Maior Dúvida da Aula

1. Não consigo entender a essência de transferência de conhecimento entre redes, para mim não faz sentido treinarmos um modelo para executar certa tarefa e ele ficar bom em resolver outra tarefa, não entra na minha cabeça isso.

2. Existe algum caso de sucesso onde se fez transfer learning para uma atividade intermediária, antes do fine-tuning na tarefa alvo, resultando em um melhor modelo?

3. O transfer learning resolve o problema de usar um "mesmo modelo" treinado em um hospital (por exemplo) em outro? Apenas fazendo um fine-tuning fino, já que esse tipo de problema generaliza mal.

4. Redes já treinadas são ainda mais difíceis de serem interpretadas, ou existem áreas de estudos que colocam esforços para entender o poder de generalização dessas ferramentas?

5. Existe algum método para verificar se a arquitetura que queremos retreinar (transfer learning) é passível de receber imagens com dimensões diferentes àquelas utilizadas no treinamento? Por exemplo: treinamento foi efetuado com imagens (3, 224, 224), porém deseja-se utilizar imagens (3, 336, 336).

6. Sobre o projeto: pretendemos usar uma segunda base de dados para testar a rede num contexto bastante diferente e ver como ela generaliza. Isso é um bom método para encontrar e testar exemplos 'difíceis'?

7. Não entendi como a tarefa pretexto funciona. Ela não é uma anotação igual a do label que eu pretendo utilizar no meu algoritmo?

8. Ainda estou um pouco confuso a respeito do processo de self supervised learning. Precisamos fazer um fine tuning da rede após treiná-la com a tarefa inicial?

9. Como decidir entre usar self-learning ou pré-treinamento à moda antiga?

10. Podemos utilizar o aprendizado auto supervisionado com dados tabulares?

11. Com o self-supervised learning nossos problemas de dados anotados acabaram.

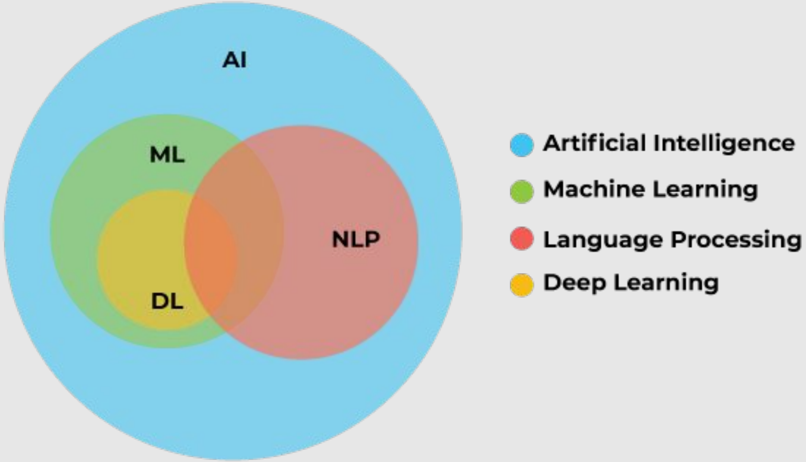
RNNs & Transformers

Machine Learning

Prof. Sandra Avila
Institute of Computing (IC/Unicamp)

MC886/MO444, November 10, 2022

Natural Language Processing



➡ Bag of Words

↓

RNNs

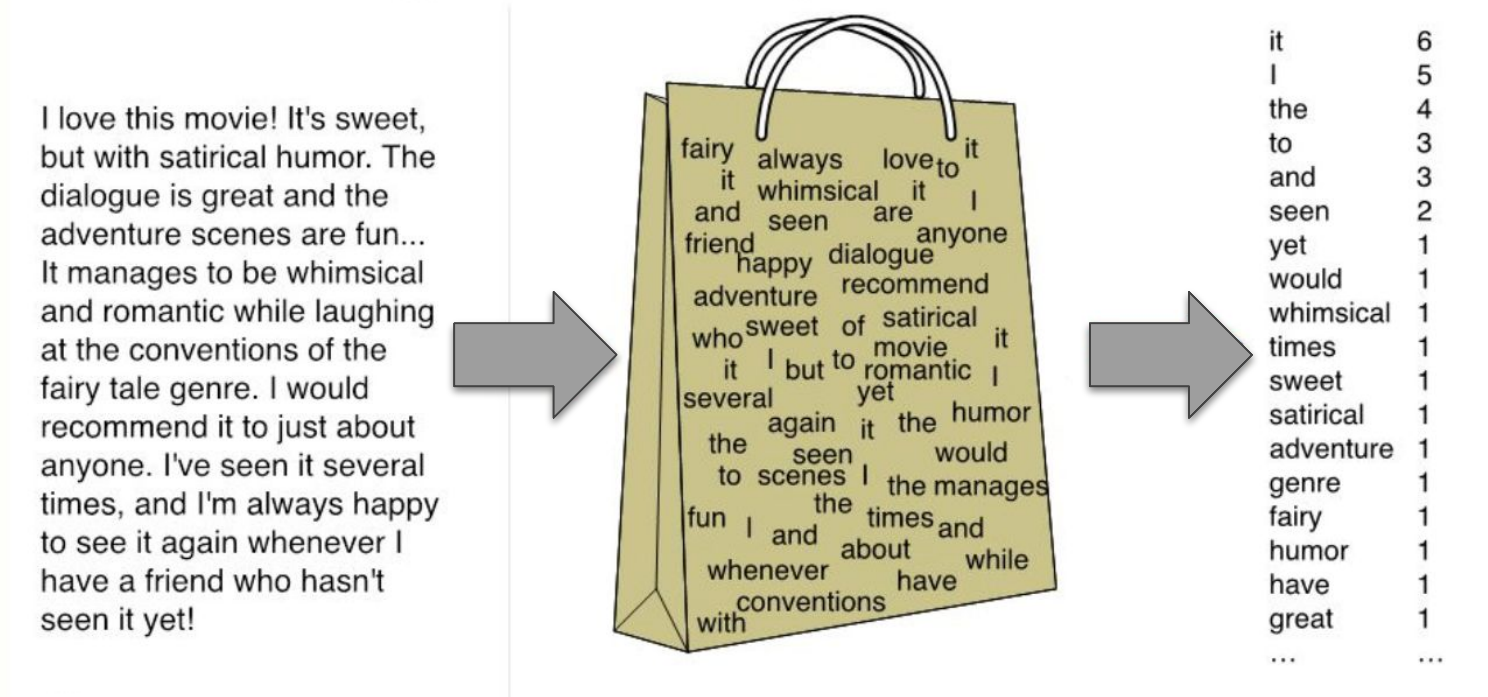
↓

LSTMs

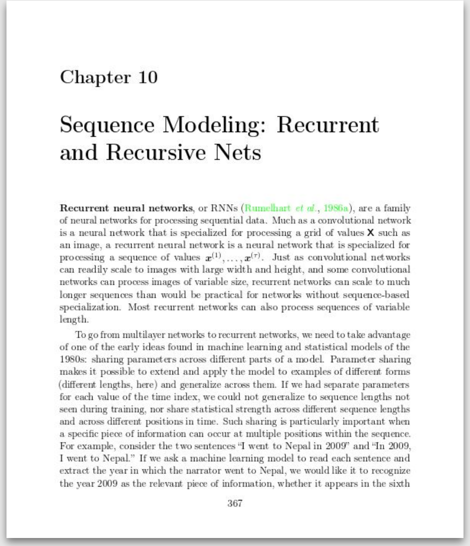
↓

Transformers

Bag of Words


I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!
fairy always love to it
it whimsical it I
and seen are anyone
friend happy dialogue
adventure recommend
who sweet of satirical
movie it
it I but to romantic I
several yet
again it the humor
the seen would
to scenes I the manages
fun I and the times and
about while
whenever have
with conventions
it 6
I 5
the 4
to 3
and 3
seen 2
yet 1
would 1
whimsical 1
times 1
sweet 1
satirical 1
adventure 1
genre 1
fairy 1
humor 1
have 1
great 1
... ...

