S&DS 365 / 665

Intermediate Machine Learning

Course wrap up

April 27

Yale

Reminders

- Assn 4 due tonight
- Final exam: Saturday May 7 at 2pm in SPL 59
- Practice exam by Monday May 2
- Three review sessions TBA

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

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

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.

Codex Private beta

The Codex models are descendants of our GPT-3 models that can understand and generate code. Their training data contains both natural language and billions of lines of public code from GitHub. Learn more.

They're most capable in Python and proficient in over a dozen languages including JavaScript, Go, Perl, PHP, Ruby, Swift, TypeScript, SQL, and even Shell.

We currently offer two Codex models:

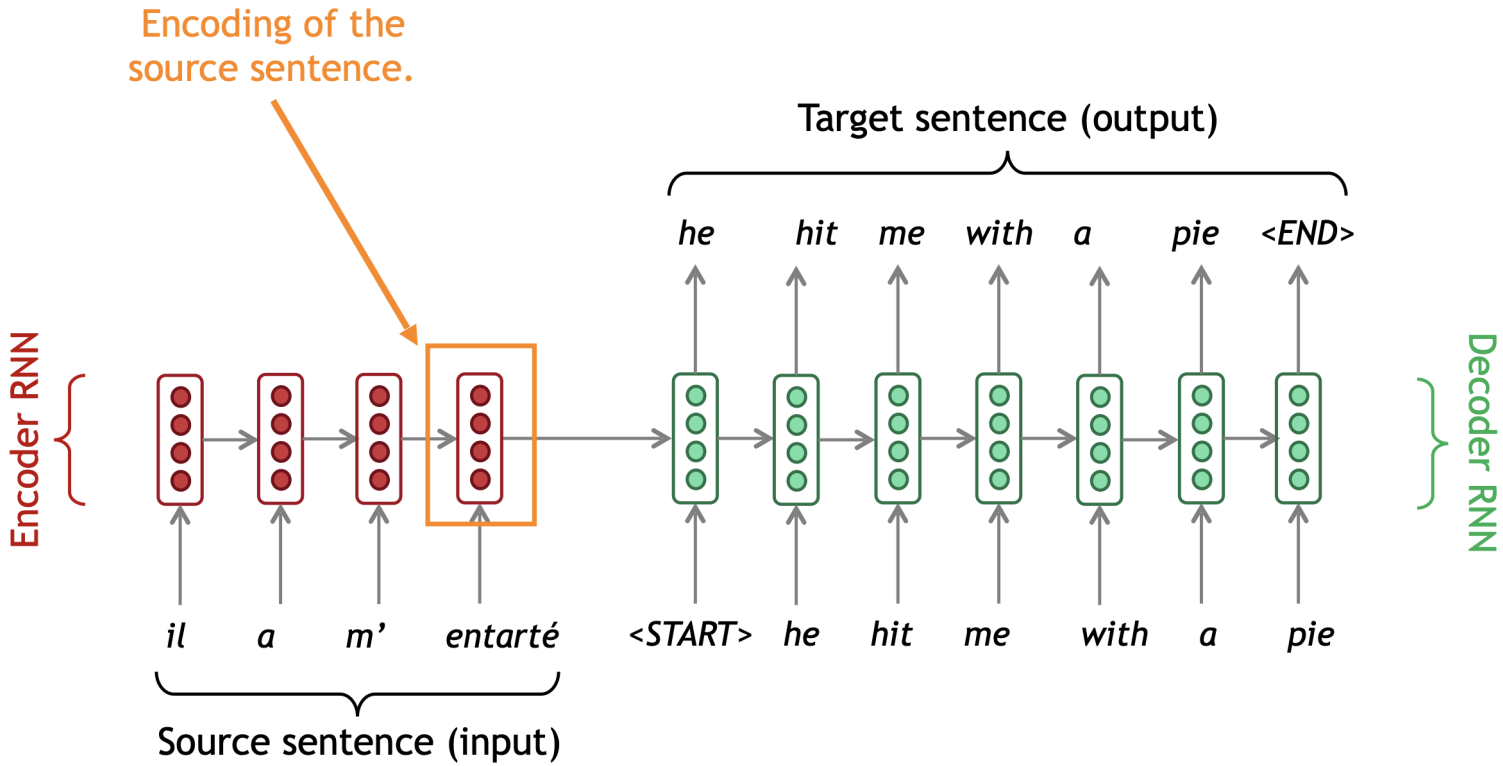
LATEST ENGINE	DESCRIPTION	MAX REQUEST	TRAINING DATA
`code-davinci-002`	Most capable Codex model. Particularly good at translating natural language to code. In addition to completing code, also supports inserting completions within code.	4,000 tokens	Up to Jun 2021
`code-cushman-001`	Almost as capable as Davinci Codex, but slightly faster. This speed advantage may make it preferable for real-time applications.	Up to 2,048 tokens	

For more, visit our guide to working with Codex.

