S&DS 365 / 665

Intermediate Machine Learning

Attention

April 25

Yale

Last week!

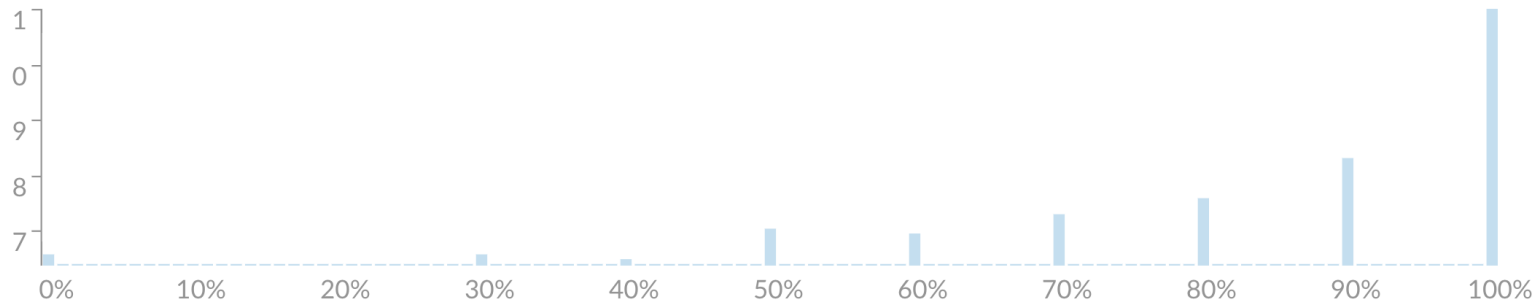
- Quiz 4 grades posted
- Assn 4 due this Wednesday
- Final exam: Saturday May 7 at 2pm in SPL 59
- Practice exam by Monday May 2
- Review sessions TBA

Quiz 4

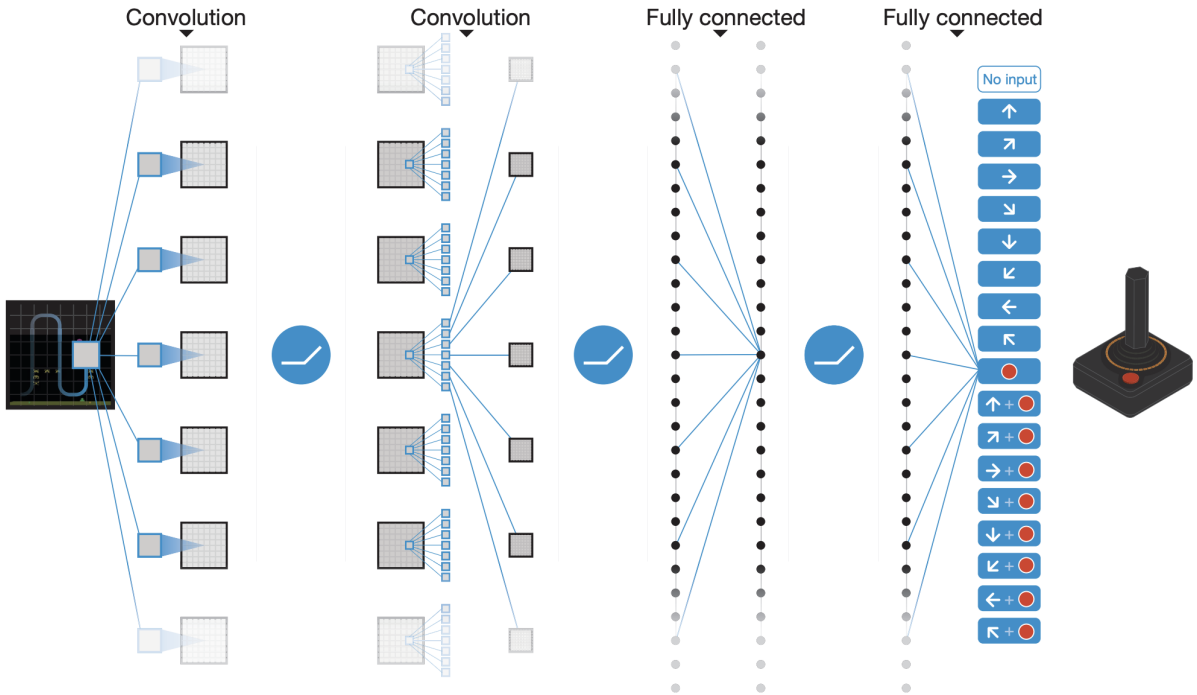
Quiz Summary

Section Filter ▾ | Student Analysis | Item Analysis

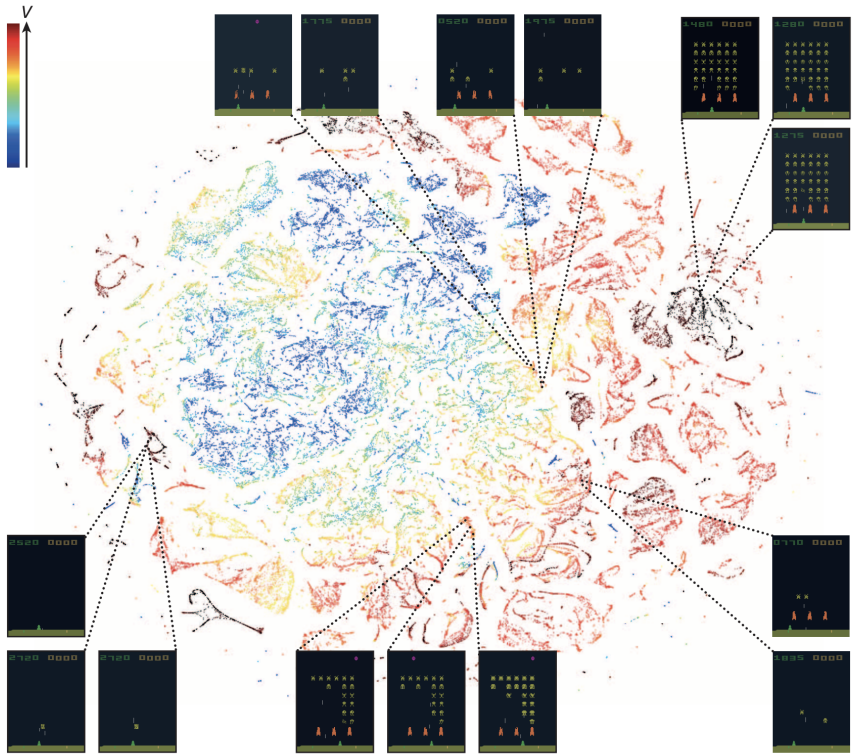
Average Score	High Score	Low Score	Standard Deviation	Average Time
84%	100%	0%	2.09	21:33



Quiz 4


Convolution
Convolution
Fully connected
Fully connected
No input

Quiz 4



accepted either third or fourth answers

For Today

For Today

Sequence models

- Memory circuits: LSTMs and GRUs (recap)
- GPT-3 demo
- Sequence to sequence models
- Attention mechanisms

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 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_ts extends much further back in time.