Bag of Words



Chapter 10

Sequence Modeling: Recurrent and Recursive Nets

Recurrent neural networks, or RNNs (Rumelhart *et al.*, 1986a), are a family of neural networks for processing sequential data. Much as a convolutional network is a neural network that is specialized for processing a grid of values $\mathbf{X}$ such as an image, a recurrent neural network is a neural network that is specialized for processing a sequence of values $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\tau)}$. Just as convolutional networks can readily scale to images with large width and height, and some convolutional networks can process images of variable size, recurrent networks can scale to much longer sequences than would be practical for networks without sequence-based specialization. Most recurrent networks can also process sequences of variable length.

To go from multilayer networks to recurrent networks, we need to take advantage of one of the early ideas found in machine learning and statistical models of the 1980s: sharing parameters across different parts of a model. Parameter sharing makes it possible to extend and apply the model to examples of different forms (different lengths, here) and generalize across them. If we had separate parameters for each value of the time index, we could not generalize to sequence lengths not seen during training, nor share statistical strength across different sequence lengths and across different positions in time. Such sharing is particularly important when a specific piece of information can occur at multiple positions within the sequence. For example, consider the two sentences "I went to Nepal in 2009" and "In 2009, I went to Nepal." If we ask a machine learning model to read each sentence and extract the year in which the narrator went to Nepal, we would like it to recognize the year 2009 as the relevant piece of information, whether it appears in the sixth

367

multilayer	10
network	20
advantage	2
learning	4
parameter	0
model	0
generalize	0

10
20
2
4
0
0
0

Bag of Words

Problems? **Sparse data**

Order matters!

“work to live” vs. “live to work”

Bag of Words

↓

RNNs

↓

LSTMs

↓

Transformers

Recurrent Neural Network

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

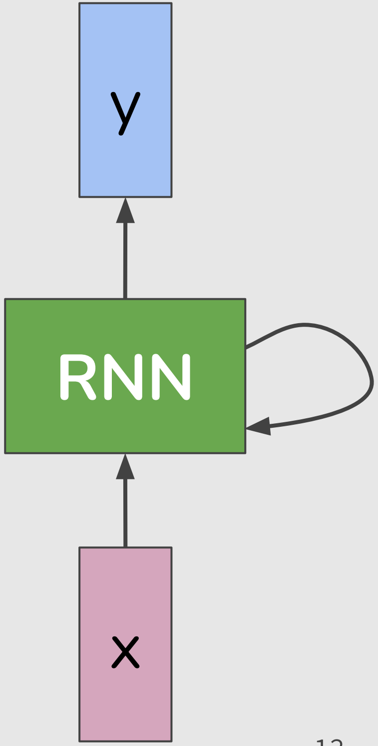
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

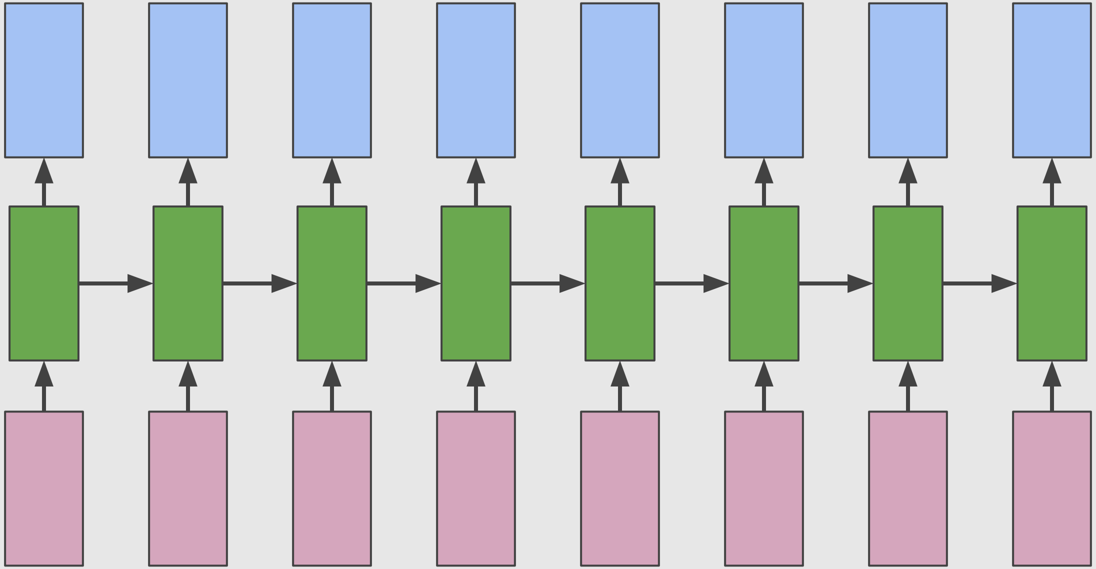
old state

input vector at some time step

some function with parameters W



Recurrent Neural Network



RNNs: Process Sequences

one to one

Vanilla Neural Networks

one to many

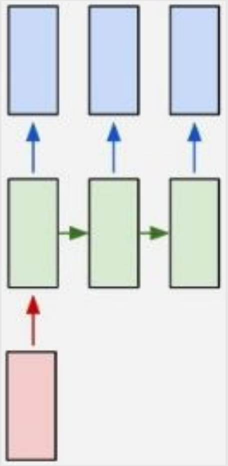


Image Captioning
image ⇒ seq. words

Image Captioning

No errors

A white teddy bear sitting in the grass

A man riding a wave on top of a surfboard

Minor errors

A man in baseball uniform throwing a ball

A cat sitting on a suitcase on the floor

Somewhat related

A woman is holding a cat in her hand

A woman standing on a beach holding a surfboard

RNNs: Process Sequences

one to one

Vanilla Neural Networks

one to many

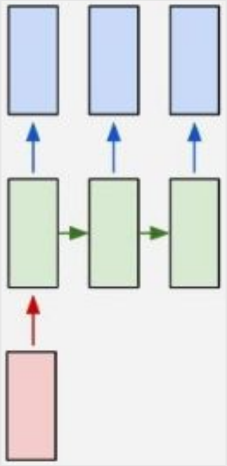
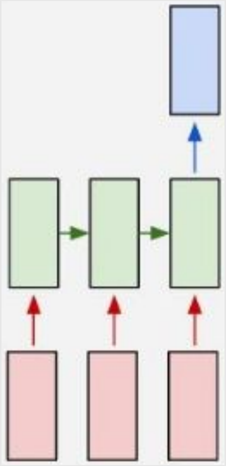


Image Captioning
image ⇒ seq. words

many to one



Sentiment Classification
seq. words ⇒ sentiment

Visual Question Answering (VQA)

COCOQA 33827
What is the color of the cat?
Ground truth: black
IMG+BOW: black (0.55)
2-VIS+LSTM: black (0.73)
BOW: gray (0.40)

COCOQA 33827a
What is the color of the couch?
Ground truth: red
IMG+BOW: red (0.65)
2-VIS+LSTM: black (0.44)
BOW: red (0.39)

DAQUAR 1522
How many chairs are there?
Ground truth: two
IMG+BOW: four (0.24)
2-VIS+BLSTM: one (0.29)
LSTM: four (0.19)

DAQUAR 1520
How many shelves are there?
Ground truth: three
IMG+BOW: three (0.25)
2-VIS+BLSTM: two (0.48)
LSTM: two (0.21)

COCOQA 14855
Where are the ripe bananas sitting?
Ground truth: basket
IMG+BOW: basket (0.97)
2-VIS+BLSTM: basket (0.58)
BOW: bowl (0.48)

COCOQA 14855a
What are in the basket?
Ground truth: bananas
IMG+BOW: bananas (0.98)
2-VIS+BLSTM: bananas (0.68)
BOW: bananas (0.14)