The Codex models are offered as a free trial. As we learn about use, we'll look to offer pricing to enable a broad set of applications.

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

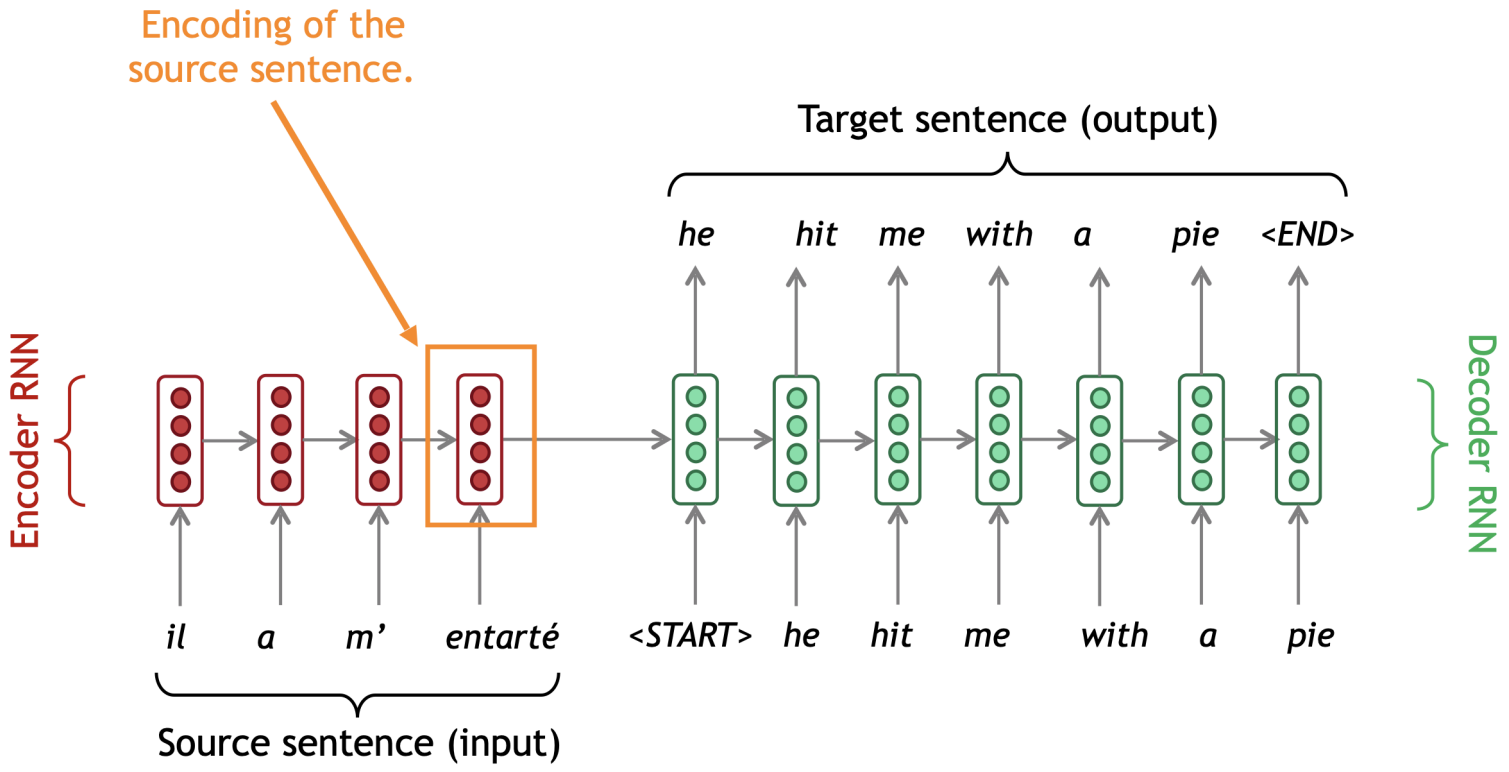
$$\prod_{t=1}^{T} p(y_t \mid h, y_1, \ldots, y_{t-1})$$

- 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.

