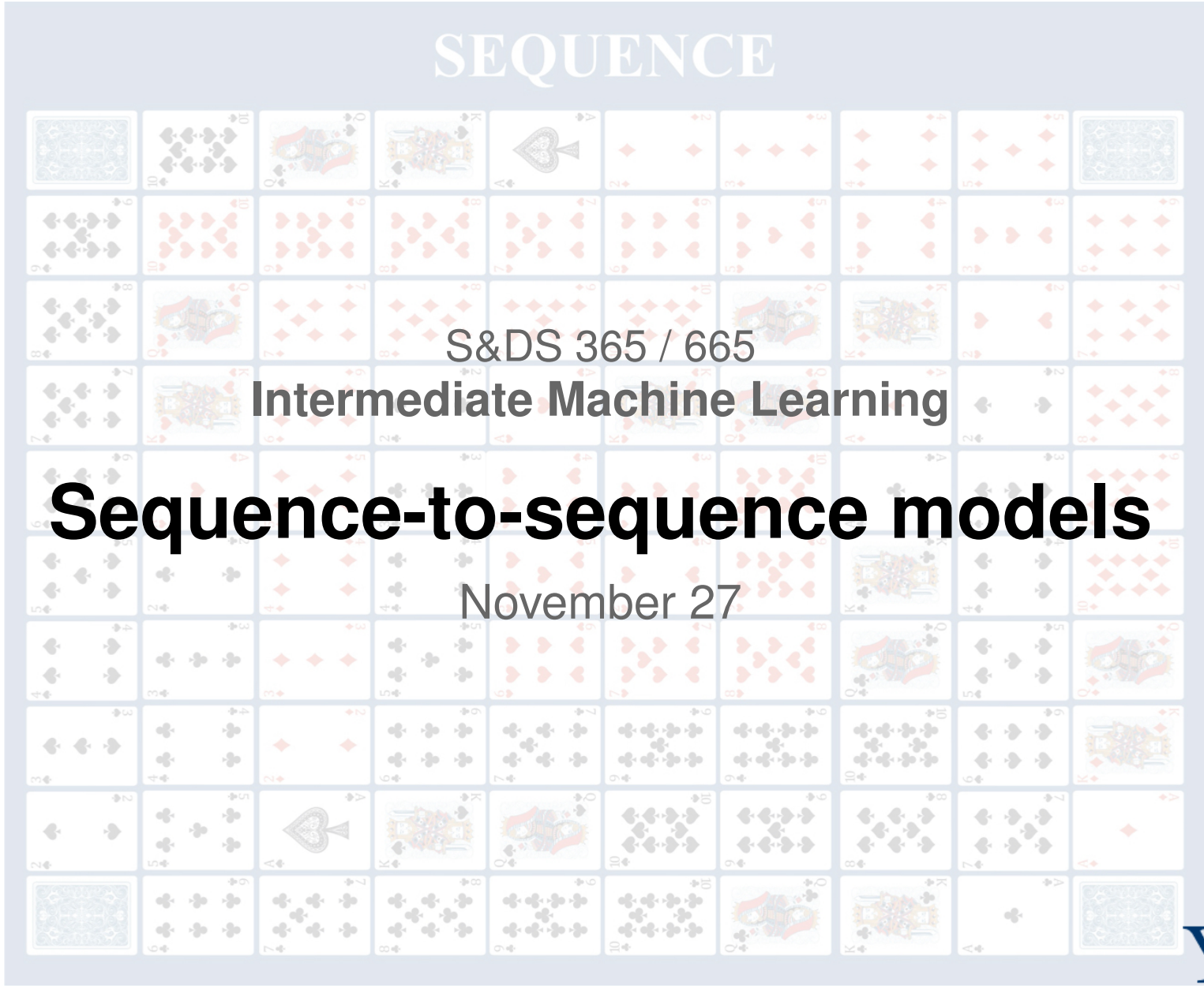


S&DS 365 / 665

Intermediate Machine Learning

Sequence-to-sequence models

November 27

Yale

While you were feasting...

TECH

OpenAI researchers warned board of AI breakthrough ahead of CEO ouster

PUBLISHED WED, NOV 22 2023•6:09 PM EST | UPDATED THU, NOV 23 2023•1:53 AM EST

REUTERS

WATCH LIVE

Sam Altman, chief executive officer of OpenAI, during a fireside chat organized by Softbank Ventures Asia in Seoul, South Korea, on Friday, June 9, 2023.

SeongJoon Cho | Bloomberg | Getty Images

While you were feasting...

The maker of ChatGPT had made progress on Q* (pronounced Q-Star), which some internally believe could be a breakthrough in the startup's search for superintelligence, also known as artificial general intelligence (AGI), one of the people told Reuters. OpenAI defines AGI as AI systems that are smarter than humans.

Given vast computing resources, the new model was able to solve certain mathematical problems, the person said on condition of anonymity because they were not authorized to speak on behalf of the company. Though only performing math on the level of grade-school students, acing such tests made researchers very optimistic about Q*'s future success, the source said. Reuters could not independently verify the capabilities of Q* claimed by the researchers.

---

https://www.cnbc.com/2023/11/22/sam-altmans-ouster-at-openai-precipitated-by-letter-to-board-about-ai-breakthrough-sources-tell-reuters.html

Mathematical reasoning

Question: What is $h(g(f(x)))$, where $f(x) = 2x + 3$, $g(x) = 7x - 4$, $h(x) = -5x - 8$?

Answer: $-70x - 165$.

- Parsing, grouping characters into entities
- Planning, identifying functions in correct order
- Sub-problems, applying operations like sum and product
- Working memory, storing intermediate values

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models, 2019

Mathematical reasoning

Question: `Solve -42*r + 27*c = -1167 and 130*r + 4*c = 372 for r.`
Answer: `4`
Question: `Calculate -841880142.544 + 411127.`
Answer: `-841469015.544`
Question: `Let x(g) = 9*g + 1. Let q(c) = 2*c + 1. Let f(i) = 3*i - 39. Let w(j) = q(x(j)). Calculate f(w(a)).`
Answer: `54*a - 30`
Question: `Let e(l) = l - 6. Is 2 a factor of both e(9) and 2?`
Answer: `False`
Question: `Let u(n) = -n**3 - n**2. Let e(c) = -2*c**3 + c. Let l(j) = -118*e(j) + 54*u(j). What is the derivative of l(a)?`
Answer: `546*a**2 - 108*a - 118`
Question: `Three letters picked without replacement from qqqkkklkqkkk. Give prob of sequence qql.`
Answer: `1/110`

Approached as sequence-to-sequence tasks.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models, 2019

For Today

Sequence models

- Memory circuits: GRUs (recap)
- Sequence-to-sequence models
- Attention mechanisms
- Next: Transformers

But first some reminders

- Quiz 5 (last!) on Wednesday, Nov 29 (RL, HMMs, RNNs, GRUs)
- Assn 5 out; due Dec. 6
- Final exam: Wednesday Dec 20, 9am in HQ L02
- Practice exams are posted
- Review sessions TBA

---

https://registrar.yale.edu/general-information/final-exams

Sequence models

- Generative process, any sequence (of words, characters, stock prices, nucleotides...) is assigned a probability

$$p(x_1, \ldots, x_n)$$

which can be factored as

$$p(x_1, \ldots, x_n) = p(x_1)p(x_2 \mid x_1) \ldots p(x_n \mid x_1, \ldots, x_{n-1})$$

Sequence generation

- Items generated one-by-one
- Output at time t chosen by sampling from a probability distribution
- State h_t encodes "features" of sequence $x_1, x_2, \ldots, x_t$
- We talked about state models: HMMs, KFs, and RNNs

Recurrent neural network

In an RNN, current output generated from a distributed state—a vector of neurons

- State is a deterministic, a nonlinear function of previous state and current input symbol.
- Dependence on earlier x_t's is encoded in the state
- Output generated stochastically by sampling from

$$\text{Softmax}(\beta^T h_t)$$

where $h_t = f(x_1, \ldots, x_{t-1})$ is the state vector.