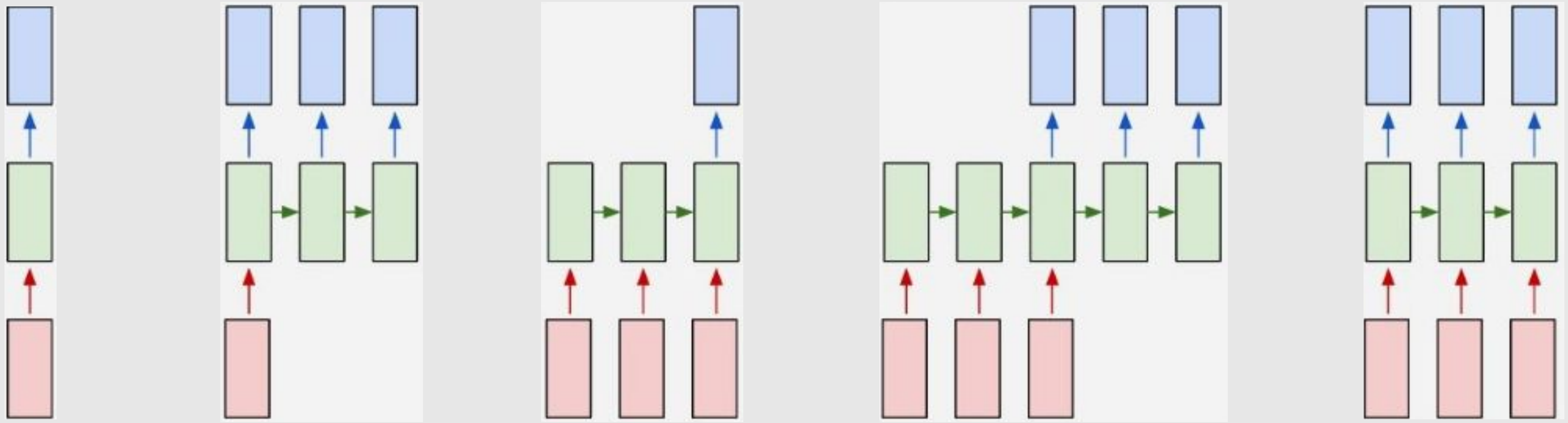
DAQUAR 585
What is the object on the chair?
Ground truth: pillow
IMG+BOW: clothes (0.37)
2-VIS+BLSTM: pillow (0.65)
LSTM: clothes (0.40)

DAQUAR 585a
Where is the pillow found?
Ground truth: chair
IMG+BOW: bed (0.13)
2-VIS+BLSTM: chair (0.17)
LSTM: cabinet (0.79)

Ren et al., “Exploring Models and Data for Image Question Answering”

RNNs: Process Sequences

one to one | one to many | many to one | many to many | many to many



Vanilla Neural Networks

Image Captioning
image ⇒ seq. words

Sentiment Classification
seq. words ⇒ sentiment

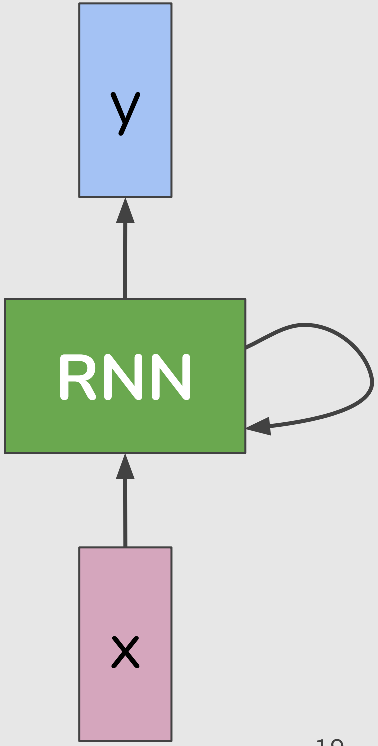
Machine Translation
seq. words ⇒ seq. of words

Video classification on frame level

Recurrent Neural Network

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

$$h_t = f_W(h_{t-1}, x_t)$$



Notice: the same function and the same set of parameters are used at every time step.

Recurrent Neural Network

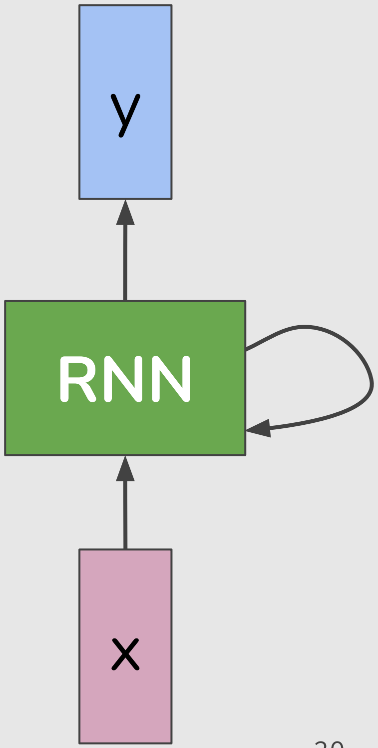
The state consists of a single "hidden" vector **h**:

