

```
In [3]: response = gpt3("Yale students are among the best in the world.", temp=.9)
print(response)
```

This is a fact that is indisputable. The students who attend Yale come from the top of their class and are among the most talented and intelligent young people in the world. The faculty at Yale is world-renowned, and the resources available to students are unmatched. Yale graduates go on to do great things in the world, and the education they receive at Yale is a big part of that.

```
In [4]: response = gpt3("Yale students are among the best in the world.", temp=.9, engine="text-ada-001")
print(response)
```

S&DS 365 / 665

## Intermediate Machine Learning

According to a report from the National Center for Education Statistics, the percentage of students who earn a minor in theater and television arts is higher than the percentage of students who earn a minor in theater and television arts. A report from 2019 found that out of 802 students at Yale who earned a minor in theater and television arts, 11% had earned a minor in theater and television arts. This means that the number of students at Yale who earn a minor in theater and television arts is high in terms of percentage of students who achieve this goal.

# Sequence-to-sequence models

November 28

# Welcome back!

- Quiz 4 grades posted
- Quiz 5 (last!) on Wednesday, Nov 30 (RL, HMMs, RNNs, GRUs)
- Assn 5 out; due next Wednesday
- Final exam: Tuesday Dec 20, 7pm in LC 102
- Practice exams are posted
- Review sessions TBA

# Quiz 4

## Quiz Summary

Section Filter ▾

 Student Analysis

 Item Analysis

 Average Score

**86%**

 High Score

**100%**

 Low Score

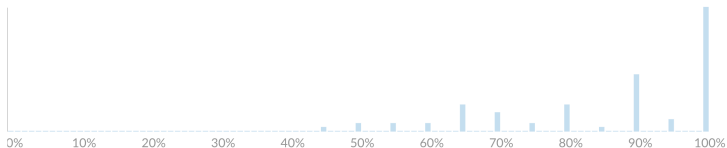
**45%**

 Standard Deviation

**1.56**

 Average Time

**26:08**



# Notes posted

- HMMs and Kalman filters
- Mixture models

# Notes posted

			vanilla	mixtures		
12	Nov 14, 16	Sequential models	<a href="#">vanilla RNN</a> <a href="#">Fakespeare</a> <a href="#">GRU</a>	Nov 14: HMMs and RNNs Nov 16: RNNs, GRUs, LSTMs, and all that	<a href="#">TensorFlow: Text generation</a> <a href="#">Notes on HMMs</a> PML Chapter 15	Nov 16: Assn 4 in <a href="#">Assn 5 out</a>
13	Nov 21, 23	No class, Thanksgiving break				
14	Nov 28, 30	Attention and sequence-to-sequence models	<a href="#">GPT-3 demo</a> <a href="#">Codex demo</a>	Nov 28: Sequence-to-sequence models Nov 30: Attention	<a href="#">Notes on mixtures</a> PML Sections 15.4, 15.5	Nov 30: Quiz 5

# For Today



# For Today

## Sequence models

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

# Sequence models

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

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

which can be factored as

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



# 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  $x_1, x_2, \dots, x_t$
- We talked about state models: HMMs and RNNs

# Hidden Markov Model

In an HMM, current output generated from a latent variable.

- State is chosen stochastically (that is, probabilistically, or randomly)
- As a consequence, the dependence on earlier  $x_t$ 's extends further back in time than just the previous time (as would be the case for a Markov model)

# 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

# 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

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

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

# Memory circuits

Intuition: In language modeling, may be useful to remember/forget gender or number of subject so that personal pronouns (“he” vs. “she” vs. “they”) can be used appropriately.

A variant called “Long Short-Term Memory” (LSTM) RNNs has a special hidden layer that “includes” or “forgets” information from the past.

A simpler alternative to the LSTM circuit is called the *Gated Recurrent Unit* (GRU)

# 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  $\Gamma$  (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:

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

$\Gamma^r$ : 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. I find this terminology super confusing. More suggestive names might be the long distance memory gate ( $\Gamma^u$ ) and the short distance memory gate ( $\Gamma^r$ ).

# GRUs

The state evolves according to

$$\mathbf{c}_t = \tanh \left( \mathbf{W}_{hx} \mathbf{x}_t + \mathbf{W}_{hh} \left( \Gamma_t^r \odot \mathbf{h}_{t-1} \right) + \mathbf{b}_h \right)$$

$$\mathbf{h}_t = (1 - \Gamma_t^u) \odot \mathbf{c}_t + \Gamma_t^u \odot \mathbf{h}_{t-1}$$

# GRUs

The state evolves according to

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

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

- $c_t$  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

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

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

where  $\sigma$  is the sigmoid function.