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_ts 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

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

We'll modify this with two types of "neural circuits" for storing information to be used downstream

Memory circuits

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

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.

Useful for things like matching parentheses, etc.

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

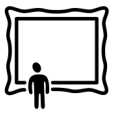
“Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation”, Cho et al. 2014
https://arxiv.org/pdf/1406.1078v3.pdf

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 (simplified)

Make use of an update "gate" Γ_t^u. This is either zero or one.

If $\Gamma_t^u = 1$ we store/update information in a memory cell c_t.

If $\Gamma_t^u = 0$ we just keep the information in the cell; don't update it.

GRUs (simplified)

The memory cell evolves according to

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

where h_t is the state computed using the usual "vanilla RNN".

The next word is predicted using the memory cell c_t together with the usual RNN state h_t.

GRUs (simplified)

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

The equations are then

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

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

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

where σ is the sigmoid function.

The next word is predicted using the memory cell c_t together with the usual RNN state h_t.

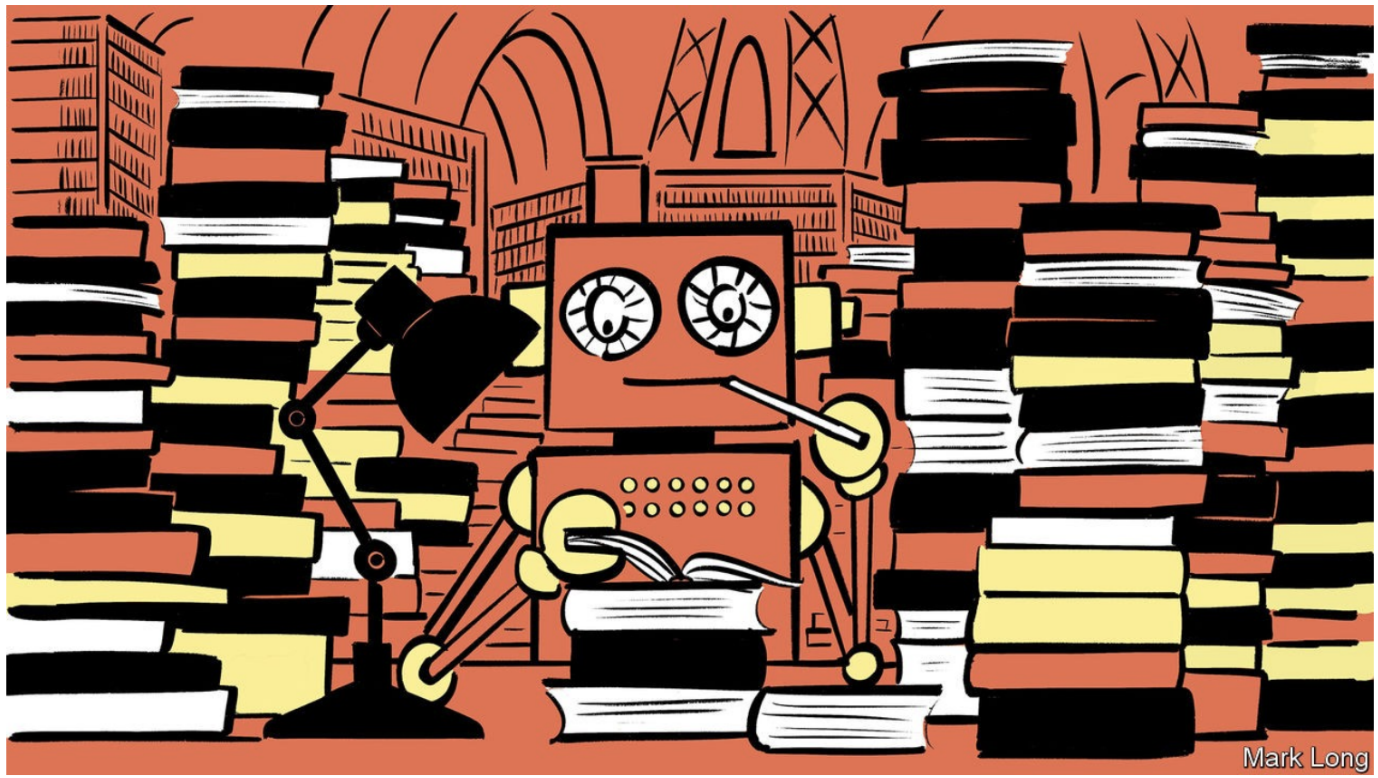
Science & technology

Aug 6th 2020 edition ›

Artificial intelligence

A new AI language model generates poetry and prose

GPT-3 can be eerily human-like—for better and for worse



The New York Times

Account

Meet GPT-3. It Has Learned to Code (and Blog and Argue).

The latest natural-language system generates tweets, pens poetry, summarizes emails, answers trivia questions, translates languages and even writes its own computer programs.

A.I. Is Mastering Language. Should We Trust What It Says?