$$h_t = f_W(h_{t-1}, x_t)$$

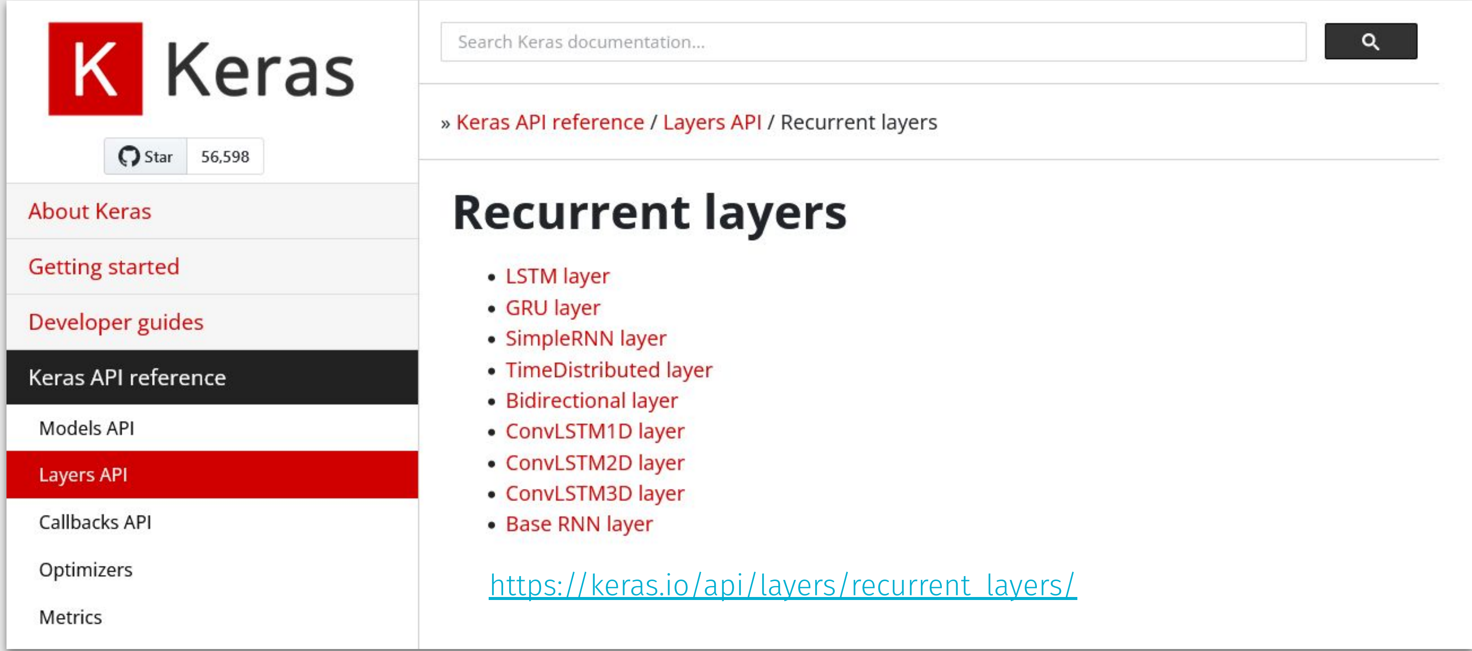
$$\Downarrow$$

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

$$y_t = W_{hy} h_t$$

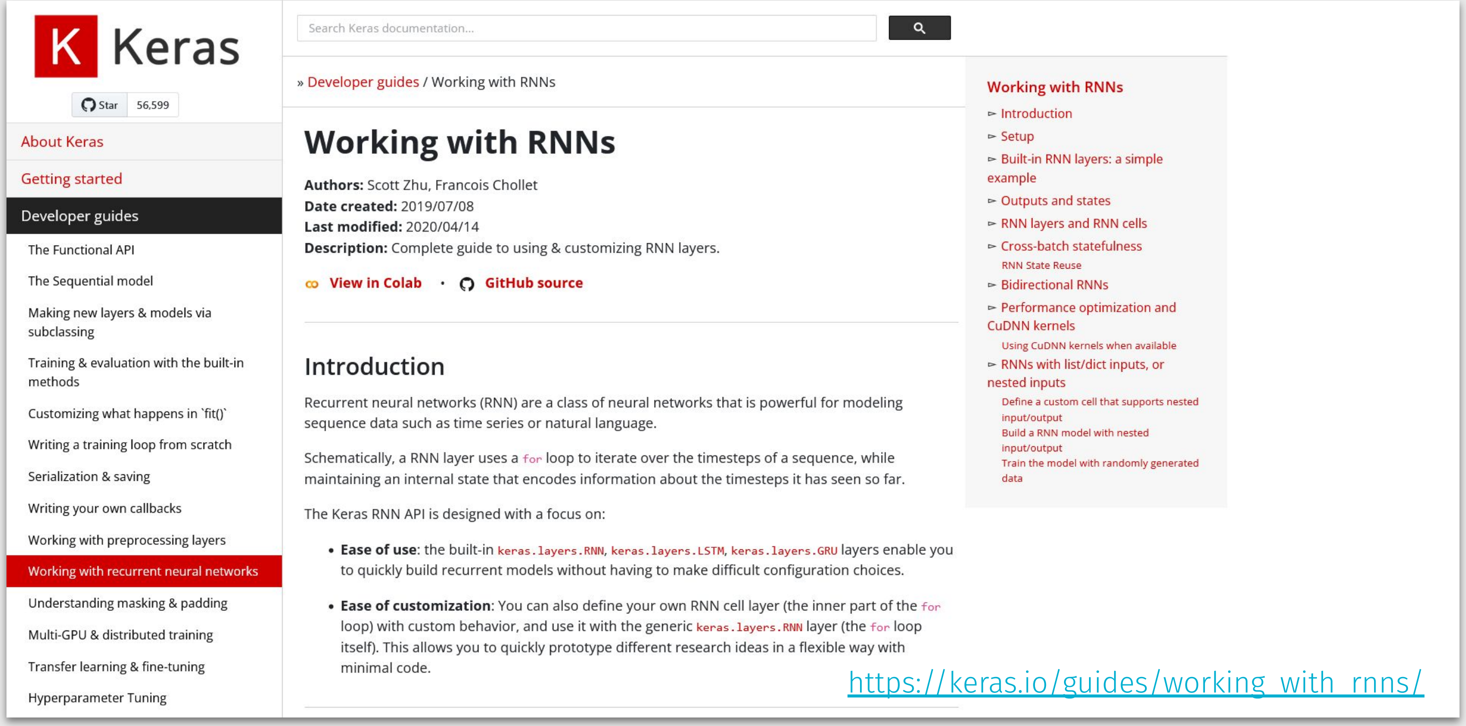


Recurrent Neural Network



https://keras.io/api/layers/recurrent_layers/

Recurrent Neural Network



Keras

Star 56,599

About Keras

Getting started

Developer guides

- The Functional API
- The Sequential model
- Making new layers & models via subclassing
- Training & evaluation with the built-in methods
- Customizing what happens in `fit()`
- Writing a training loop from scratch
- Serialization & saving
- Writing your own callbacks
- Working with preprocessing layers
- Working with recurrent neural networks
- Understanding masking & padding
- Multi-GPU & distributed training
- Transfer learning & fine-tuning
- Hyperparameter Tuning

Search Keras documentation...

» Developer guides / Working with RNNs

Working with RNNs

Authors: Scott Zhu, Francois Chollet
Date created: 2019/07/08
Last modified: 2020/04/14
Description: Complete guide to using & customizing RNN layers.

View in Colab • **GitHub source**

Introduction

Recurrent neural networks (RNN) are a class of neural networks that is powerful for modeling sequence data such as time series or natural language.

Schematically, a RNN layer uses a `for` loop to iterate over the timesteps of a sequence, while maintaining an internal state that encodes information about the timesteps it has seen so far.

The Keras RNN API is designed with a focus on:

- **Ease of use**: the built-in `keras.layers.RNN`, `keras.layers.LSTM`, `keras.layers.GRU` layers enable you to quickly build recurrent models without having to make difficult configuration choices.
- **Ease of customization**: You can also define your own RNN cell layer (the inner part of the `for` loop) with custom behavior, and use it with the generic `keras.layers.RNN` layer (the `for` loop itself). This allows you to quickly prototype different research ideas in a flexible way with minimal code.

Working with RNNs

- Introduction
- Setup
- Built-in RNN layers: a simple example
- Outputs and states
- RNN layers and RNN cells
- Cross-batch statefulness
  - RNN State Reuse
- Bidirectional RNNs
- Performance optimization and CuDNN kernels
  - Using CuDNN kernels when available
- RNNs with list/dict inputs, or nested inputs
  - Define a custom cell that supports nested input/output
  - Build a RNN model with nested input/output
  - Train the model with randomly generated data

https://keras.io/guides/working_with_rnns/

Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models)

Translations: Chinese (Simplified), Japanese, Korean, Russian

Watch: MIT's Deep Learning State of the Art lecture referencing this post

May 25th update: New graphics (RNN animation, word embedding graph), color coding, elaborated on the final attention example.

Note: The animations below are videos. Touch or hover on them (if you're using a mouse) to get play controls so you can pause if needed.

Sequence-to-sequence models are deep learning models that have achieved a lot of success in tasks like machine translation, text summarization, and image captioning. Google Translate started using such a model in production in late 2016. These models are explained in the two pioneering papers (Sutskever et al., 2014, Cho et al., 2014).

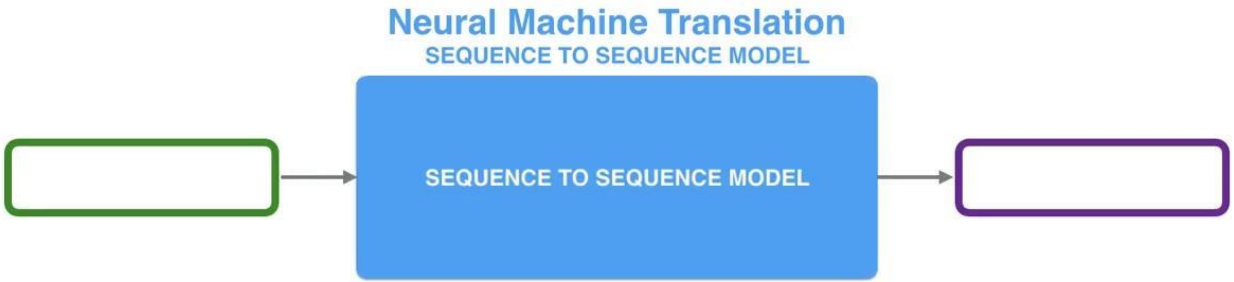
https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models)

Translations: Chinese (Simplified), Japanese, Korean, Russian

Watch: MIT's Deep Learning State of the Art lecture referencing this post

May 25th update: New graphics (RNN animation, word embedding graph), color coding, elaborated on the final attention example.