“Vanilla” RNNs

In principle the state h_t can carry information from far in the past.

In practice, the gradients vanish (or explode) so this doesn’t really happen

We need other mechanisms to “remember” information from far away and use it to predict future words

“Vanilla” RNNs

State is updated according to

$$h_t = \tanh\left(W_{hh}h_{t-1} + W_{hx}x_t + b_h\right)$$

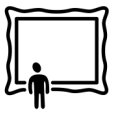
We modify this with two types of “neural circuits” for storing information to be used downstream

LSTMs and GRUs

Both LSTMs and GRUs have longer-range dependencies than vanilla RNNs.

We went through this in detail for GRUs, which are simpler, more efficient, and more commonly used

Gated recurrent units (GRUs)



High level idea:

- Learn when to update hidden state to “remember” important pieces of information
- Keep them in memory until they are used
- Reset or “forget” this information when no longer useful

GRUs

GRUs make use of “gates” denoted by Γ (Greek G for “Gate”)

$\Gamma = 1$: “the gate is open” and information flows through

$\Gamma = 0$: “the gate is closed” and information is blocked

GRUs

Two types of gates are used:

Γ^u: When open, information from long-term memory is propagated. When closed, information from local state is used.

Γ^r: When closed, the local state is reset. When open, the state is updated as in a “vanilla” RNN.

Note: These are usually called the “update” and “reset” gates.

GRUs

The state evolves according to

$$c_t = \tanh\left(W_{hx} x_t + \Gamma_t^r \odot W_{hh} h_{t-1} + b_h\right)$$

$$h_t = (1 - \Gamma_t^u) \odot c_t + \Gamma_t^u \odot h_{t-1}$$

GRUs

The state evolves according to

$$c_t = \tanh\left(W_{hx}x_t + \Gamma^r_t \odot W_{hh}h_{t-1} + b_h\right)$$

$$h_t = (1 - \Gamma^u_t) \odot c_t + \Gamma^u_t \odot h_{t-1}$$

- c_t is the "candidate state" computed using the usual "vanilla RNN" state, after possibly resetting some components.
- When the long-term memory gate is open ($\Gamma^u = 1$), the information gets sent through directly. *This deals with vanishing gradients*.
- The gates are multi-dimensional, applied componentwise
- Prediction of the next word is made using h_t.

GRUs

Everything needs to be differentiable, so the gate is actually “soft” and between zero and one.

The gates are computed as

$$\Gamma_t^u = \sigma\left(W_{ux} x_t + W_{uh} h_{t-1} + b_u\right)$$

$$\Gamma_t^r = \sigma\left(W_{rx} x_t + W_{rh} h_{t-1} + b_r\right)$$

where σ is the sigmoid function.

Putting it together

GRU state update equations

$$\Gamma_t^u = \sigma\left(W_{ux} x_t + W_{uh} h_{t-1} + b_u\right)$$

$$\Gamma_t^r = \sigma\left(W_{rx} x_t + W_{rh} h_{t-1} + b_r\right)$$

$$c_t = \tanh\left(W_{hx} x_t + \Gamma_t^r \odot W_{hh} h_{t-1} + b_h\right)$$

$$h_t = (1 - \Gamma_t^u) \odot c_t + \Gamma_t^u \odot h_{t-1}$$

There are minor variants on this architecture that are sometimes used.

Putting it together

The reset gate is sometimes moved inside the linear map

GRU state update equations

$$\Gamma_t^u = \sigma\left(W_{ux} x_t + W_{uh} h_{t-1} + b_u\right)$$

$$\Gamma_t^r = \sigma\left(W_{rx} x_t + W_{rh} h_{t-1} + b_r\right)$$

$$c_t = \tanh\left(W_{hx} x_t + W_{hh}\left(\Gamma_t^r \odot h_{t-1}\right) + b_h\right)$$

$$h_t = (1 - \Gamma_t^u) \odot c_t + \Gamma_t^u \odot h_{t-1}$$

There are minor variants on this architecture that are sometimes used.

GRUs: Example

Example:

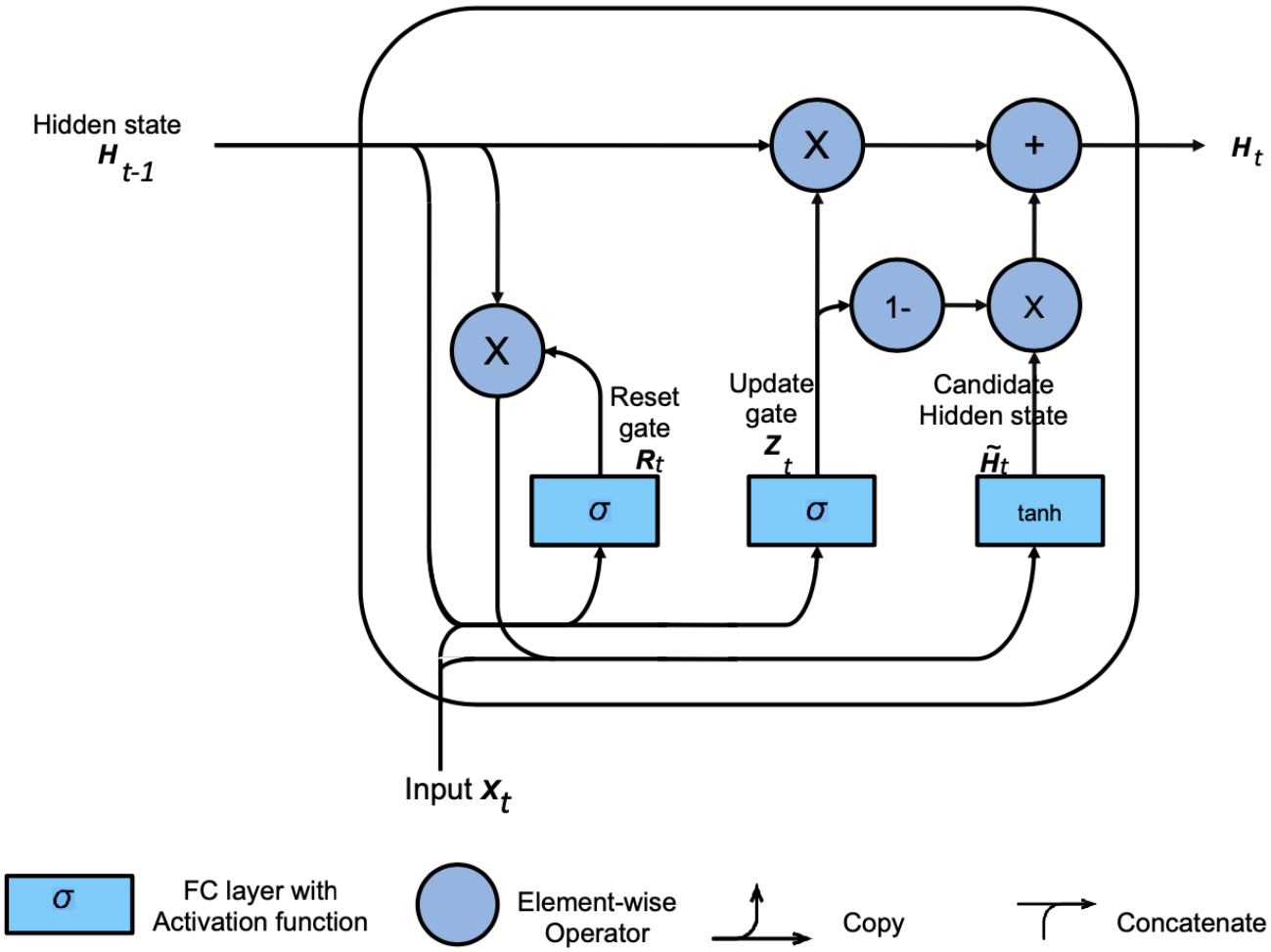
```
The leaves, as the weather turned cold in New
Haven, fell silently from the trees in November.
```