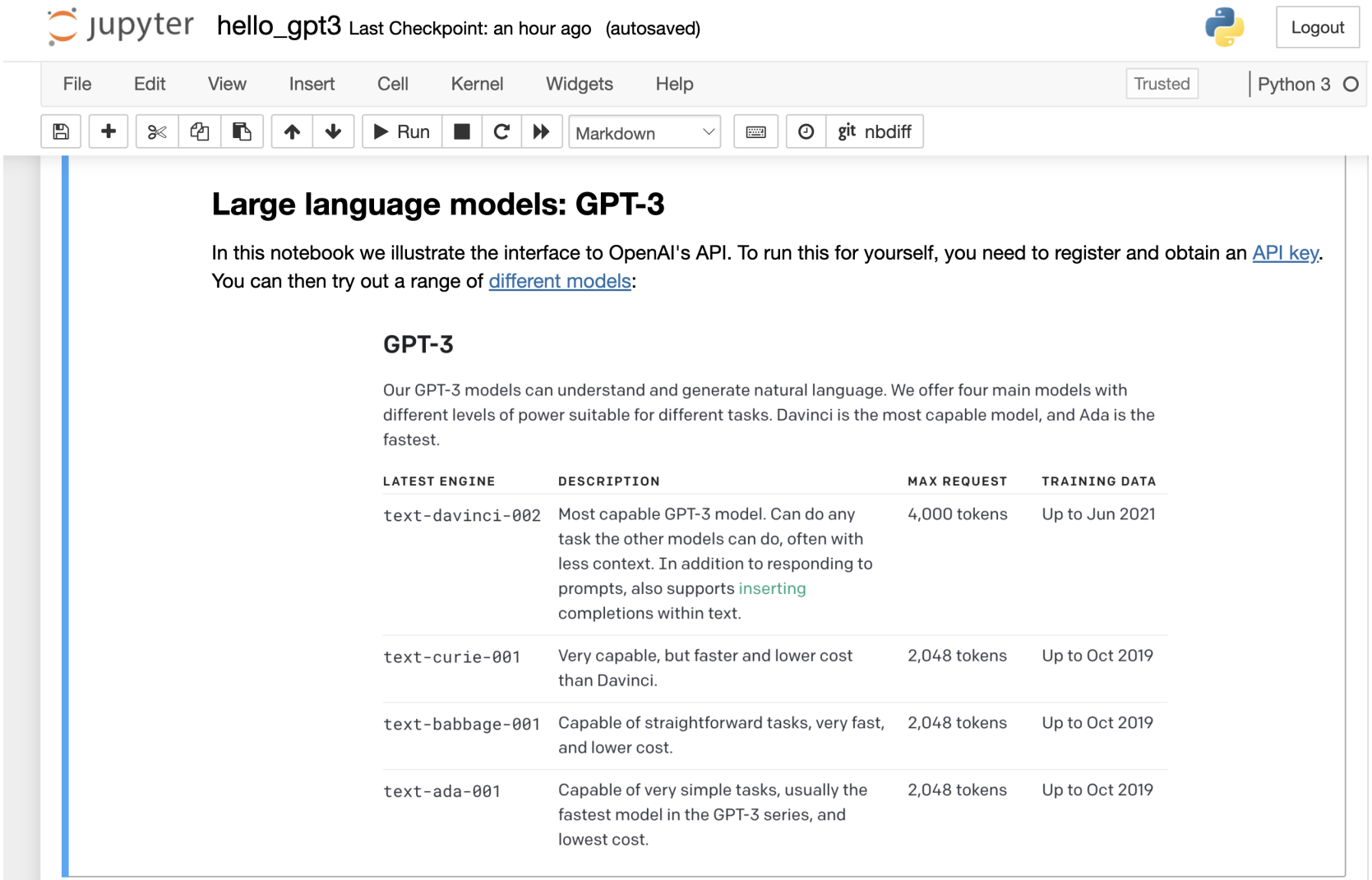
OpenAI's GPT-3 and other neural nets can now write original prose with mind-boggling fluency — a development that could have profound implications for the future.

GPT-3

Our GPT-3 models can understand and generate natural language. We offer four main models with different levels of power suitable for different tasks. Davinci is the most capable model, and Ada is the fastest.

LATEST ENGINE	DESCRIPTION	MAX REQUEST	TRAINING DATA
`text-davinci-002`	Most capable GPT-3 model. Can do any task the other models can do, often with less context. In addition to responding to prompts, also supports inserting completions within text.	4,000 tokens	Up to Jun 2021
`text-curie-001`	Very capable, but faster and lower cost than Davinci.	2,048 tokens	Up to Oct 2019
`text-babbage-001`	Capable of straightforward tasks, very fast, and lower cost.	2,048 tokens	Up to Oct 2019
`text-ada-001`	Capable of very simple tasks, usually the fastest model in the GPT-3 series, and lowest cost.	2,048 tokens	Up to Oct 2019

While Davinci is generally the most capable, the other models can perform certain tasks extremely well with significant speed or cost advantages. For example, Curie can perform many of the same tasks as Davinci, but faster and for 1/10th the cost.


jupyter hello_gpt3 Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Markdown
git nbdiff
Large language models: GPT-3
In this notebook we illustrate the interface to OpenAI's API. To run this for yourself, you need to register and obtain an API key.
You can then try out a range of different models:
GPT-3
Our GPT-3 models can understand and generate natural language. We offer four main models with different levels of power suitable for different tasks. Davinci is the most capable model, and Ada is the fastest.
LATEST ENGINE
DESCRIPTION
MAX REQUEST
TRAINING DATA
text-davinci-002
Most capable GPT-3 model. Can do any task the other models can do, often with less context. In addition to responding to prompts, also supports inserting completions within text.
4,000 tokens
Up to Jun 2021
text-curie-001
Very capable, but faster and lower cost than Davinci.
2,048 tokens
Up to Oct 2019
text-babbage-001
Capable of straightforward tasks, very fast, and lower cost.
2,048 tokens
Up to Oct 2019
text-ada-001
Capable of very simple tasks, usually the fastest model in the GPT-3 series, and lowest cost.
2,048 tokens
Up to Oct 2019

Chilling out

The code uses a `temperature` parameter.