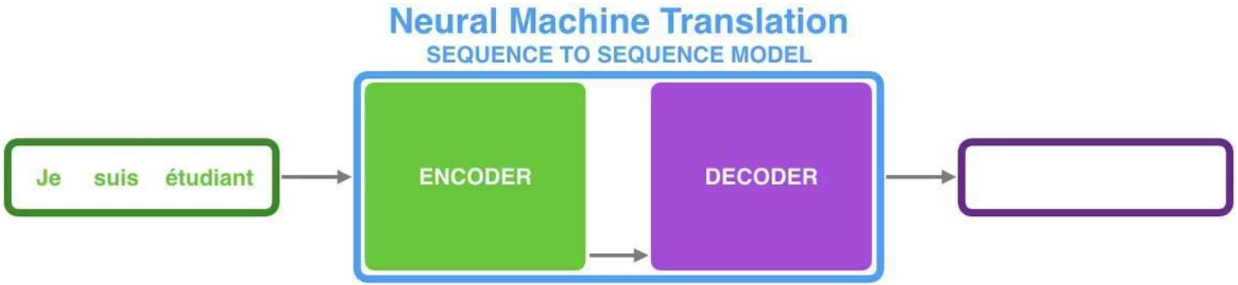
https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models)

Translations: Chinese (Simplified), Japanese, Korean, Russian

Watch: MIT's Deep Learning State of the Art lecture referencing this post

May 25th update: New graphics (RNN animation, word embedding graph), color coding, elaborated on the final attention example.



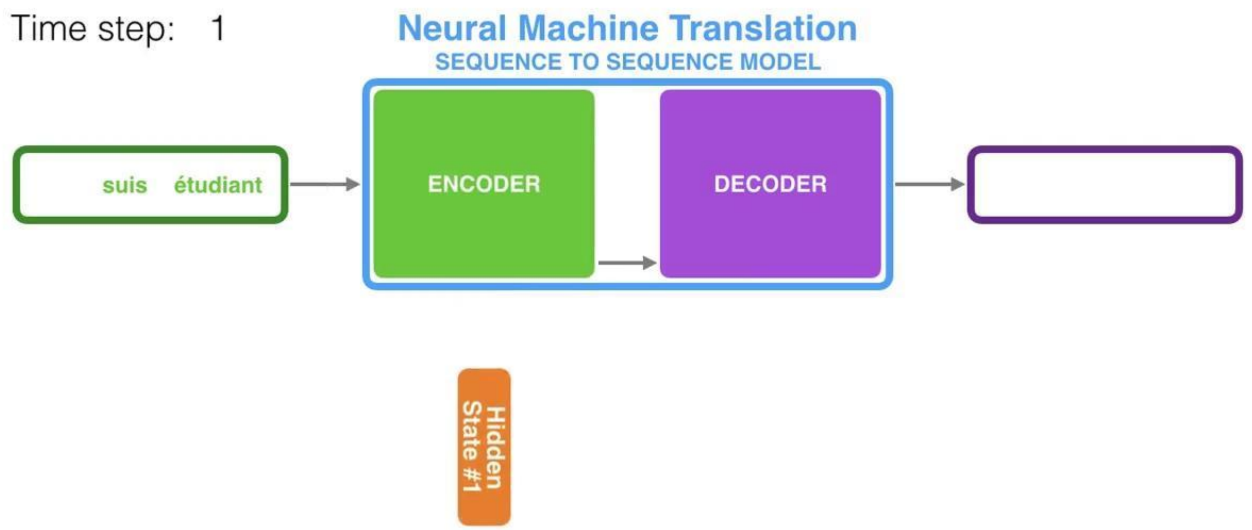
https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models)

Translations: Chinese (Simplified), Japanese, Korean, Russian

Watch: MIT's Deep Learning State of the Art lecture referencing this post

May 25th update: New graphics (RNN animation, word embedding graph), color coding, elaborated on the final attention example.



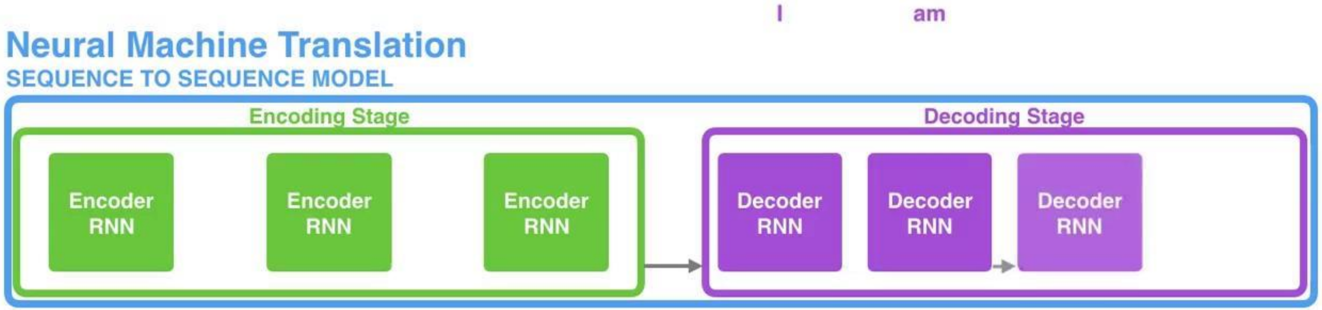
https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models)

Translations: Chinese (Simplified), Japanese, Korean, Russian

Watch: MIT's Deep Learning State of the Art lecture referencing this post

May 25th update: New graphics (RNN animation, word embedding graph), color coding, elaborated on the final attention example.



https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

https://gist.github.com/karpathy/d4dee566867f8291f086 (Andrej Karpathy)

RNN Vanilla: 112 lines of Python

GitHub Gist Search… All gists Back to GitHub

karpathy / **min-char-rnn.py**

Last active 3 days ago • Report abuse

Subscribe | Unstar 3,594 | Fork 1.2k

Code | Revisions 7 | Stars 3,589 | Forks 1,243 | Embed <script src="https://(| Download ZIP

Minimal character-level language model with a Vanilla Recurrent Neural Network, in Python/numpy

min-char-rnn.py Raw

```
"""
Minimal character-level Vanilla RNN model. Written by Andrej Karpathy (@karpathy)
BSD License
"""
import numpy as np

# data I/O
data = open('input.txt', 'r').read() # should be simple plain text file
chars = list(set(data))
data_size, vocab_size = len(data), len(chars)
print 'data has %d characters, %d unique.' % (data_size, vocab_size)
char_to_ix = { ch:i for i,ch in enumerate(chars) }
ix_to_char = { i:ch for i,ch in enumerate(chars) }

# hyperparameters
hidden_size = 100 # size of hidden layer of neurons
seq_length = 25 # number of steps to unroll the RNN for
learning_rate = 1e-1

# model parameters
Wxh = np.random.randn(hidden_size, vocab_size)*0.01 # input to hidden
Whh = np.random.randn(hidden_size, hidden_size)*0.01 # hidden to hidden
Why = np.random.randn(vocab_size, hidden_size)*0.01 # hidden to output
bh = np.zeros((hidden_size, 1)) # hidden bias
by = np.zeros((vocab_size, 1)) # output bias

def lossFun(inputs, targets, hprev):
  """
  inputs,targets are both list of integers.
  hprev is Hx1 array of initial hidden state
  returns the loss, gradients on model parameters, and last hidden state
  """
  xs, hs, ys, ps = {}, {}, {}, {}
```