We want to keep `leaves` in memory. It's the subject of the sentence, and plural — syntax

It also has a "foliage" meaning that is relevant when we predict the words `trees` and `November` — semantics

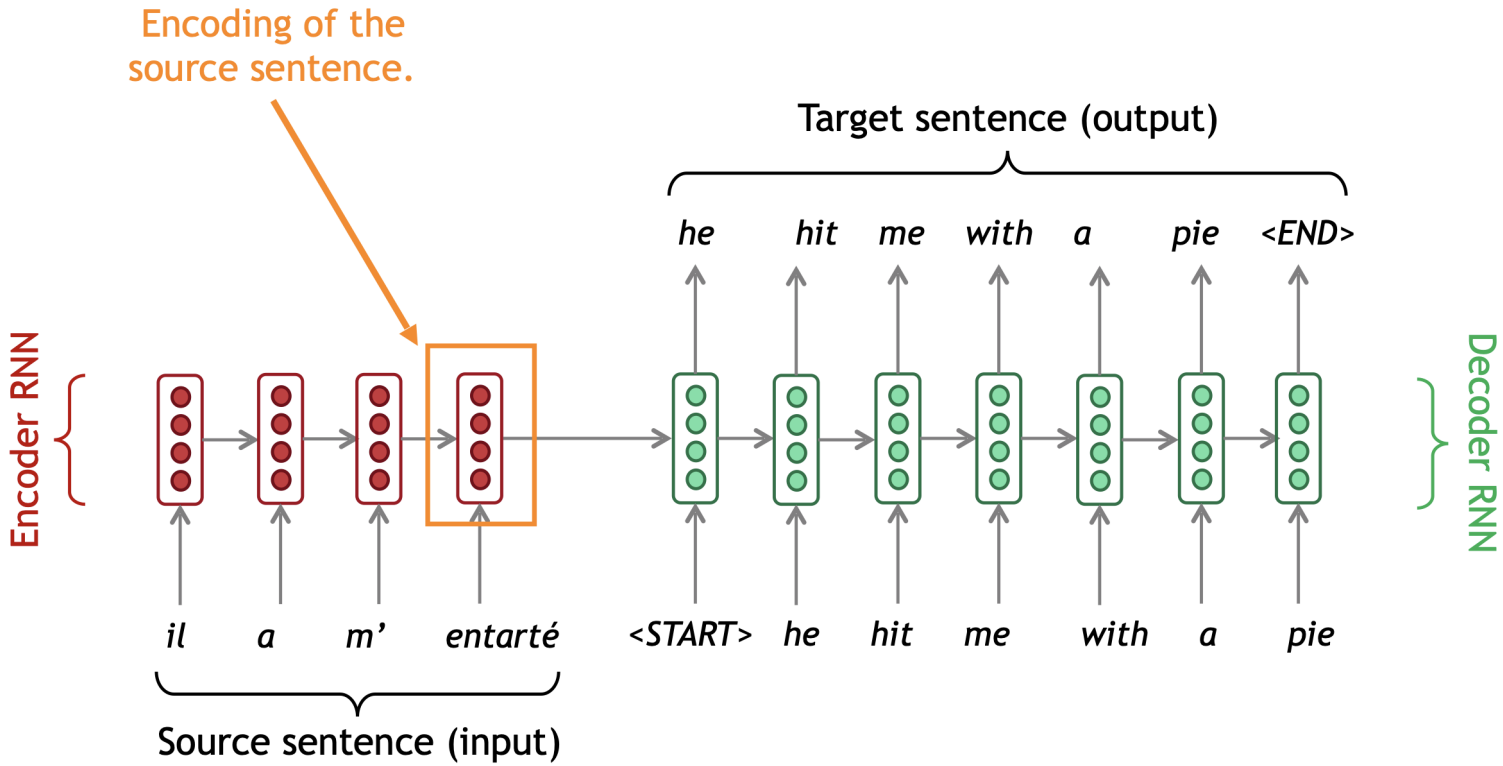
We stepped through on the blackboard how the GRU handles this

GRU Diagram (using slightly different notation)



Sequence-to-sequence models

- Important in translation
- Uses two RNNs (GRUs or LSTMs): Encoder and Decoder



"Sequence to sequence learning with neural networks," Sutskever et al. 2014, https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf. Figure source: http://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture08-nmt.pdf

Sequence-to-sequence models

- The goal of Seq2seq is to estimate the conditional probability

$$p(y_1, \ldots, y_T \mid x_1, \ldots, x_S)$$

- Encoder RNN computes the fixed dimensional representation $h(x_{1:S})$; decoder RNN then computes

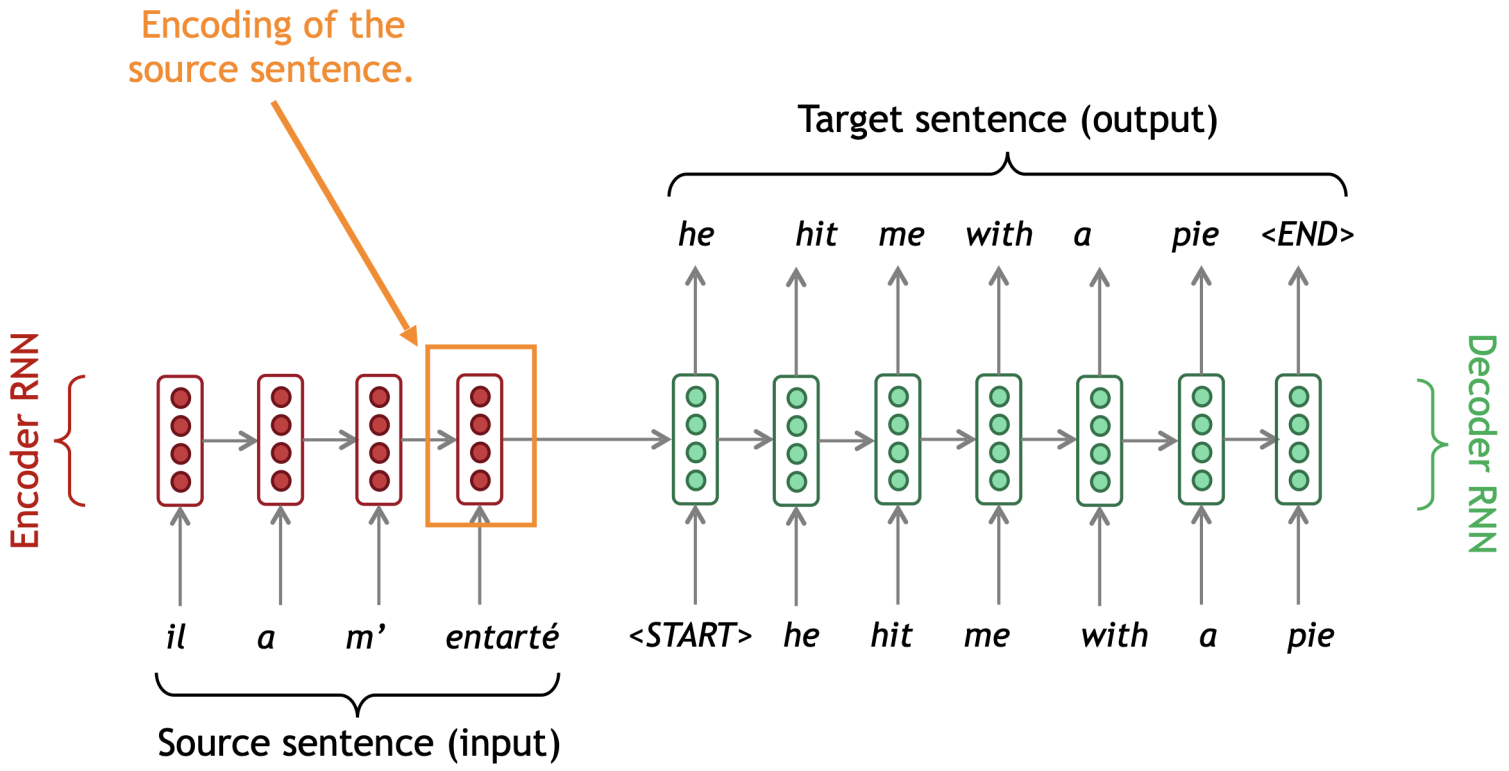
$$\prod_{t=1}^{T} p(y_t \mid h, y_1, \ldots, y_{t-1})$$