# Putting it together

## GRU state update equations

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

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

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

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

# Putting it together

The reset gate is sometimes moved outside the linear map (Assn 5):

## GRU state update equations

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

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

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

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

# GRUs: Example

Example:

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

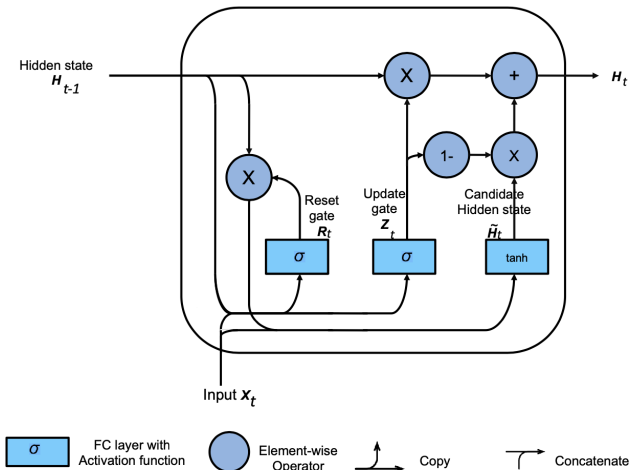
We want to keep `leaves` in memory. It's the subject of the sentence, and plural — syntax

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

Let's step through on the blackboard how the GRU handles this.

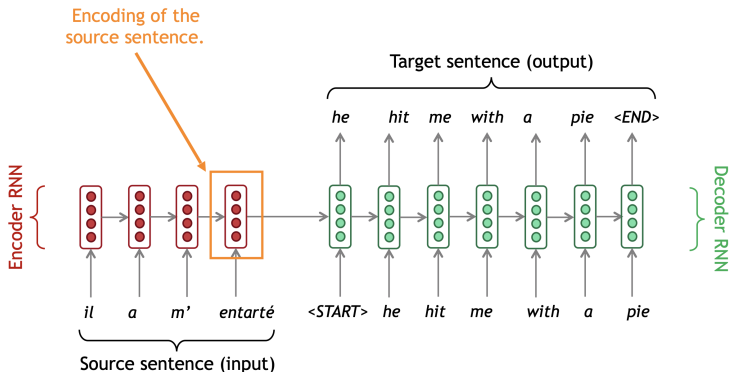


# GRU Diagram (using slightly different notation)



# 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

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

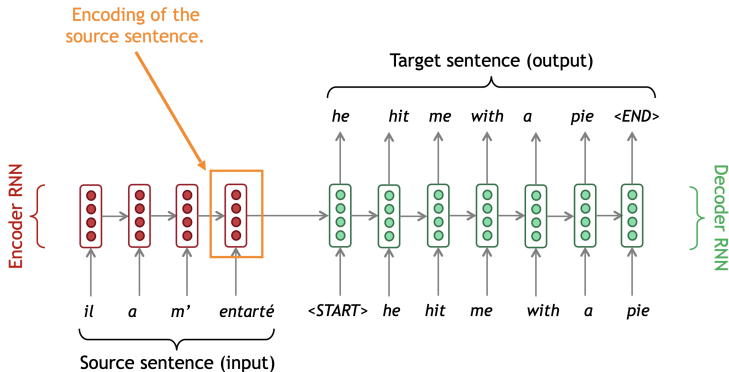
- Encoder RNN computes the fixed dimensional representation  $h(x)$ ; decoder RNN then computes

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

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

# Sequence-to-sequence models

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



Important modification: Attention

# Attention



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

# Attention



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

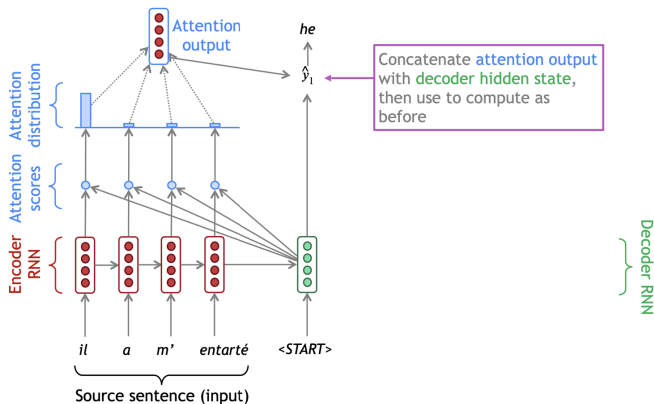


figure source: <http://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture08-nmt.pdf>

# Attention



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

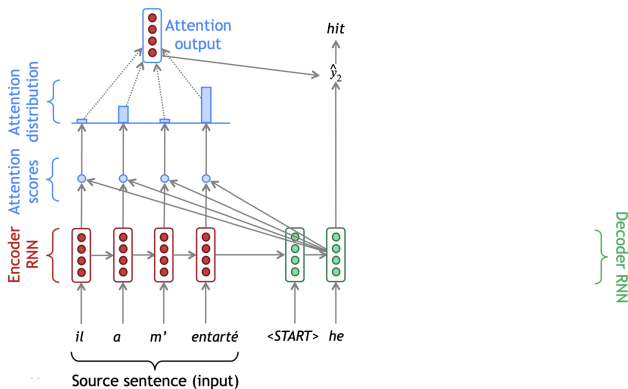


figure source: <http://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture08-nmt.pdf>