Consider models of the form

$$p(w_1, \ldots, w_n) \propto e^{-\frac{1}{T} E(w_1, \ldots, w_n)}$$

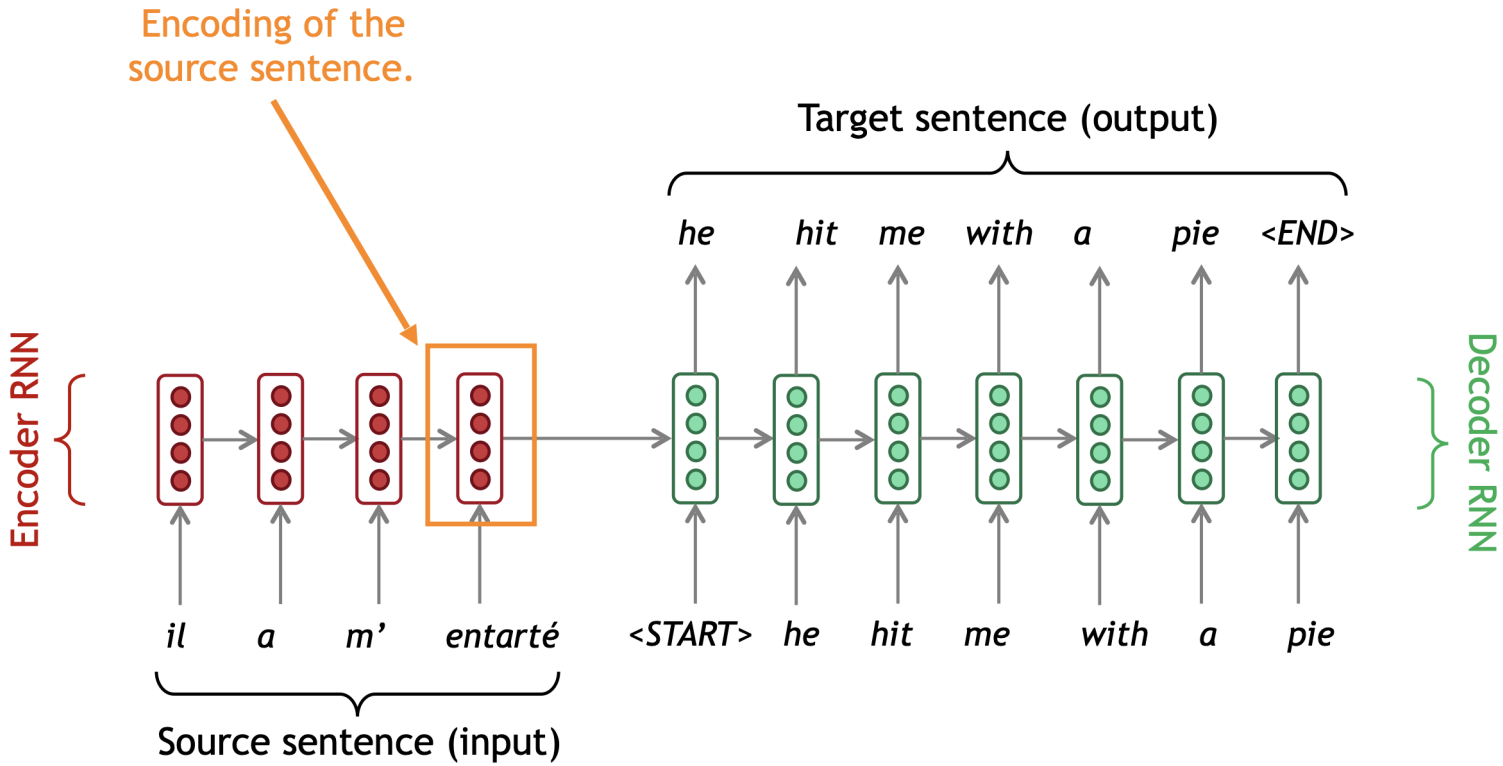
where E is an "energy" and T is a "temperature"

As $T \to 0$ the probability becomes peaked on *low energy* sequences.

For low temperatures, the code generates the more likely words. For higher temperatures the generative model is more "creative"

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 $h(x)$; decoder RNN then computes

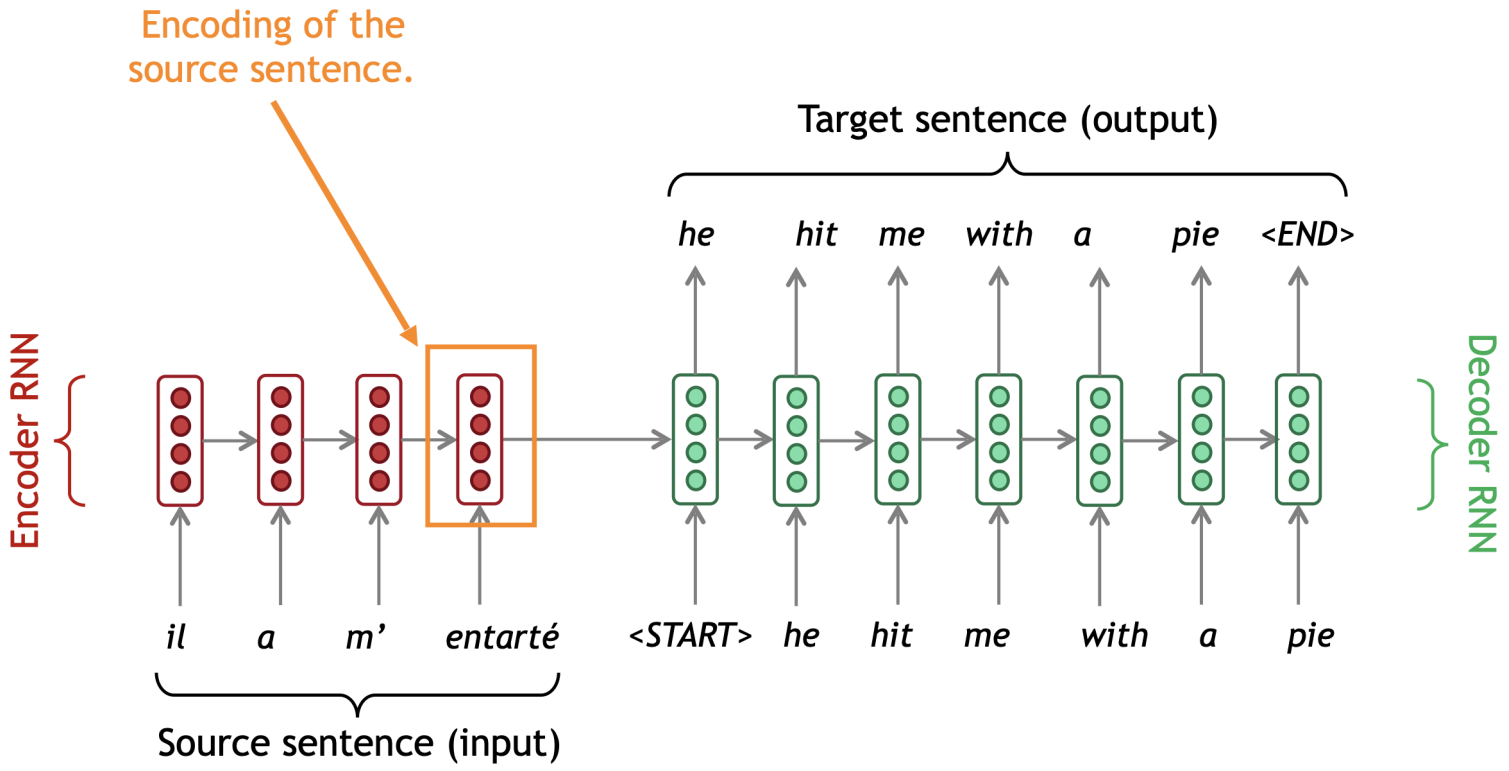
$$\prod_{t=1}^{T} p(y_t \mid h, y_1, \ldots, y_{t-1})$$