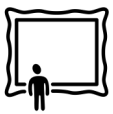
Sequence-to-sequence models

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

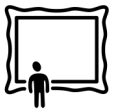


Important modification: Attention

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

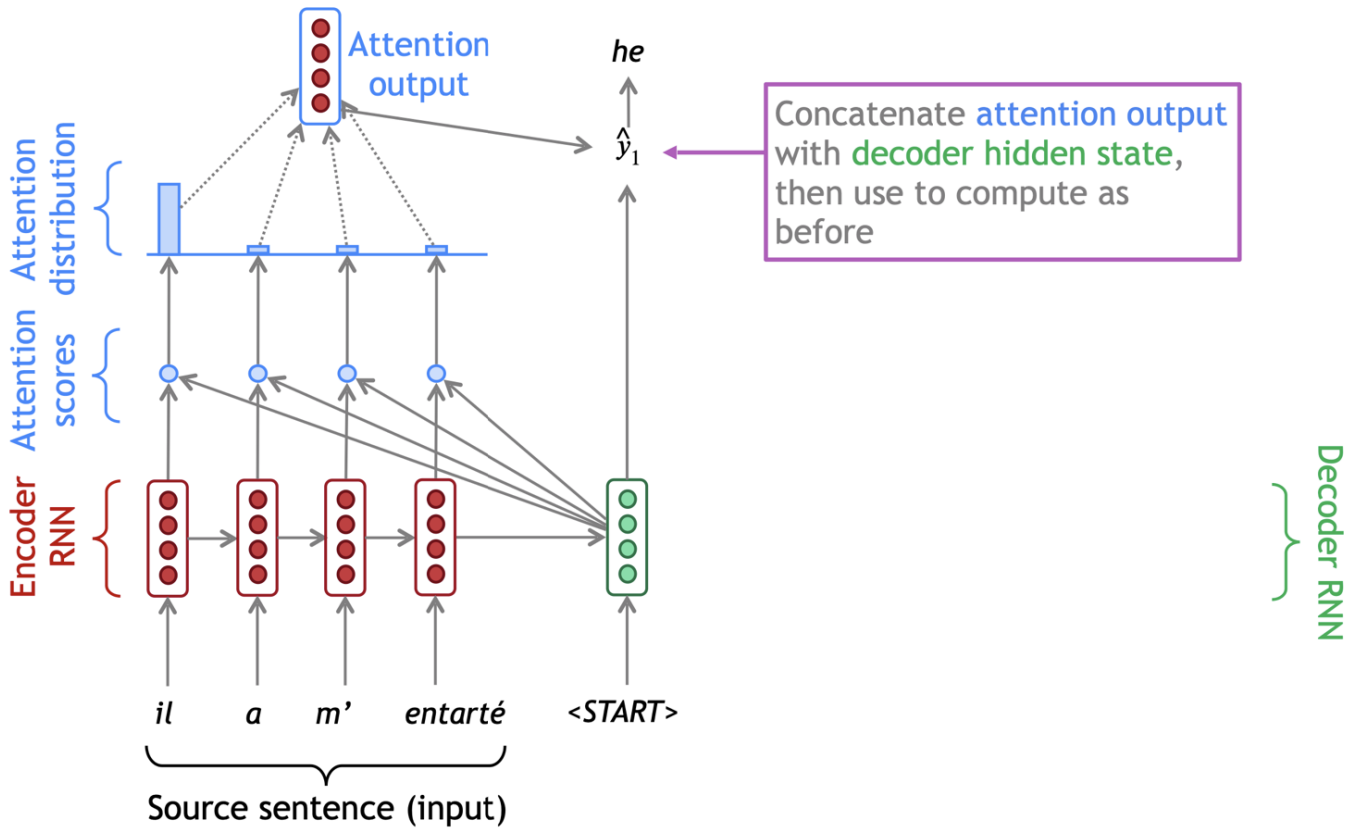
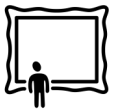


figure source: http://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture08-nmt.pdf