- Original paper uses 4-layer LSTMs with 1000 neurons in each layer; trained for 10 days for a machine translation task.

In the example of the previous slide, the state h is the red vector over entarté. Recall: This is very similar to what we did with LaTeX equation models conditioned on topics.

Sequence-to-sequence models

This results in a “bottleneck” problem—all the information needs to be funneled through that final state.



Important modification: Attention

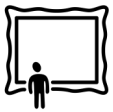
Pay attention!

entarté!

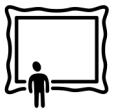
---

https://en.wikipedia.org/wiki/Pieing

Attention/Alignment

 On each step of decoding, directly connect to the encoder, and focus on a particular part of the source sequence

Attention/Alignment



On each step of decoding, directly connect to the encoder, and focus on a particular part of the source sequence

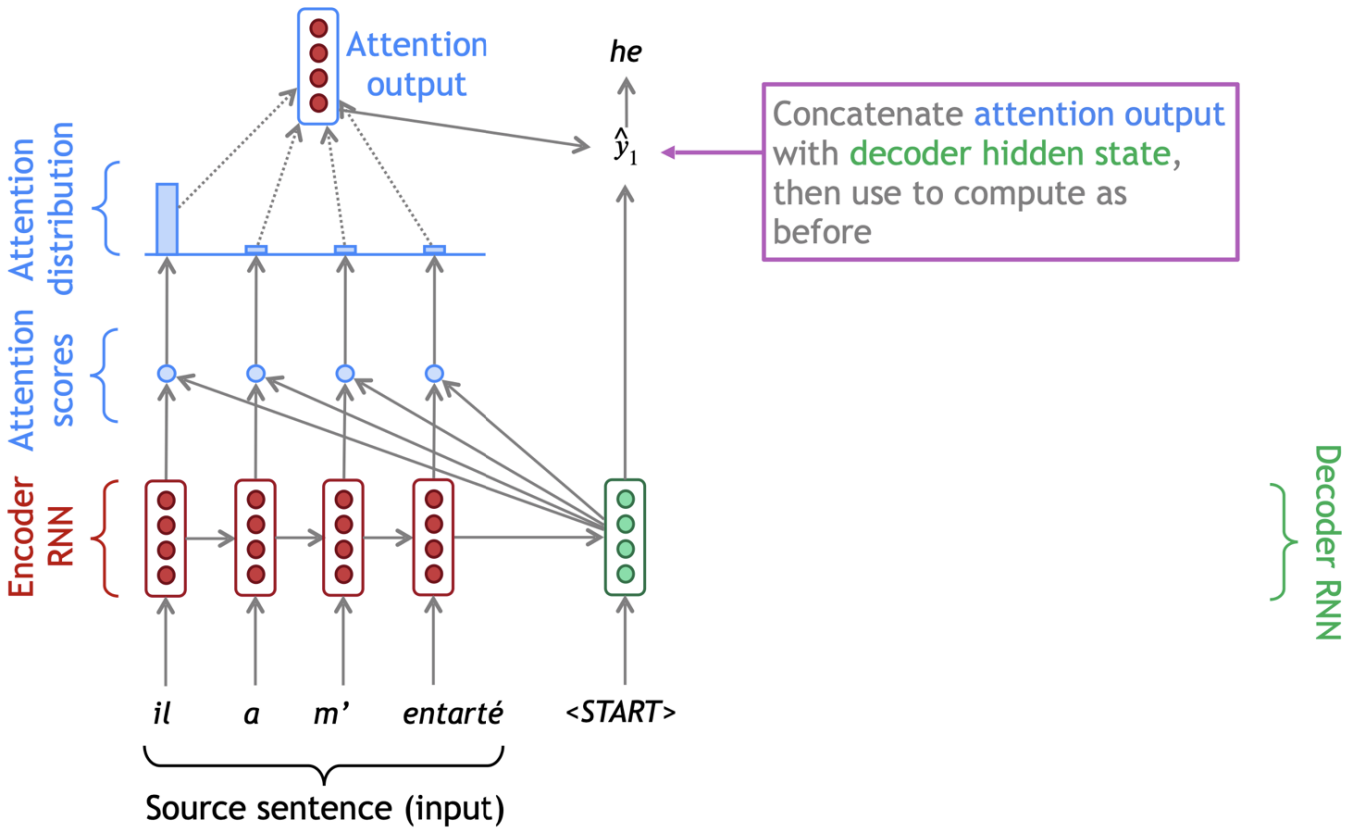
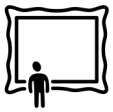


figure source: http://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture08-nmt.pdf

Attention/Alignment



On each step of decoding, directly connect to the encoder, and focus on a particular part of the source sequence