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

Recall: This is very similar to what we did with LaTeX equation models conditioned on topics.

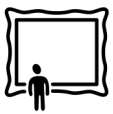
Sequence-to-sequence models

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

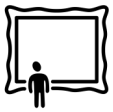


Important modification: Attention

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

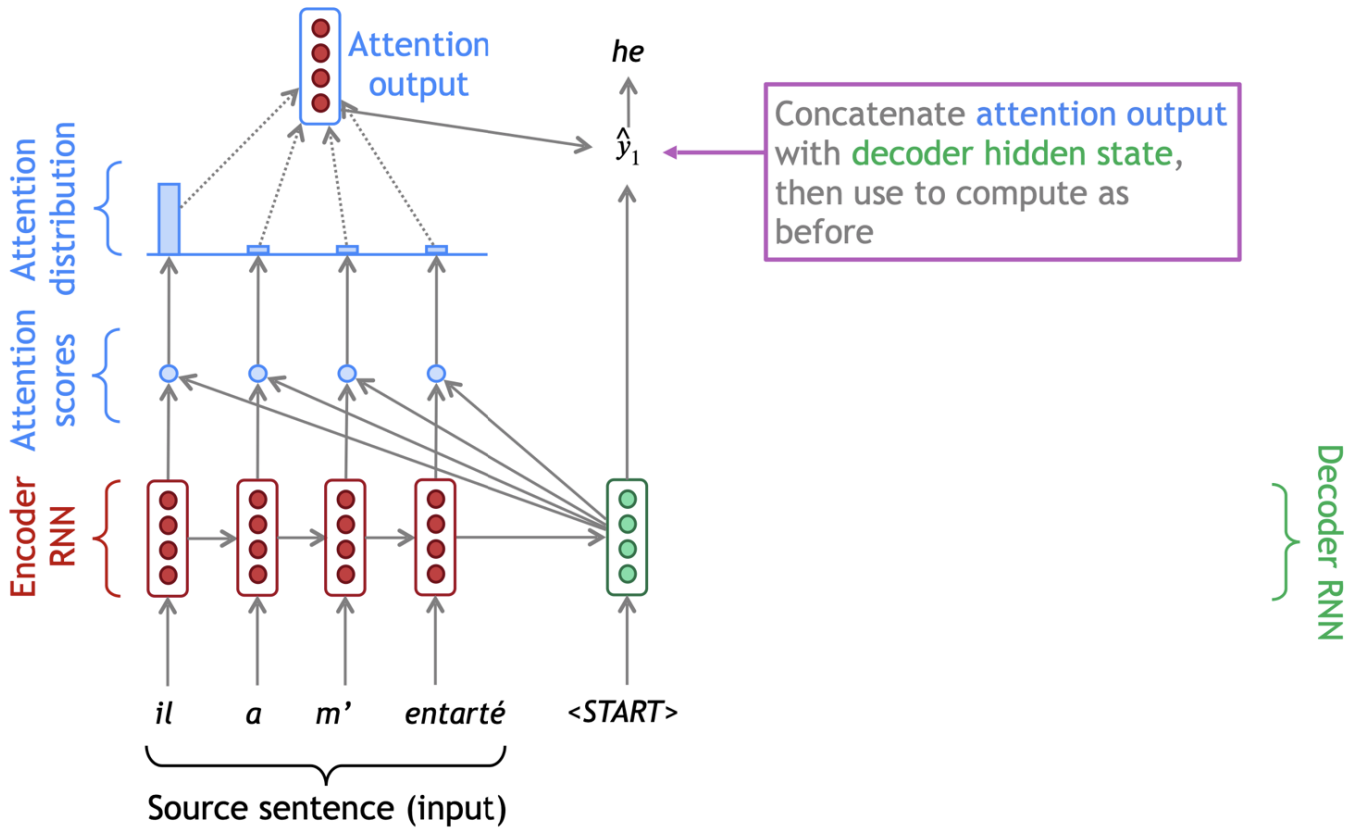
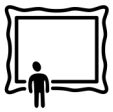


figure source: http://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture08-nmt.pdf