Attention



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

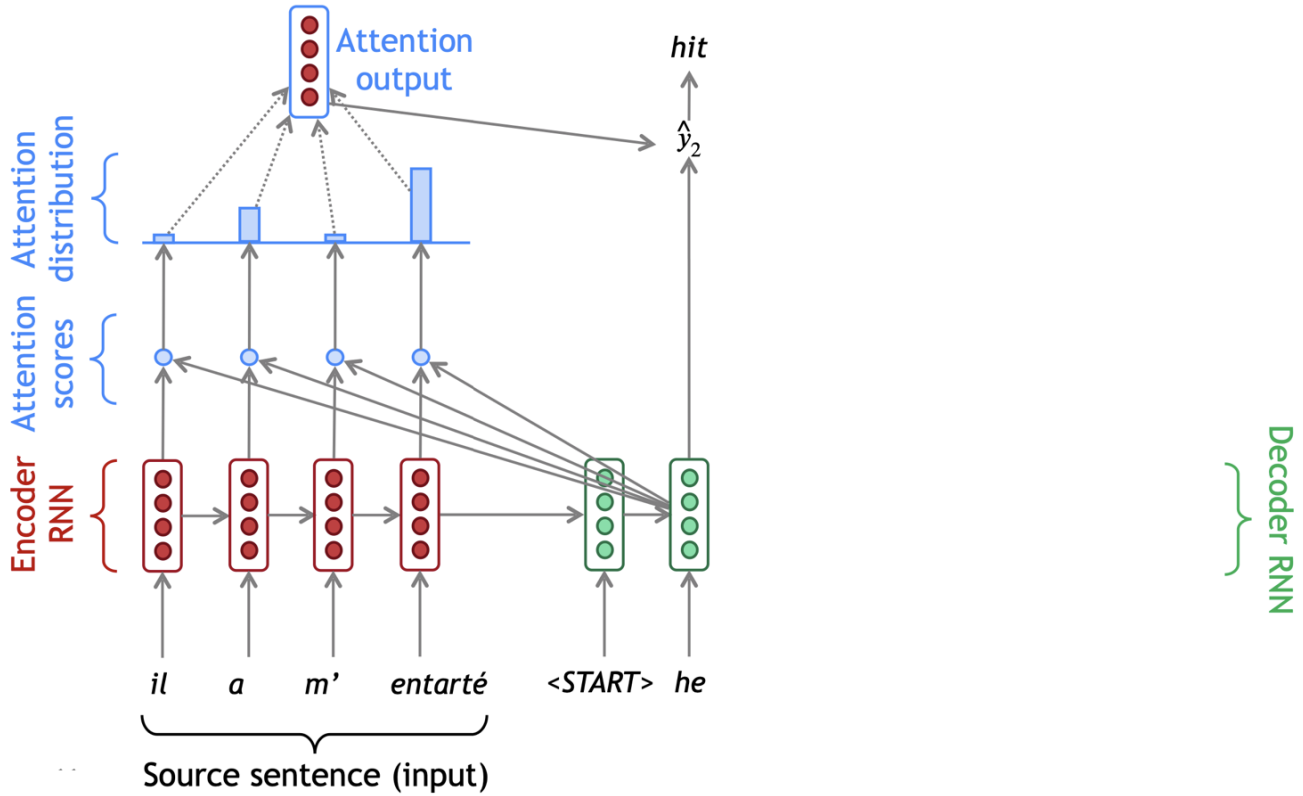
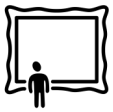


figure source: http://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture08-nmt.pdf