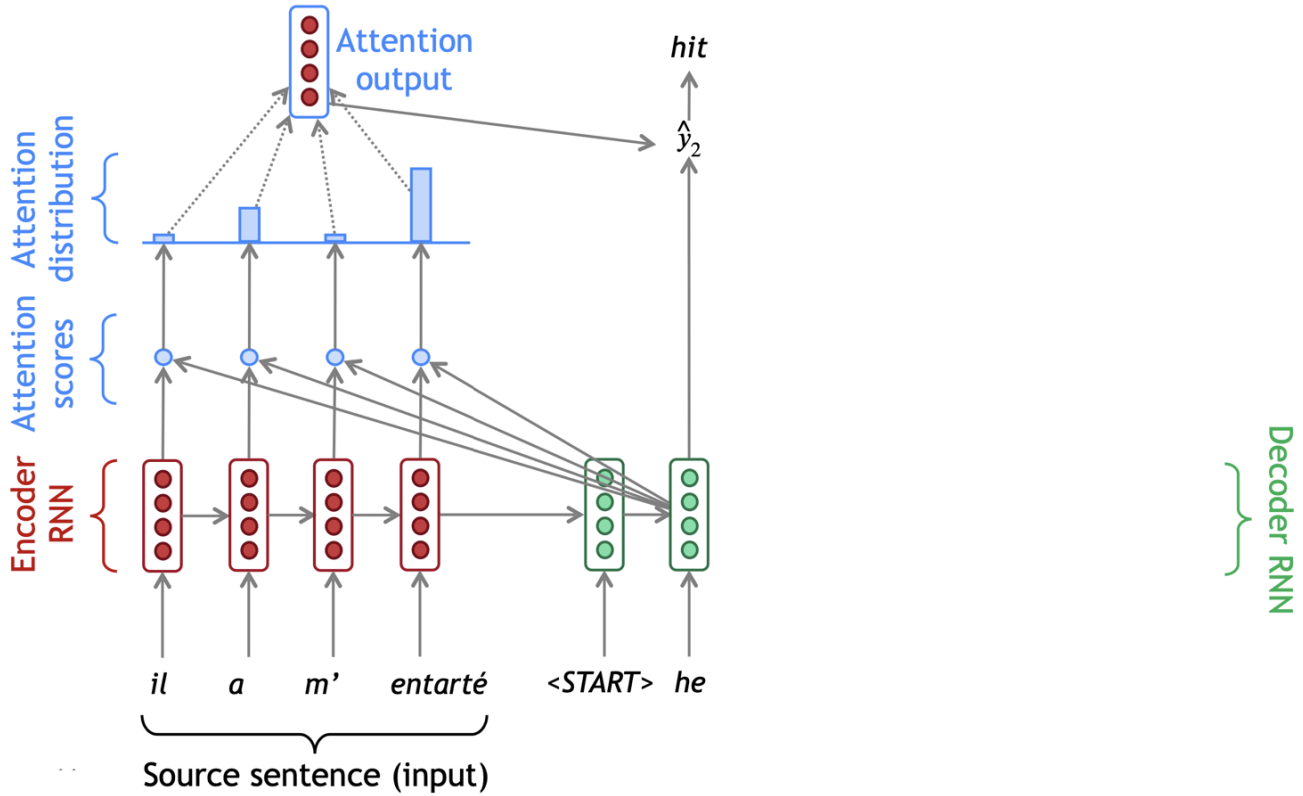
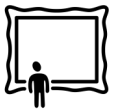


figure source: http://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture08-nmt.pdf

Attention/Alignment



On each step of decoding, directly connect to the encoder, and focus on a particular part of the source sequence

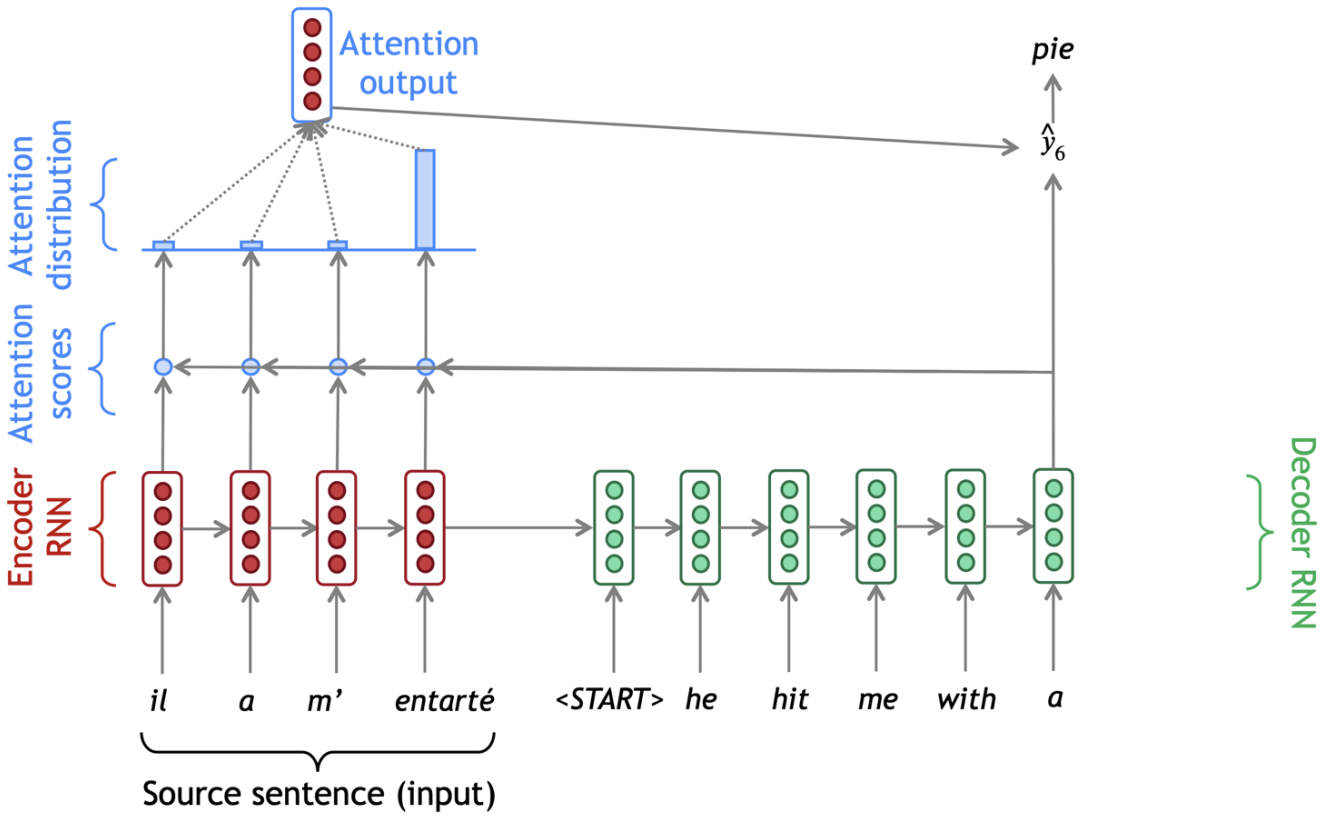
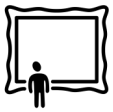


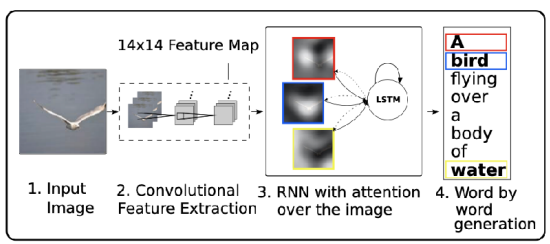
figure source: http://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture08-nmt.pdf