Attention



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

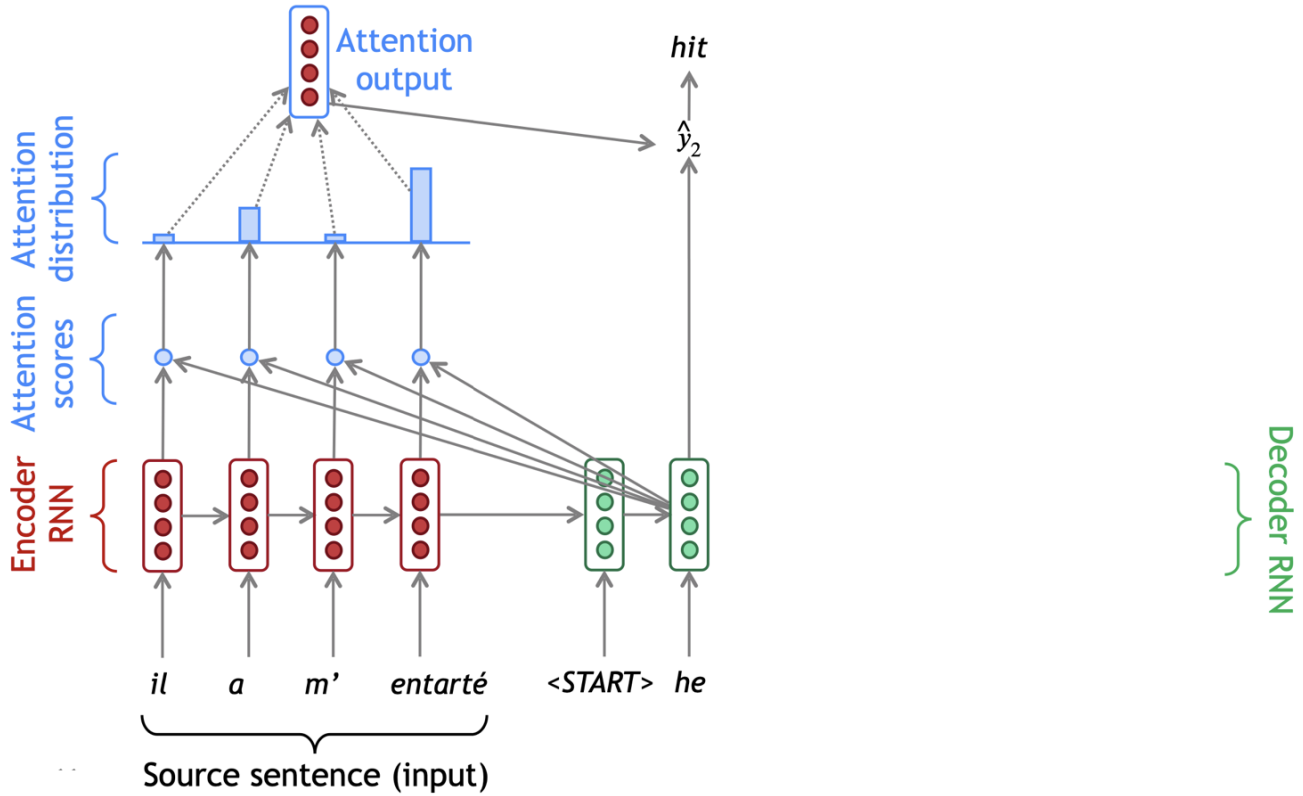
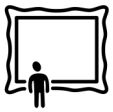


figure source: http://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture08-nmt.pdf

Attention



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

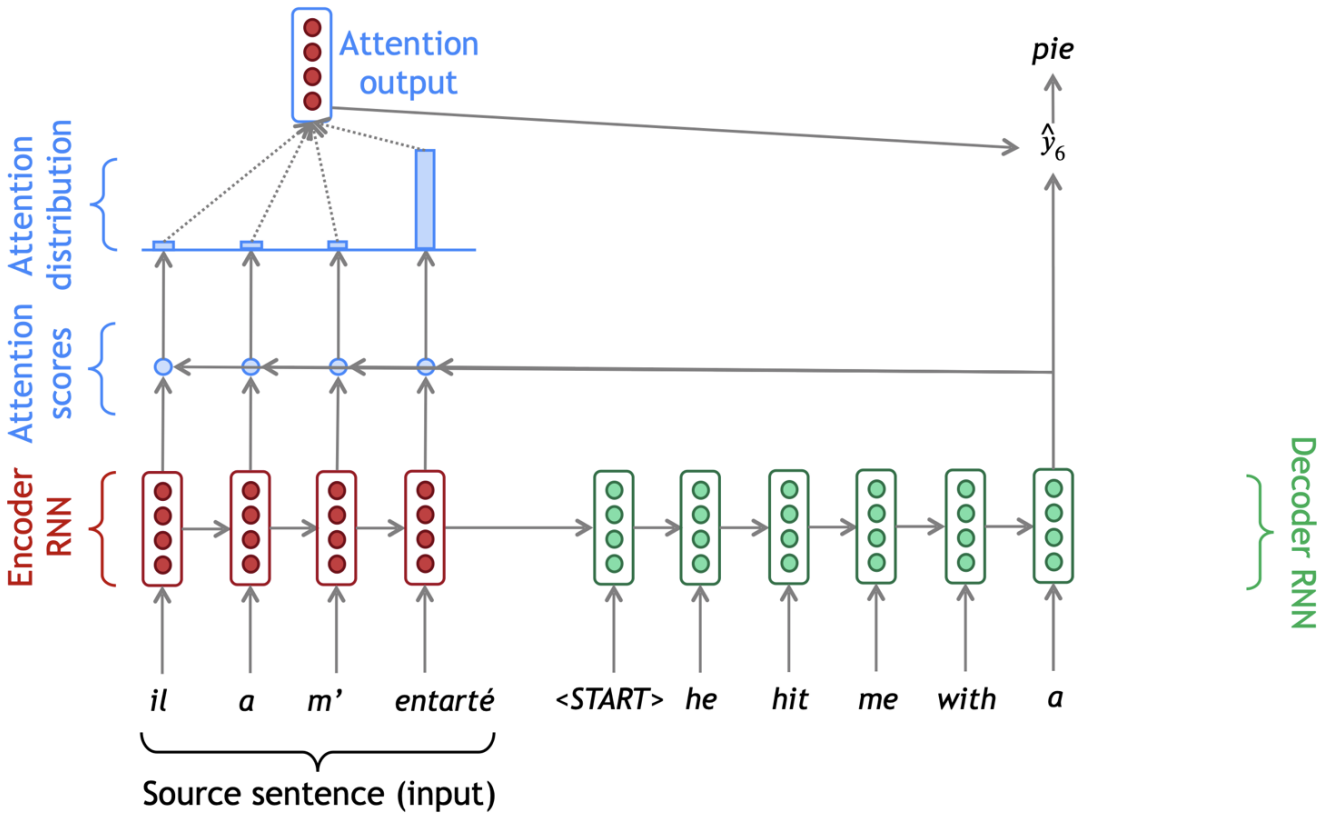
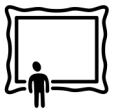