Attention



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

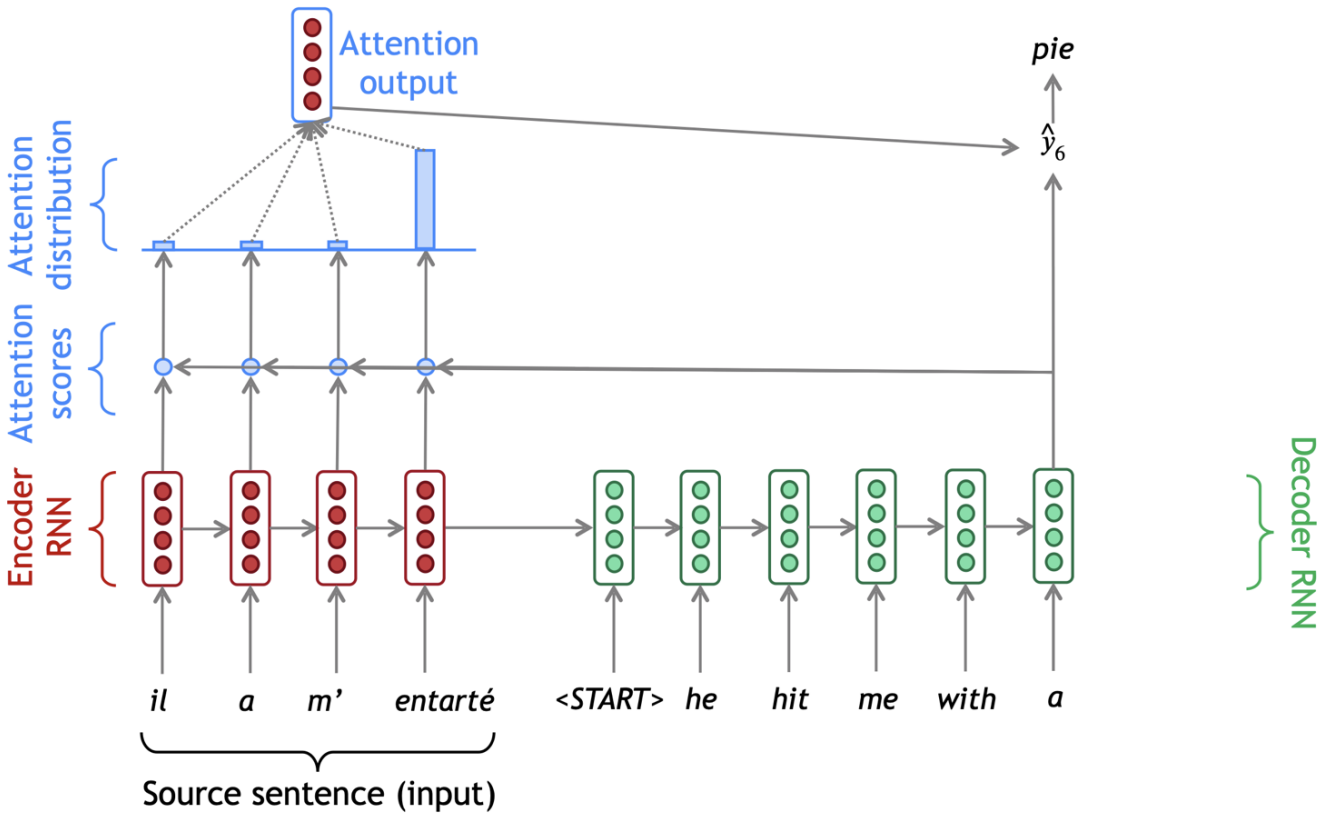
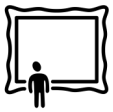


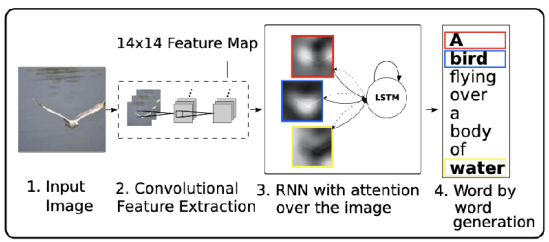
figure source: http://web.stanford.edu/class/cs224n/slides/cs224n-2020-lecture08-nmt.pdf