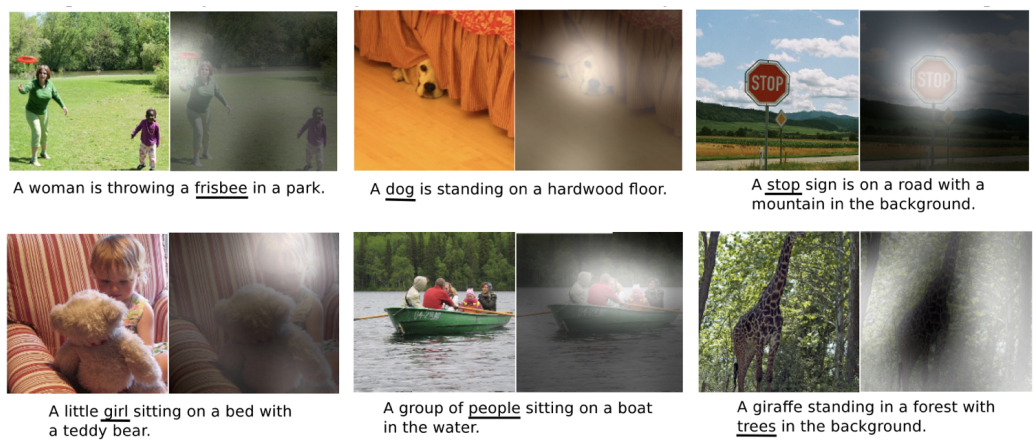
Attention/Alignment



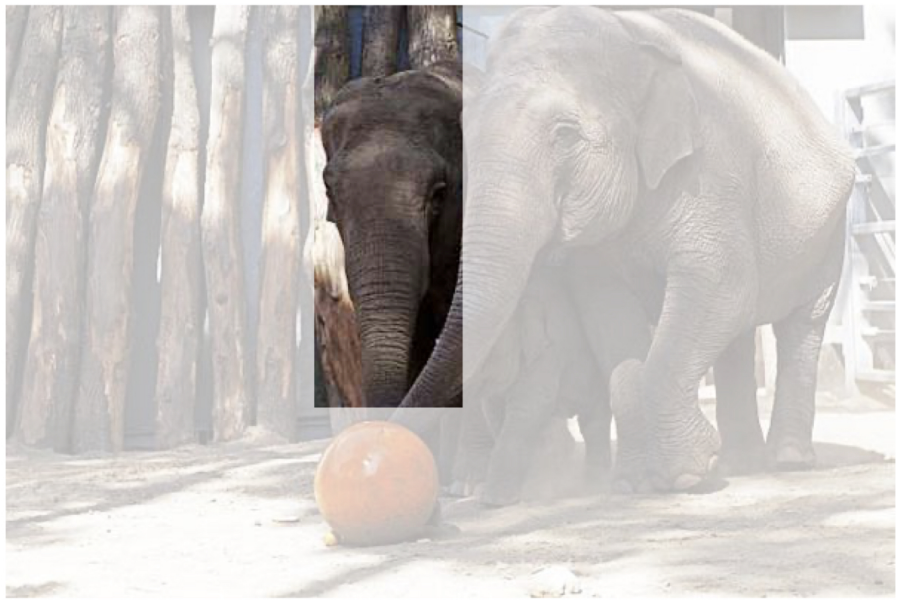
On each step of decoding, directly connect to the encoder, and focus on a particular part of the source sequence

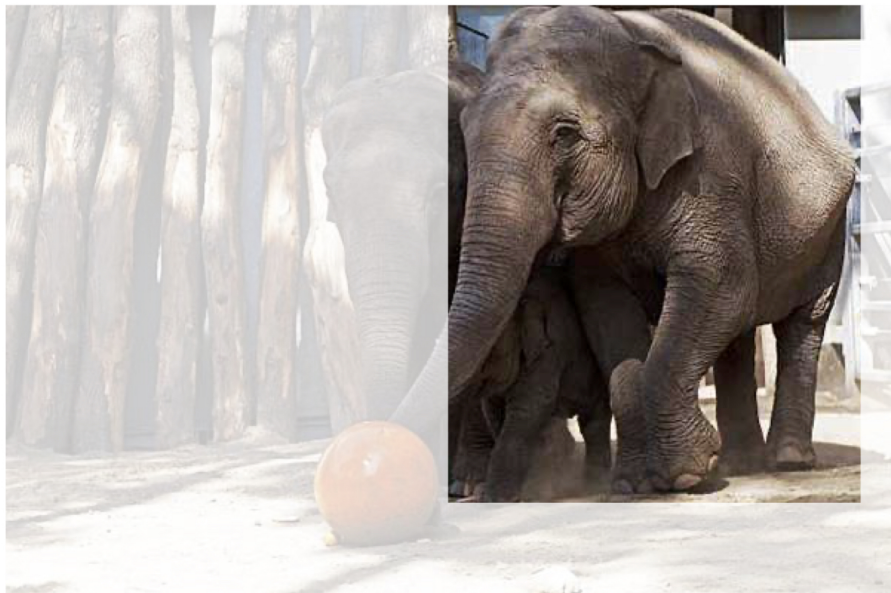
Attention can also be used for other generative models





* "Show, attend and tell: neural image caption generation with visual attention", Xu et al. 2016, https://arxiv.org/pdf/1502.03044.pdf





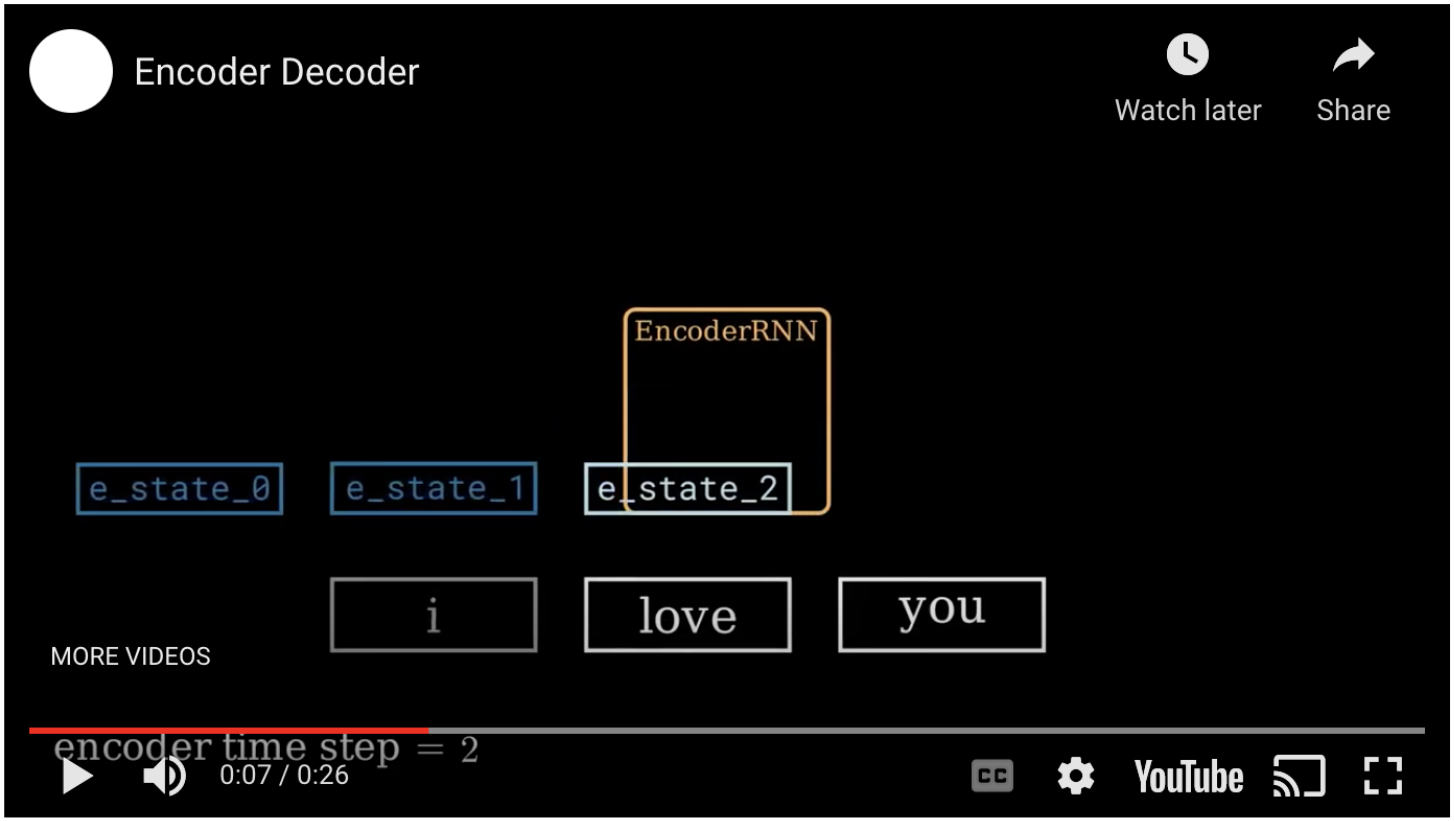
The two elephants played with an orange ball