Training: “Maior Dúvida da Aula” 2017

```
##### GoogLeNet, Inception Module

Não entendi muito bem sobre as inception layers na GoogLeNet. Entendi a ideia de fazer a mesma coisa de um filtro grande com vários
filtros menores. Com vários filtros menores temos menos parâmetros que um filtro grande?

Quando fazemos inception e concatenados os resultados, podemos comparar isso à criação de vetor de características? Porque estamos
retirando tipos diferentes de informações de uma mesma camada de input e juntando elas pra formar um output.

Acho que não consegui entender muito bem o inception module da arquitetura GoogLeNet. Para que ele serve exatamente? Obrigada.

no modelo de inception v4,  usa a paralelizacao para obter menos parametros, entao esso quer dizer que enquanto menos parametros e
mais profundo da melhores resultados?

Não entendi exatamente que fator possibilitou a remoção das camadas fully connected na GoogleLeNet. Pelo que eu entendi, as redes mais
modernas voltaram com a camada fully connected. Então quando usá-la ou não usá-la?

## Números de parâmetros

Em relação a arquiterua proposta na rede GoogLeNet, não ficou muito claro para mim as camadas internas, principalmente na parte em que
aplicar vários filtros menores, equilave a aplicar um filtro maior (embora o resultado não seja o mesmo).

Não ficou claro para mim qual a vantagem de se utilizar, por exemplo, 3 pequenos filtros 3x3 ao invés de um 7x7. Na aula você comentou
que é para evitar diminuir drasticamente a imagem, mas qual a desvantagem disso?

Eu nao entendi aquelas contas dos filtros que reduziam o numero de parametros

##### ResNet Filtro 1x1

Achei um pouco confuso as dimensões do filtro 1x1. Achei confuso a parte da convolução de tal filtro.

Não consegui entender a dinâmica dos small filters. O que se passa com uma convolução 1x1? Além disso, na LeNet as camadas foram
aumentadas para 512, 1024 e 512. O fato de serem potências de dois ajudou em algum aspecto do problema?
```

Training: “Maior Dúvida da Aula” 2017

```
iter 0, loss: 107.601633
----
 'ōqIE:ō:3(é
O Q.L"cÉhíL'uàfMO)êoâz.àãâéláç-)D(iéêdàF(lLFLrRcFA0nC(Pô(á#HM5éI?#ázHrtGTRF)5wlGaúa2éj?pd7,u
xp5LQ"r24F7élefL"Cabvêúhyldã 7àã2àObmxv?qnAodí'P)mTg4(u4F7úl3ómrQnmeFNbãoúvâ3i?sxsuRãjáécó.-
záy
---
```

```
iter 46000, loss: 23.238596
----
 és GoogLeNet. E a rede aprede?

O Daras dúvrvilg. ( ende no pré-tro "rar outlara destidas? Com uttres dessar algo us filtros
parte

e nar
```

```
iter 204000, loss: 10.733449
----
 to, ina utir alpal asvelum motrio tarada mexexenterna mai reviso de enter meiss grandas

##### ResNet Filtro 1x1? Alheing?

Não entendi exatamente que fia, confenhalo deset desecta..

##### Como as
```

Bag of Words

↓

RNNs

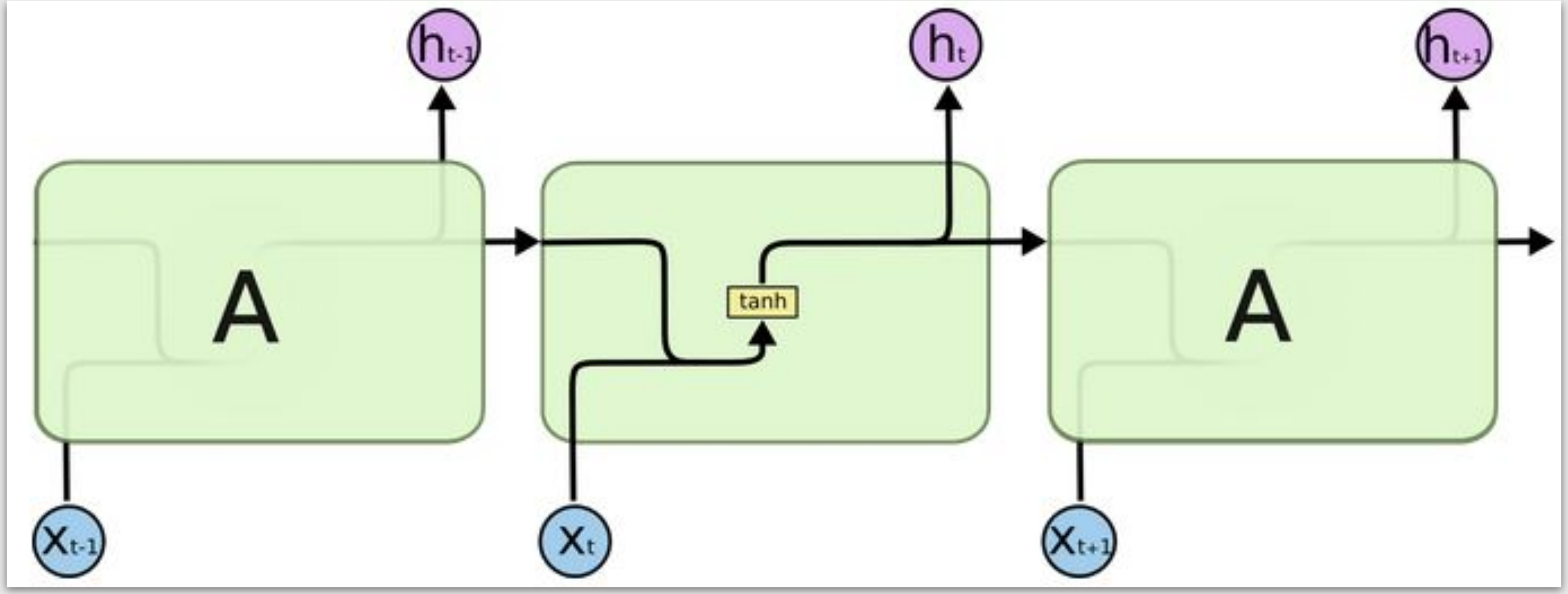
↓

LSTMs

↓

Transformers

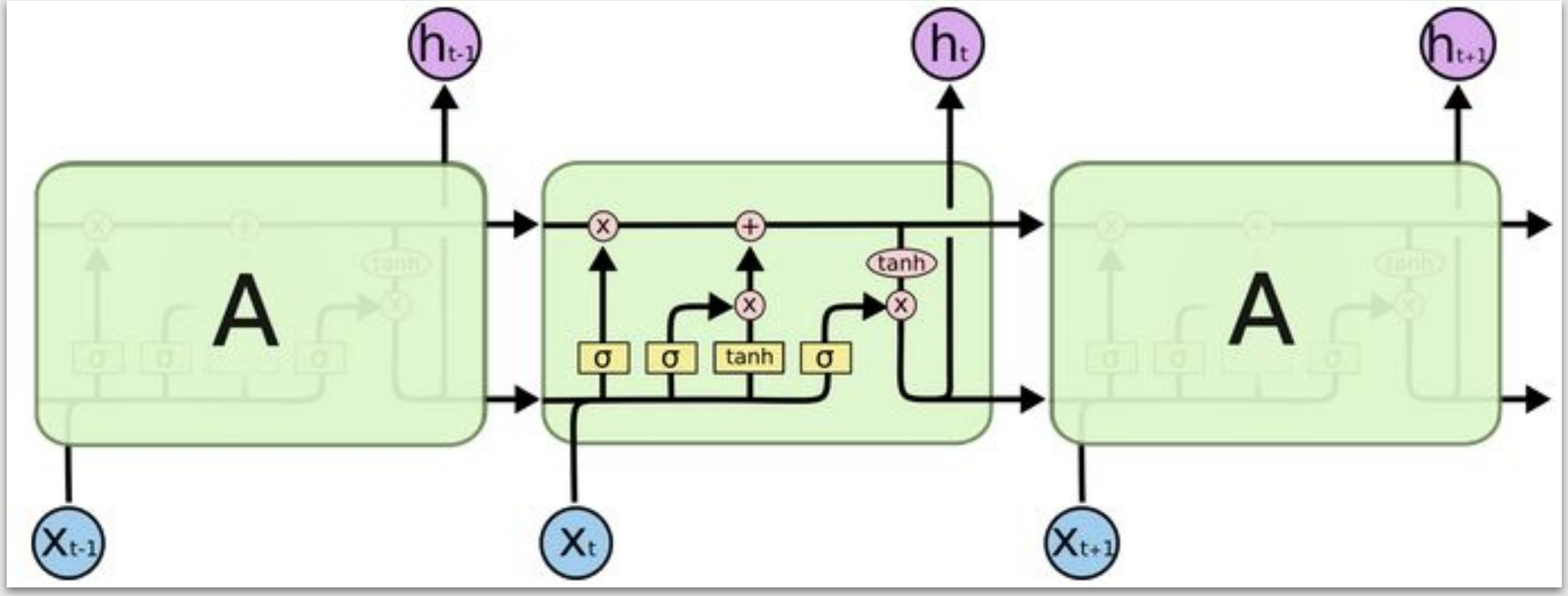
Long Short Term Memory (LSTM)