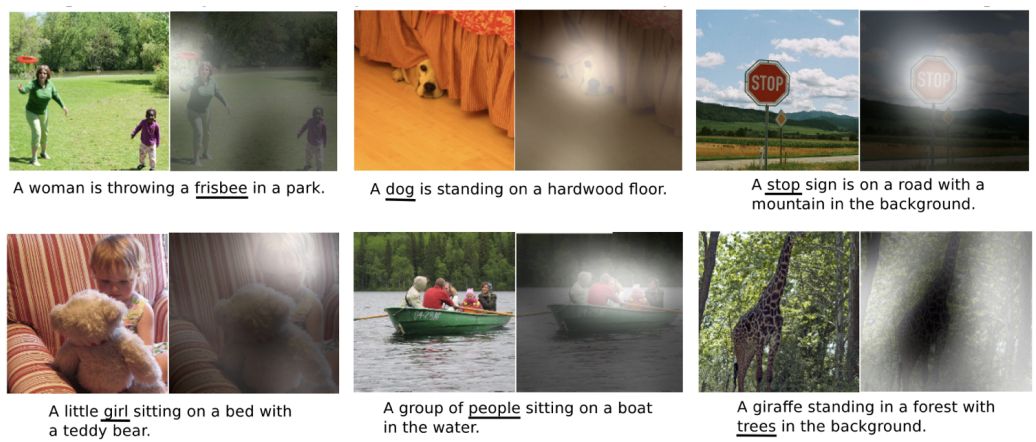
Attention



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

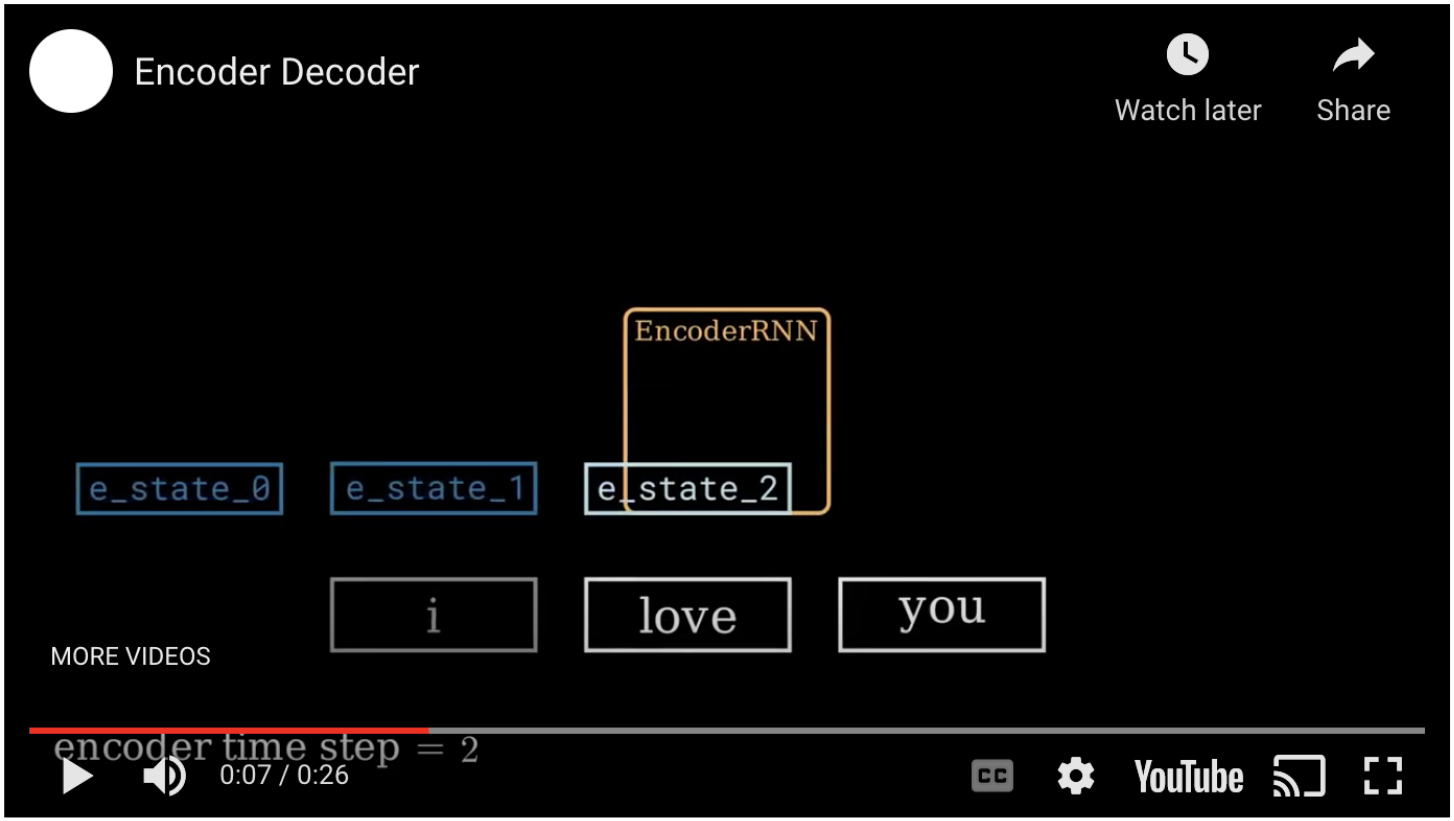
Attention can also be used for other generative models





