention output with the decoder hidden state and proceed as in the non-attention seq2seq model



## 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

he hit me wit a pie

* By inspecting attention distribution, we can see   h what the decoder was focusing on il
* We get (soft) alignment for free! a
* This is cool because we never explicitly trained  m’ an alignment system entarté
* The network just learned alignment by itself

### Attention is a *general* Deep Learning technique

* We’ve seen that attention is a great way to improve the sequence-to-sequence model for Machine Translation.
* However: You can use attention in many architectures

(not just seq2seq) and many tasks (not just MT)

|  |  |
| --- | --- |
| • | **More general definition of attention**:  • Given a set of vector *values*, and a vector *query*, **attention** is a technique to compute a weighted sum of the values, dependent on the query. |

* We sometimes say that the query *attends to* the values.
* For example, in the seq2seq + attention model, each decoder hidden state (query) *attends to* all the encoder hidden states (values).

### Attention is a *general* Deep Learning technique

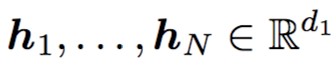
**More general definition of attention**:

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

**Intuition**:

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

## There are *several* attention variants

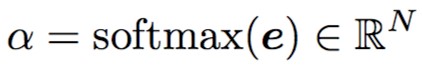
• We have some *values* and a *query*  • Attention always involves:

There are

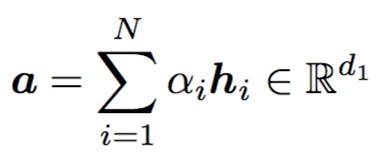
1. Computing the *attention scores* multiple ways



1. Taking softmax to get *attention distribution* ⍺:  to do this



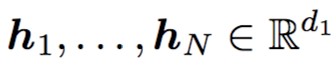
1. Using attention distribution to take weighted sum of values:



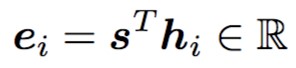
thus obtaining the *attention output* ***a*** (sometimes called the *context vector*)

|  |
| --- |
| You’ll think about the relative advantages/ disadvantages of these in Assignment 4! |

## Attention variants

There are several ways you can compute from and :

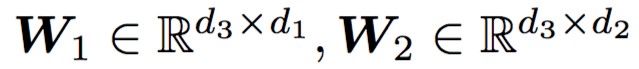
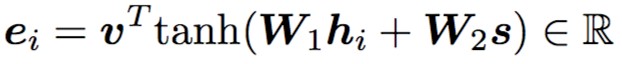
* Basic dot-product attention:



* Note: this assumes
* This is the version we saw earlier
* Multiplicative attention:
* Where is a weight matrix



* Additive attention:
* Where are weight matrices and is a weight vector.



* *d*3 (the attention dimensionality) is a hyperparameter

**More information:**

“Deep Learning for NLP Best Practices”, Ruder, 2017. [http://ruder.io/deep-learning-nlp-best-practices/ index.html#attention](http://ruder.io/deep-learning-nlp-best-practices/index.html#attention)

“Massive Exploration of Neural Machine Translation Architectures”, Britz et al, 2017, [https://arxiv.org/pdf/](https://arxiv.org/pdf/1703.03906.pdf)

[1703.03906.pdf](https://arxiv.org/pdf/1703.03906.pdf)

## Summary of today’s lecture

* We learned some history of Machine Translation (MT)
* Since 2014, Neural MT rapidly  replaced intricate Statistical MT

* Sequence-to-sequence is the  architecture for NMT (uses 2 RNNs)
* Attention is a way to *focus on*  *particular parts* of the input
* Improves sequence-to-sequence a lot!