The two elephants played with an orange ball

 → orange

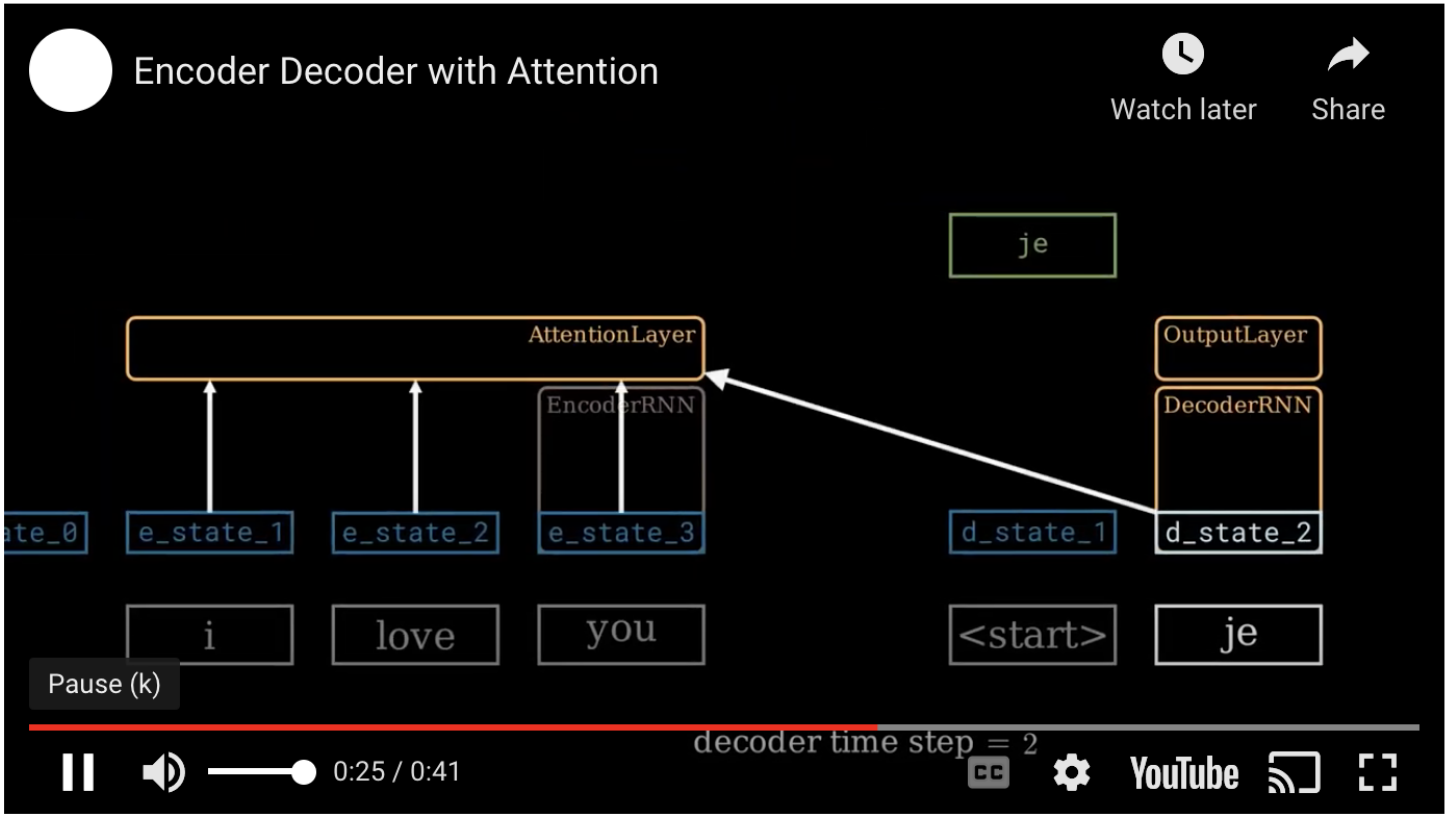
 → ball

Attention animated



Encoder's hidden state on the last times step becomes the initial state of Decoder. Note: <start> and <end> are special words/tokens to indicate the start and end of the sequence in Decoder. There's nothing special about <start> and <end> tokens other than their role as markers.

Attention animated



We use context vector combined with decoder hidden state to generate the final output in each time step.

https://medium.com/@cakraww/from-recurrent-neural-network-rnn-to-attention-explained-intuitively-7fd7caeef6af

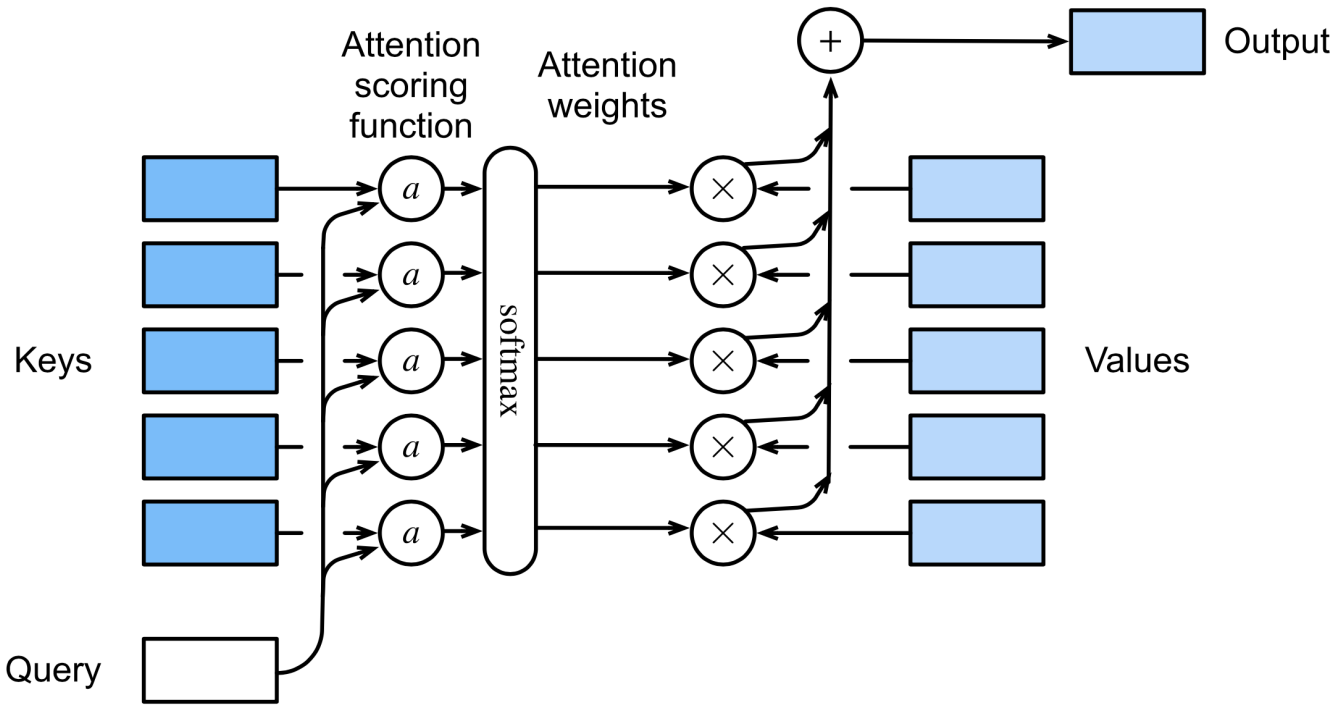
Attention in terms of query/key/values

Basic idea behind attention mechanisms:

- Neural networks so far: Hidden activations are linear combinations of input activations, followed by nonlinearity: $h = \varphi(Wu)$
- A more flexible model:
  - We have a set of m feature vectors or values $V \in \mathbb{R}^{m \times v}$
  - The model dynamically chooses which to use
  - based on how similar a query vector $q \in \mathbb{R}^q$ is to a set of m keys $K \in \mathbb{R}^{m \times k}$.
  - If q is most similar to key i, then we use value (feature) v_i.