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short Term Memory (LSTM)

Hochreiter and Schmidhuber, “Long Short Term Memory”, Neural Computation, 1997



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

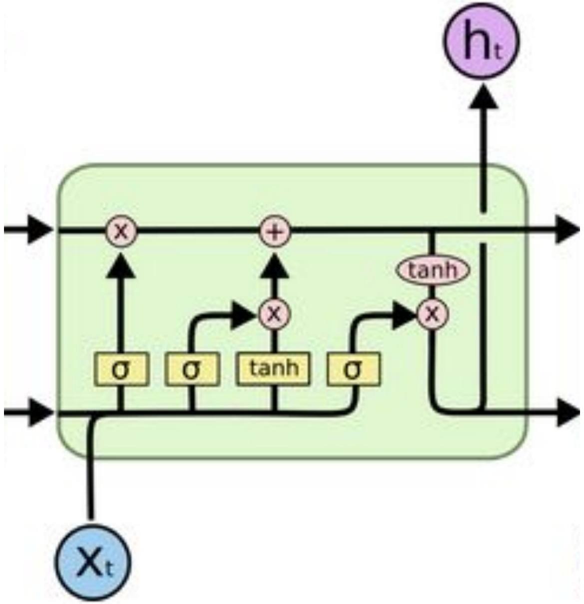
Long Short Term Memory (LSTM)

Hochreiter and Schmidhuber, "Long Short Term Memory", Neural Computation, 1997

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

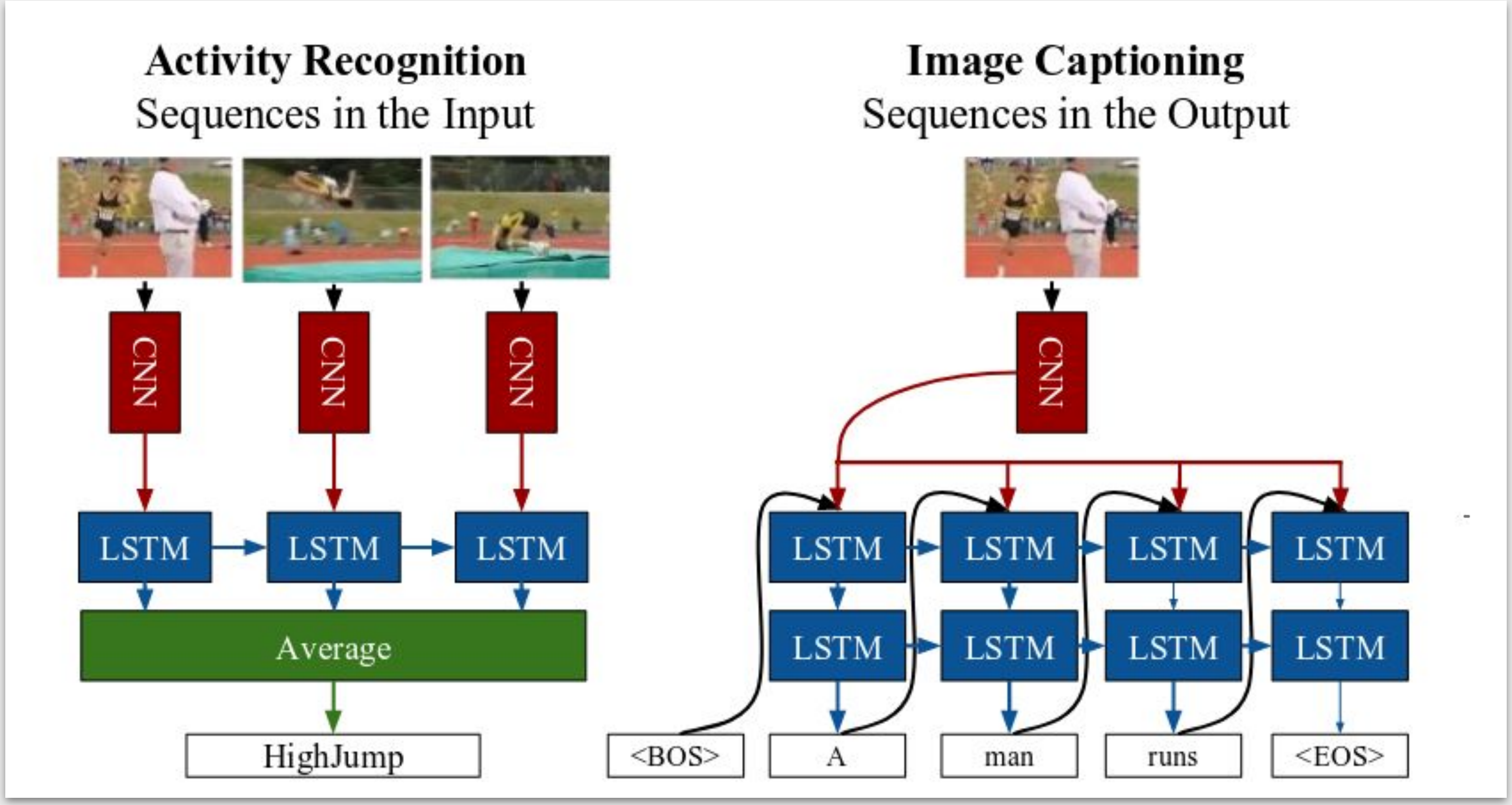
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Donahue et al., “Long-term Recurrent Convolutional Networks for Visual Recognition and Description”



https://towardsdatascience.com/the-fall-of-rnn-lstm-2d1594c74ce0

The fall of RNN / LSTM

Eugenio Culurciello Follow
Apr 13, 2018 · 8 min read

We fell for Recurrent neural networks (RNN), Long-short term memory (LSTM), and all their variants. **Now it is time to drop them!**