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

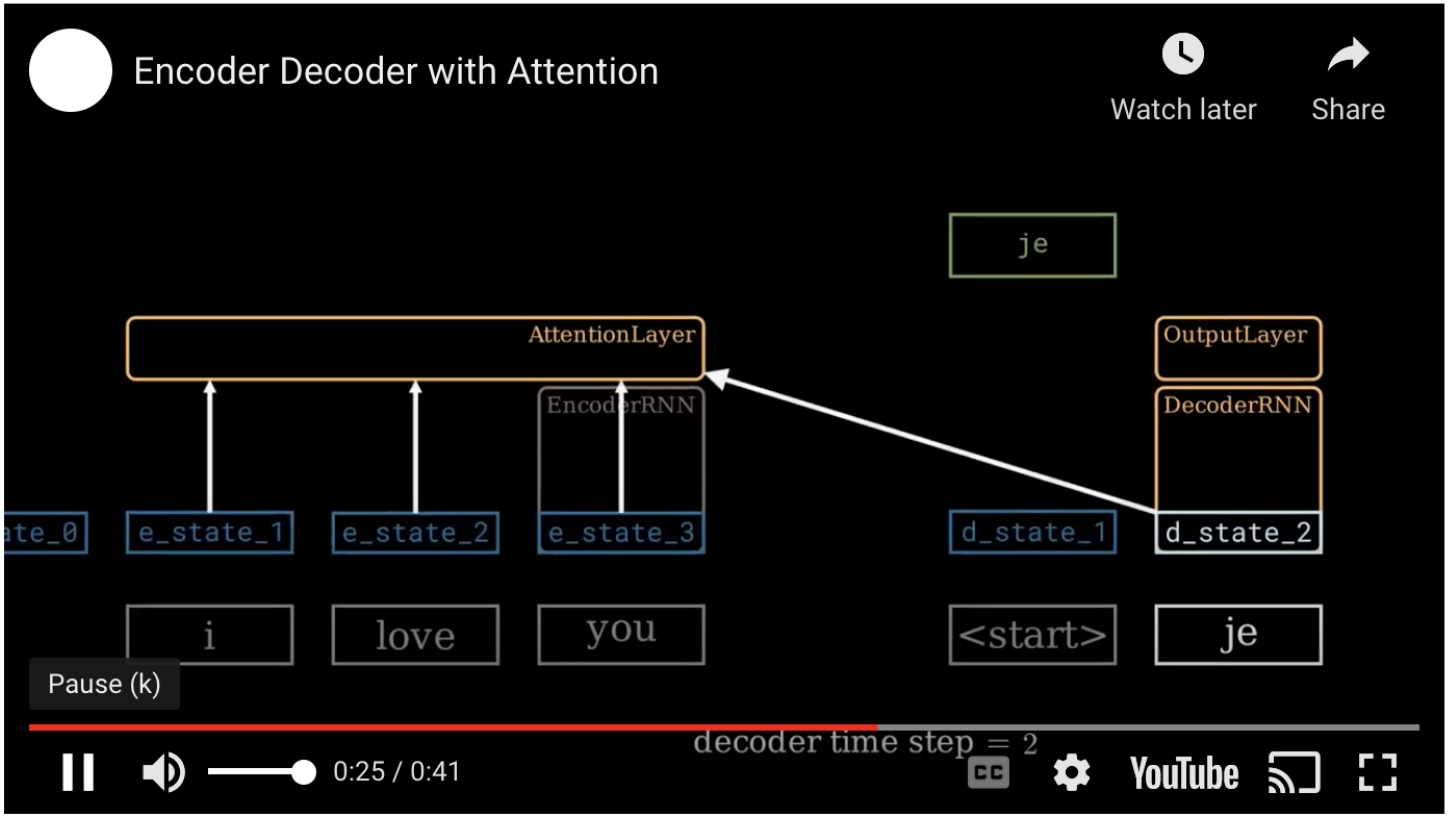
Attention animated



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.

https://medium.com/@sakrawy/from-recurrent-neural-network-rnn-to-attention-explained-intuitively-7fd7eaaef6af

Attention animated



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

---

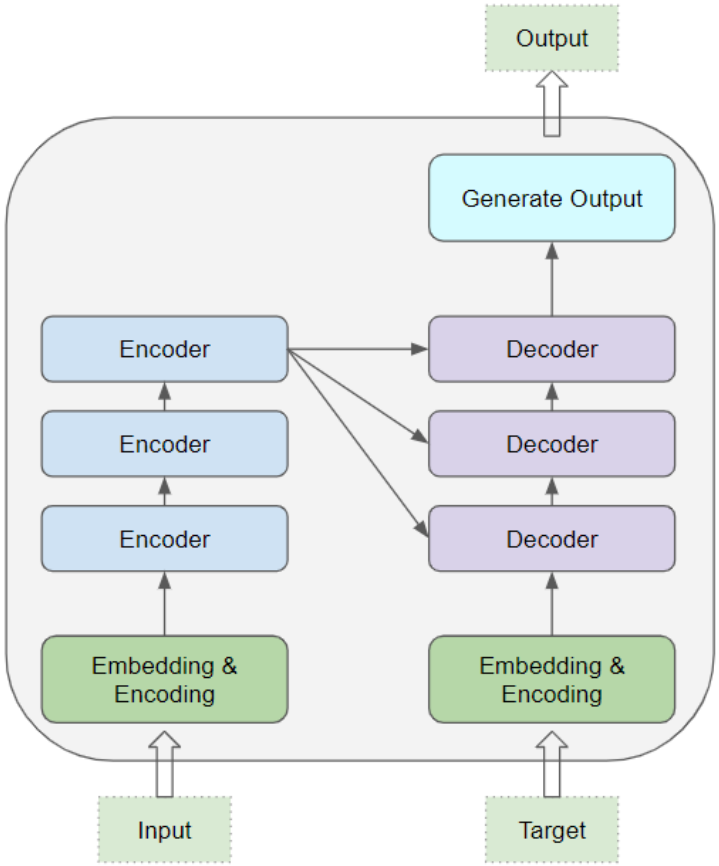
https://medium.com/@cakraww/from-recurrent-neural-network-rnn-to-attention-explained-intuitively-7fd7caeef6af