Attention in terms of query/key/values

Basic idea behind attention mechanisms:



Attention in terms of query/key/values

Basic idea behind attention mechanisms:

$$\begin{aligned}\text{Attn}(q, \{(k_1, v_1), \ldots (k_m, v_m)\}) &= \text{Attn}(q, (k_{1:m}, v_{1:m})) \\ &= \sum_{i=1}^{m} \alpha(q, k_{1:m}) v_i\end{aligned}$$

where weights α_i are softmax of attention scores $a(q, k)$:

$$\alpha_i(q, k_{1:m}) = \frac{\exp(a(q, k_i))}{\sum_{j=1}^{m} \exp(a(q, k_j))}$$

Kernel regression as attention

Recall the kernel regression estimator:

$$\widehat{m}(x) = \sum_{i=1}^{n} \alpha(x, x_{1:n}) y_i$$

Here the attention scores (for Gaussian kernel) are

$$a(x, x_i) = -\frac{1}{2h^2} \|x - x_i\|^2$$

query:	test point x
keys:	data $x_1, \ldots, x_n$
values:	responses $y_1, \ldots, y_n$

Dot product attention

Note that if x_i and x have a fixed norm then the attention scores are just scaled dot products:

$$a(x, x_i) = \frac{1}{h^2} x^T x_i$$

If query and keys have same dimension d, dot product attention is

$$a(q, k) = q^T k / \sqrt{d} \in \mathbb{R}$$

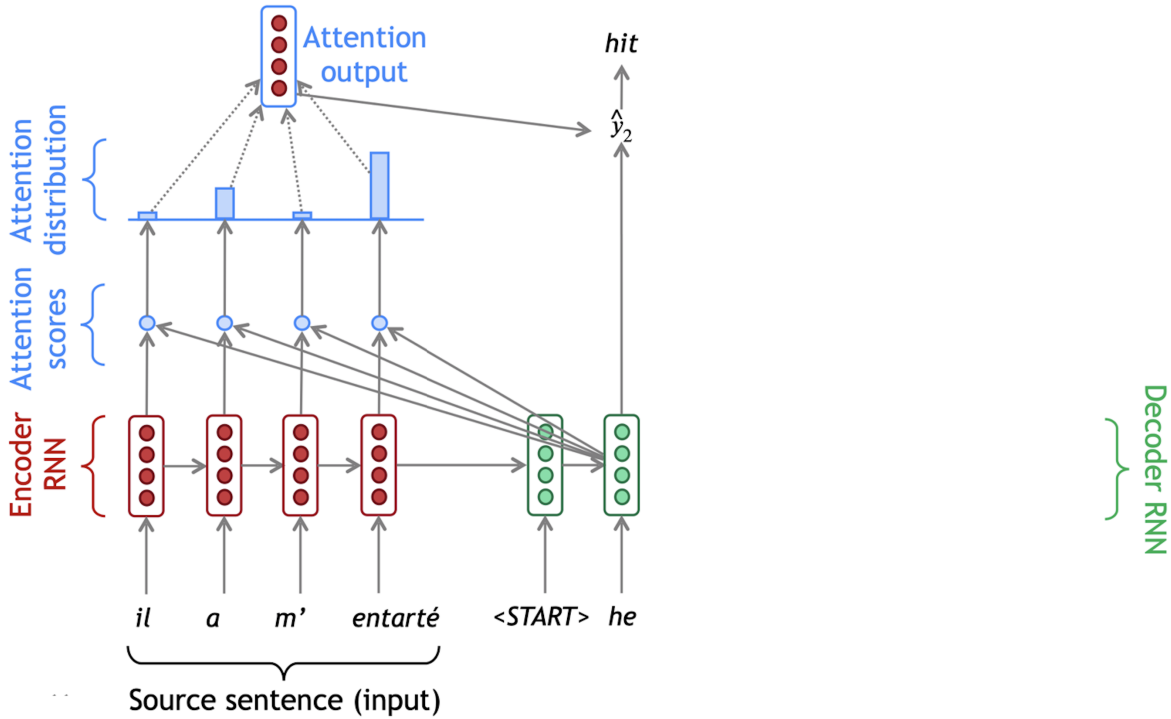
and

$$\text{Attn}(Q, K, V) = \text{Softmax}\left(QK^T / \sqrt{d}\right) V$$

Scaling by $\sqrt{d}$ keeps variance constant

Retour d'entarté

In the *entarté* example, the keys and values are the red state vectors, the queries are the green state vectors



Transformers (next lecture)

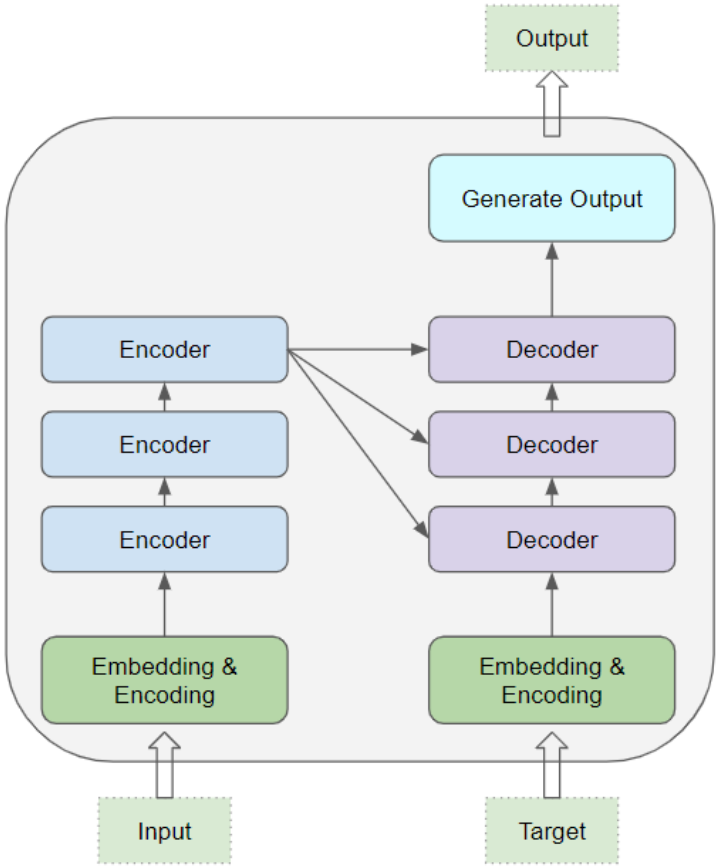
The current state-of-the-art is based on *transfomers*

- Attention is the key ingredient
- Rather than processing sequences word-by-word, transformers handle larger chunks of text at once
- The separation between words matters less

Combining attention vectors

- Different attention vectors are stacked up on top of each other
- Like channels and feature maps in a CNN
- Words “hungry” and “sweet” are predicted using all of them

Transformer architecture



Summary

- Seq2seq pairs together two RNNs, encoding the input sequence as a state, and decoding to generate an output sequence.
- Attention is a type of alignment between related words
- Attention allows dynamic construction of more informative states for predicting the next word
- Transformers process all words together, using layers of encoders and decoders, each with an attention component — details next time