# Attention



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

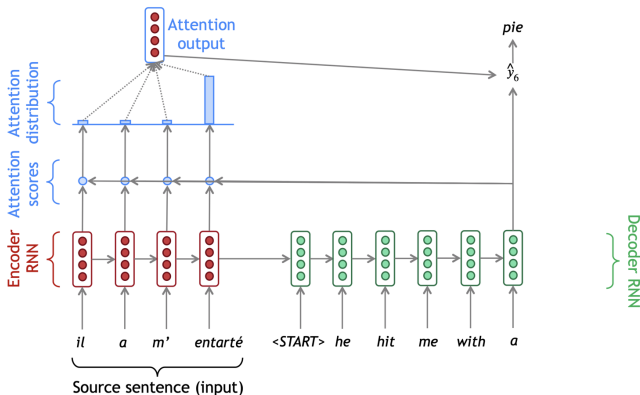


figure source: <http://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture08-nmt.pdf>

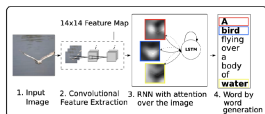


# Attention



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



A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.



A stop sign is on a road with a mountain in the background.



A little girl sitting on a bed with a teddy bear.



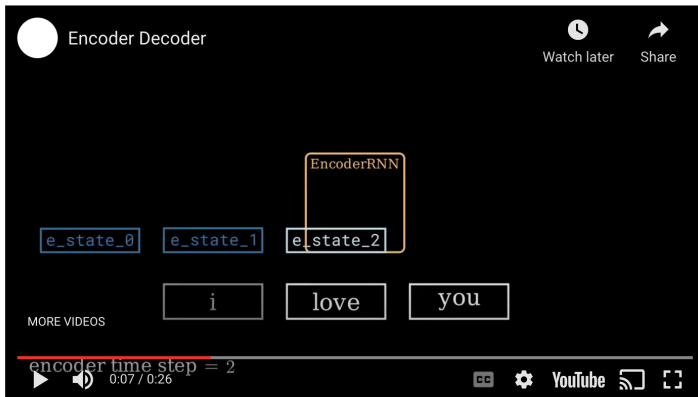
A group of people sitting on a boat in the water.



A giraffe standing in a forest with trees in the background.

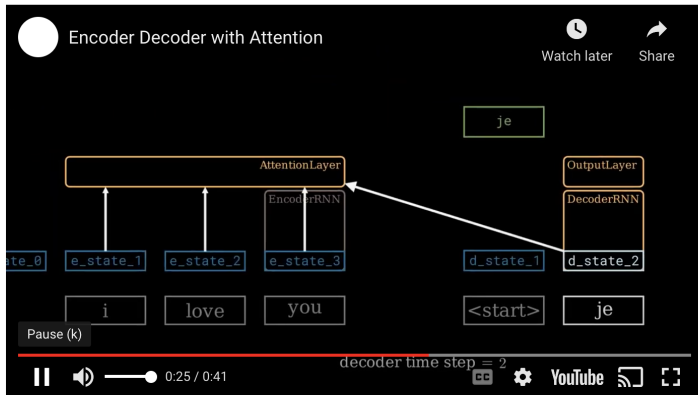
\* "Show, attend and tell: neural image caption generation with visual attention", Xu et al. 2016, <https://arxiv.org/pdf/1502.03044.pdf>

# 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



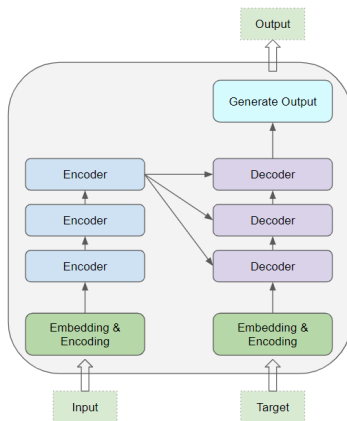
We use context vector combined with decoder hidden state to generate the final output in each time step.

# Transformers (next class)

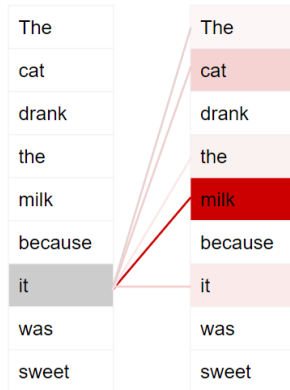
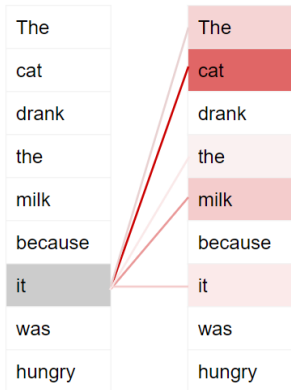
The current state-of-the-art is based on *transformers*

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

# Attention: example



# Attention: example



# 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