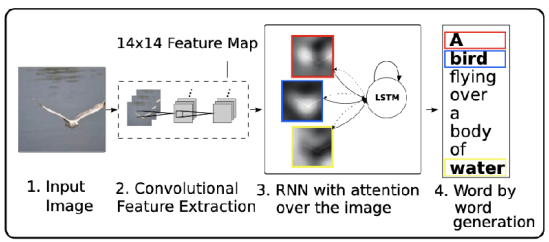
figure source: http://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture08-nmt.pdf

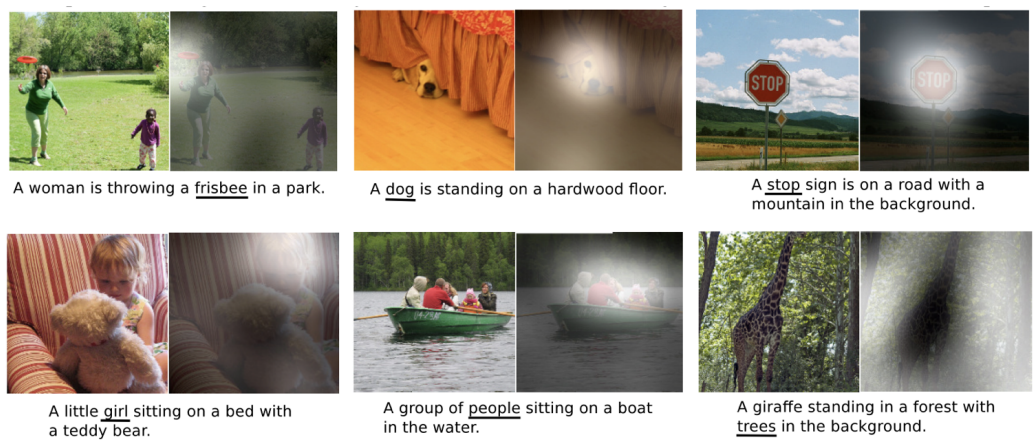
Attention



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

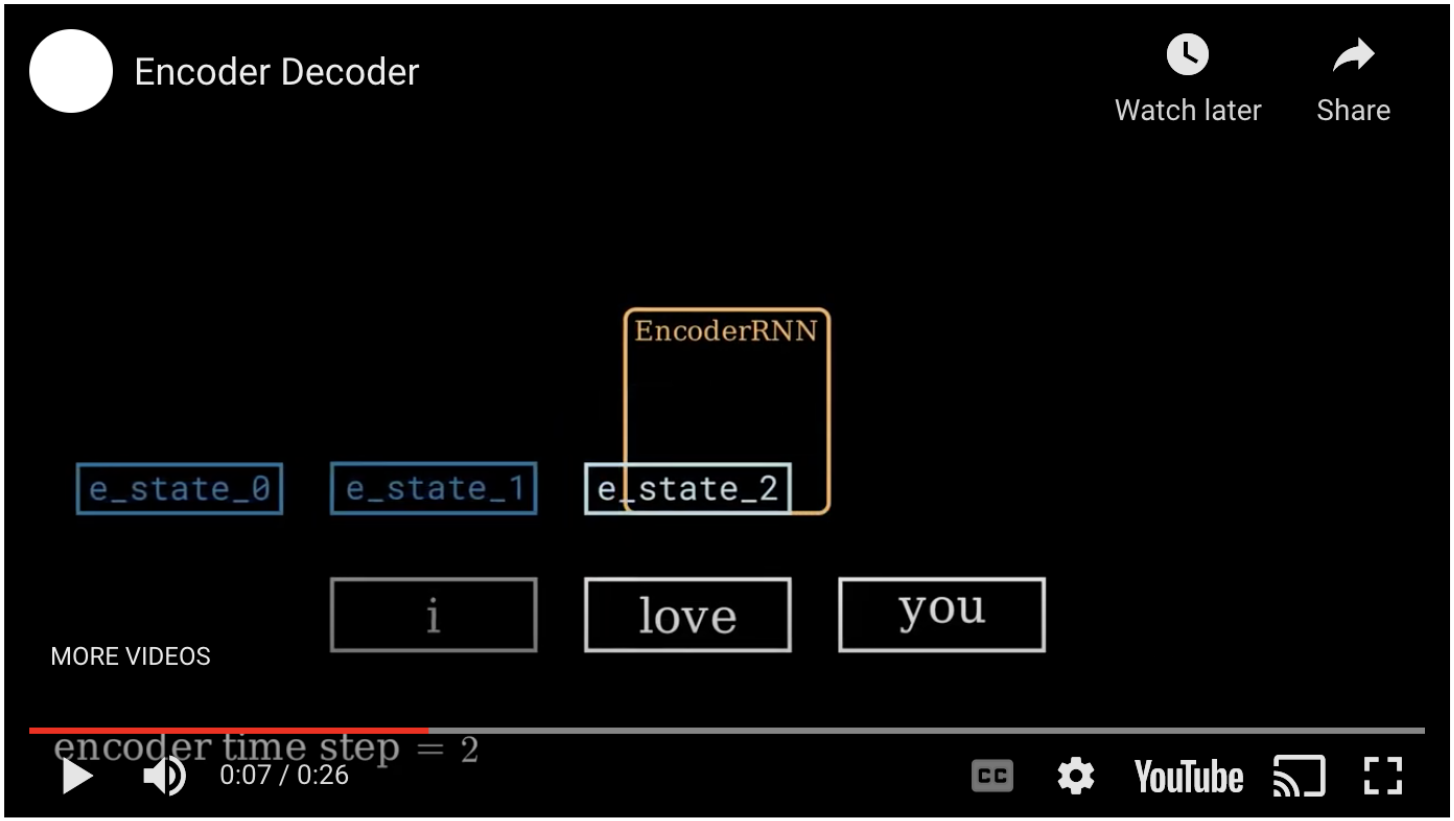
Attention can also be used for other generative models