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

Reading

"Probabilistic Machine Learning: An Introduction" has an in-depth treatment of attention and transfomers

- Relates attention to kernel regression
- Describes other attention mechanisms
- Compares CNNs, RNNs, and transformers

Week	Dates	Topics	Demos & Tutorials	Lecture Slides	Readings & Notes	Assignments & Exams
1	Jan 26, 28	Course overview	Python elements Pandas and regression Lasso example	Jan 26: Course overview Jan 28: Sparse regression	PML Section 11.4	
2	Jan 31, Feb 2	Smoothing and kernels	Smoothing example Using different kernels Mercer kernels	Jan 31: Smoothing Feb 2: Mercer Kernels	PML Sections 16.3, 17.1 Notes on Mercer kernels	
3	Feb 7, 9	Density estimation and risk bounds	Density estimation demo	Feb 7, 9: Sync up	Bias-variance tradeoff for density estimation	Feb 9: Assn1 out
4	Feb 14, 16	Neural networks for classification	TensorFlow playground Convolution demo Problem 4 warmup	Feb 9: Neural networks Feb 14: Convolutional neural networks Feb 16: CNNs continued	PML Sections 13.1, 13.2 Notes on backpropagation	Feb 16: Quiz 1

5	Feb 21, 23	Nonparametric Bayes	Parametric Bayes Gaussian processes Dirichlet processes	Feb 21: Gaussian processes Feb 23: Gaussian and Dirichlet processes	PML Section 17.2 Notes on Bayesian inference Notes on nonparametric Bayes	Feb 23: Assn 1 in; Assn2 out
6	Feb 28, Mar 2	Gibbs sampling	DP demo, ver. 2 Gibbs sampling	Feb 28: Dirichlet processes Mar 2: Gibbs sampling	Notes on Gibbs sampling	Mar 2: Quiz 2
7	Mar 7, 9	Variational inference	Variational autoencoders	Mar 7: Introduction to approximate inference Mar 9: Variational inference and VAEs	PML Section 20.3 Notes on variational inference	Mar 9: Assn 2 in
8	Mar 14, 16	Review and midterm		Mar 14: VAEs and review Mar 16: Midterm	Practice midterm	Mar 16: Midterm exam

9	Mar 28, 30	Graphs and structure learning	Graphical lasso demo Graph neural networks	Mar 28: Sparsity and graphs Mar 30: Discrete data and graph neural nets	Notes on graphs and structure learning PML Section 23.4	Mar 30: Assn3 out
10	Apr 4, 6	Deep reinforcement learning	Q-learning demo	Apr 4: Reinforcement learning Apr 6: Deep reinforcement learning		Apr 6: Quiz 3
11	Apr 11, 13	Policy gradient methods	DQN demo Policy gradients demo Actor-critic demo	Apr 11: Policy gradient methods Apr 13: Actor-critic methods		Apr 13: Assn 3 in Assn4 out
12	Apr 18, 20	Sequential and sequence-to-sequence models	RNN demo: Fakespeare TensorFlow: Text generation	Apr 18: HMMs and RNNs Apr 20: RNNs, GRUs, LSTMs, and all that	PML Chapter 15	Apr 20: Quiz 4
13	Apr 25, 27	Attention and language models	GPT-3 demo	Apr 25: Sequence-to-sequence models and attention		Apr 27: Assn 4 in
	May 7	Final exam, 2pm in SPL 59				Registrar: final exam schedule

Final exam

- Final exam Saturday, May 7, 2022 at 2pm in SPL 59
- `https://registrar.yale.edu/general-information/final-exams`
- Review sessions, time and place TBA
- Length: About 1.5X Midterm
- Emphasis on material after midterm
- Any topic could be on exam...except

Vote a topic off the exam!

Nominations?

Your input

- Please complete a course review!
- I value your comments and feedback
- Feel free to send me comments privately
- Let me know how you use ML!

Thank you!