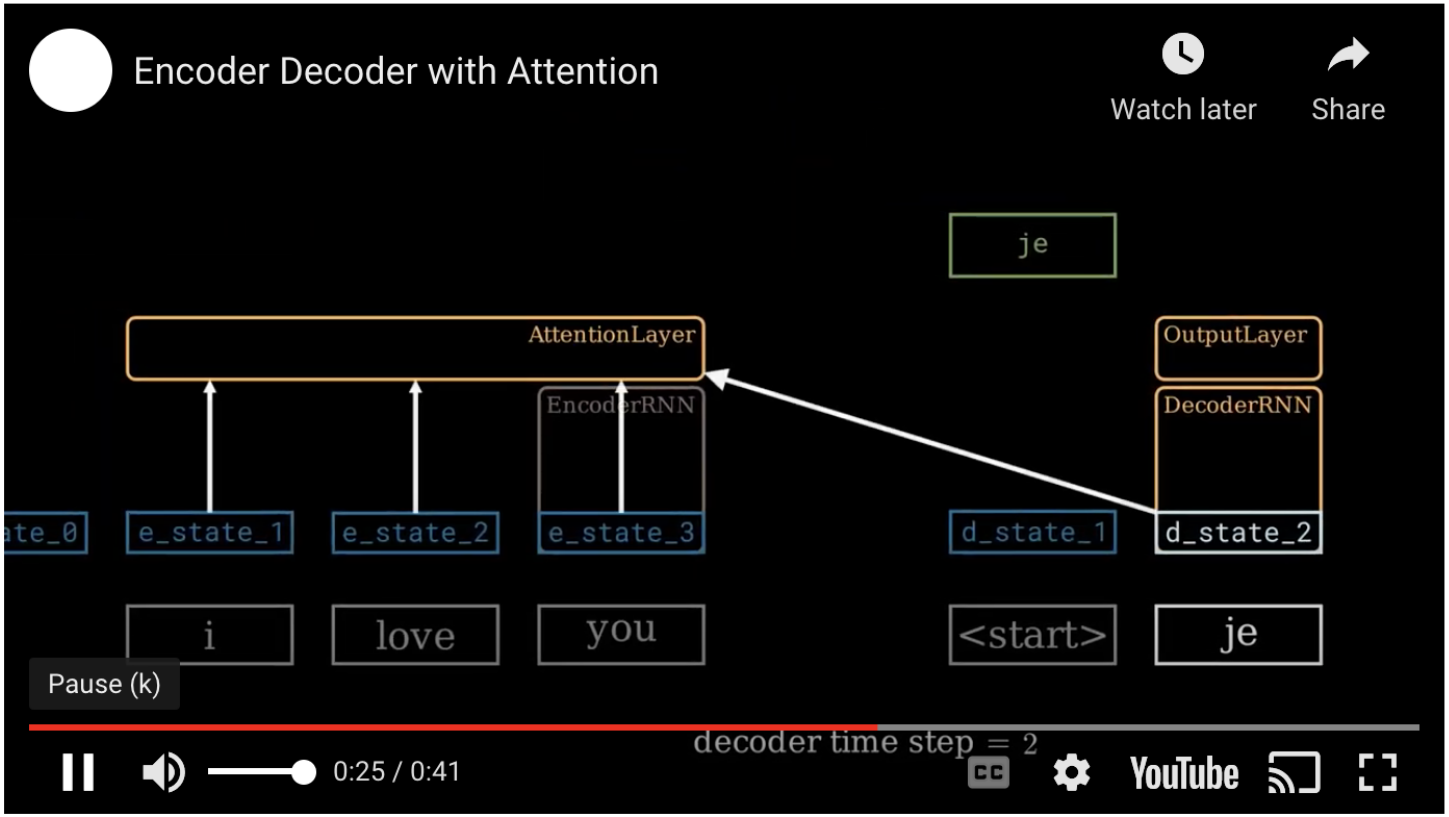
* "Show, attend and tell: neural image caption generation with visual attention", Xu et al. 2016, https://arxiv.org/pdf/1502.03044.pdf

Animated Attention



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.

Animated Attention



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

---

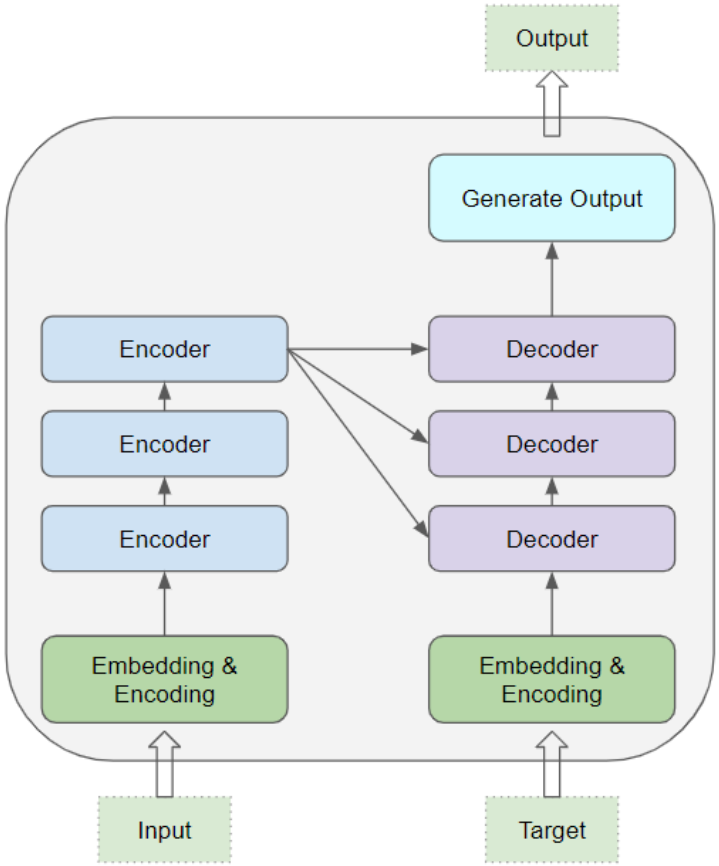
https://medium.com/@cakraww/from-recurrent-neural-network-rnn-to-attention-explained-intuitively-7fd7caeef6af

Transformers

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 distance between words matters less

Attention: example



https://towardsdatascience.com/transformers-explained-visually-part-1-overview-of-functionality-95a6dd460452

Attention: example

https://towardsdatascience.com/transformers-explained-visually-part-1-overview-of-functionality-95a6dd460452

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