Bag of Words

↓

RNNs

↓

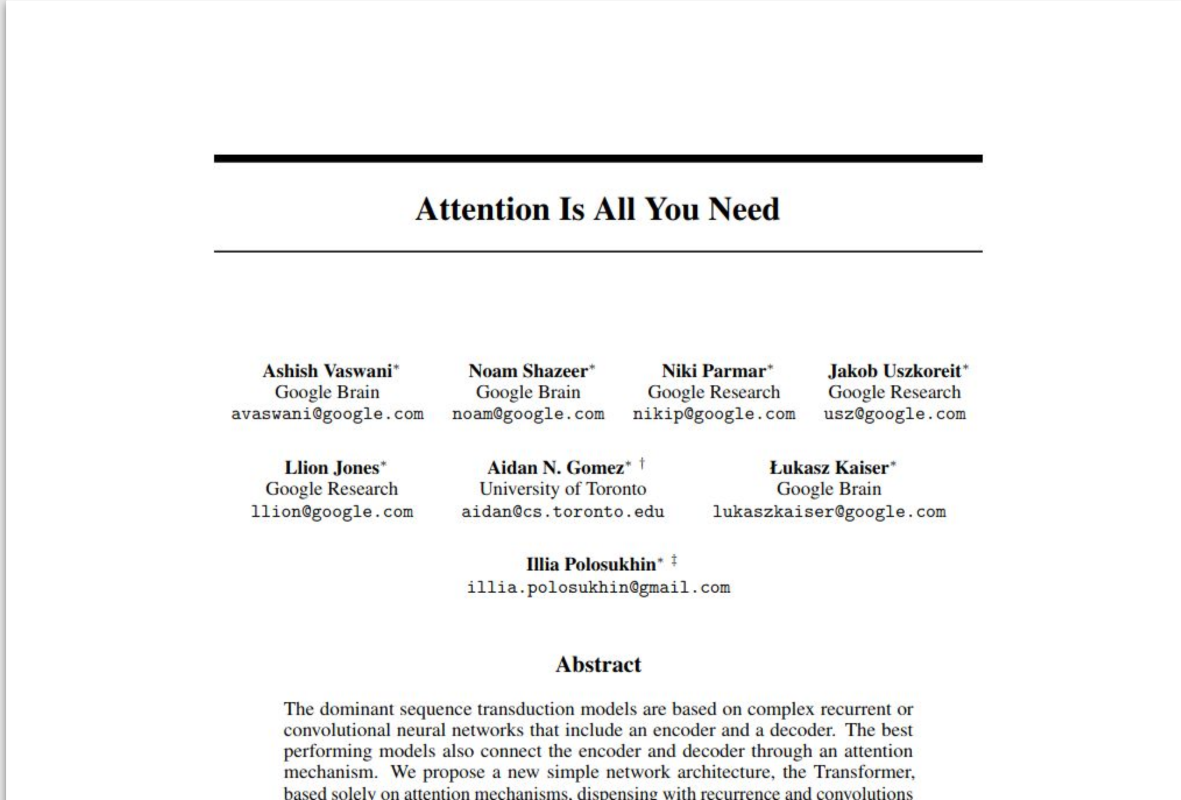
LSTMs

↓

➡ Transformers

Transformers

NeurIPS, 2017



Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions

https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention)

Translations: Chinese (Simplified), Japanese, Korean, Russian

Watch: MIT's Deep Learning State of the Art lecture referencing this post

May 25th update: New graphics (RNN animation, word embedding graph), color coding, elaborated on the final attention example.

Note: The animations below are videos. Touch or hover on them (if you're using a mouse) to get play controls so you can pause if needed.

Sequence-to-sequence models are deep learning models that have achieved a lot of success in tasks like machine translation, text summarization, and image captioning. Google Translate started using such a model in production in late 2016. These models are explained in the two pioneering papers (Sutskever et al., 2014, Cho et al., 2014).

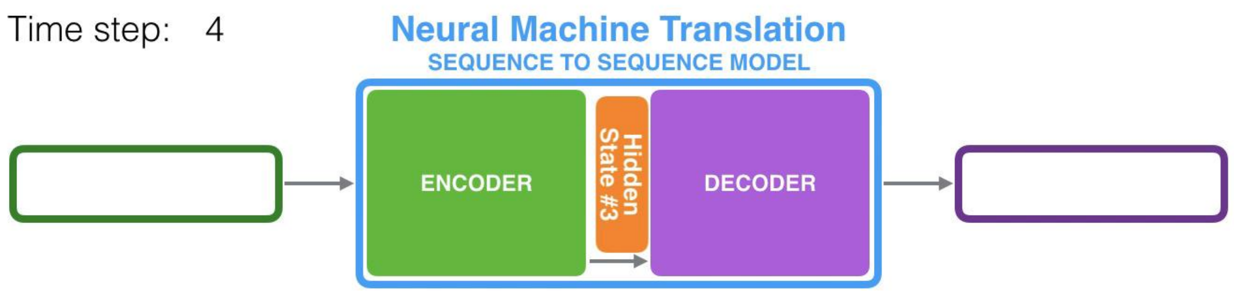
https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention)

Translations: Chinese (Simplified), Japanese, Korean, Russian

Watch: MIT's Deep Learning State of the Art lecture referencing this post

May 25th update: New graphics (RNN animation, word embedding graph), color coding, elaborated on the final attention example.



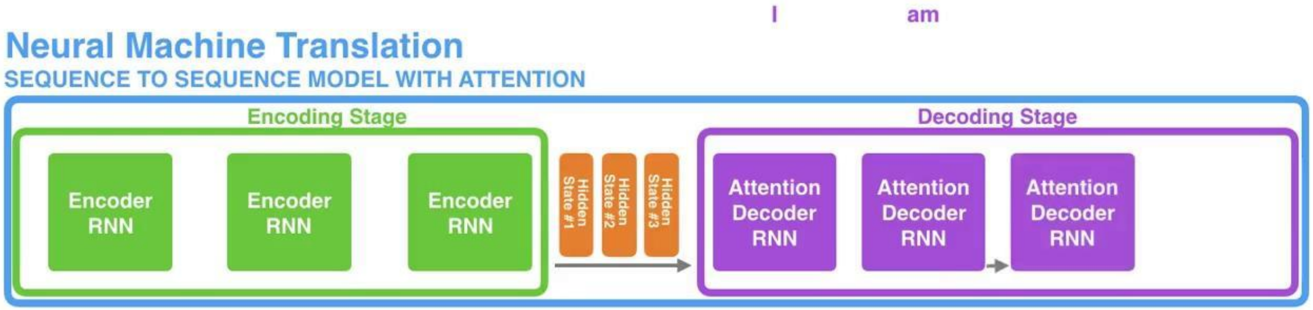
https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention)

Translations: Chinese (Simplified), Japanese, Korean, Russian

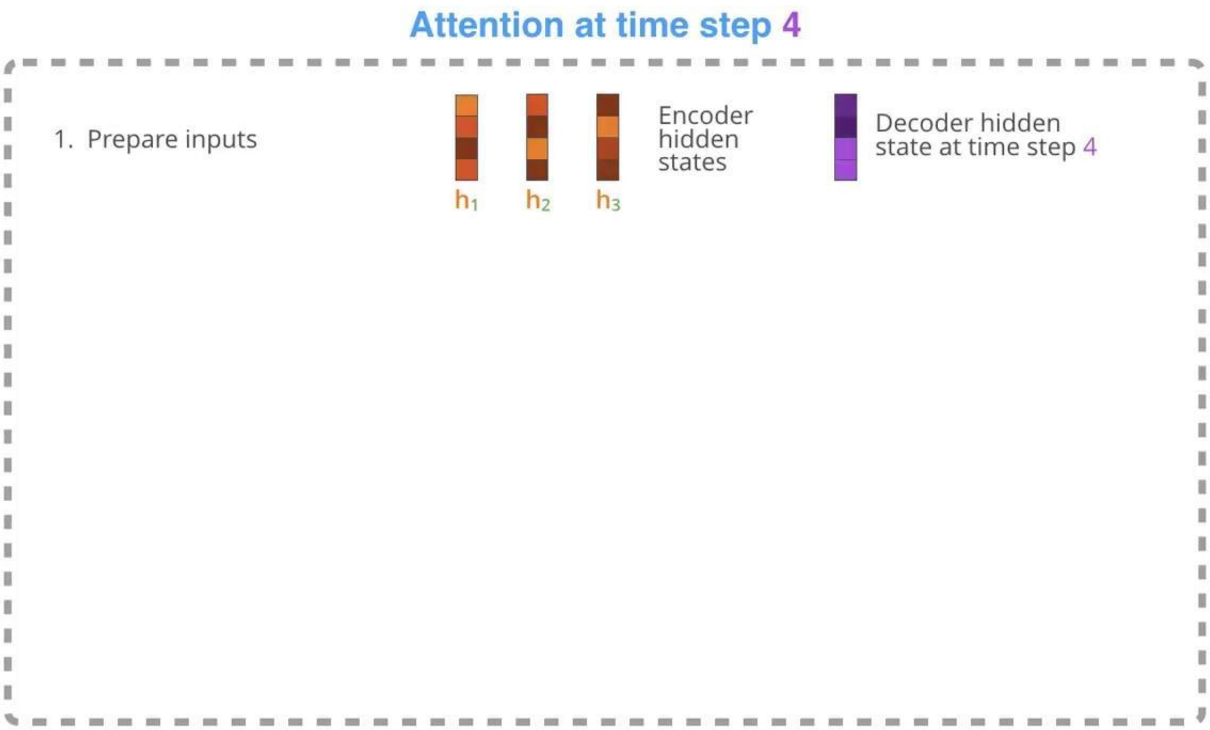
Watch: MIT's Deep Learning State of the Art lecture referencing this post

May 25th update: New graphics (RNN animation, word embedding graph), color coding, elaborated on the final attention example.



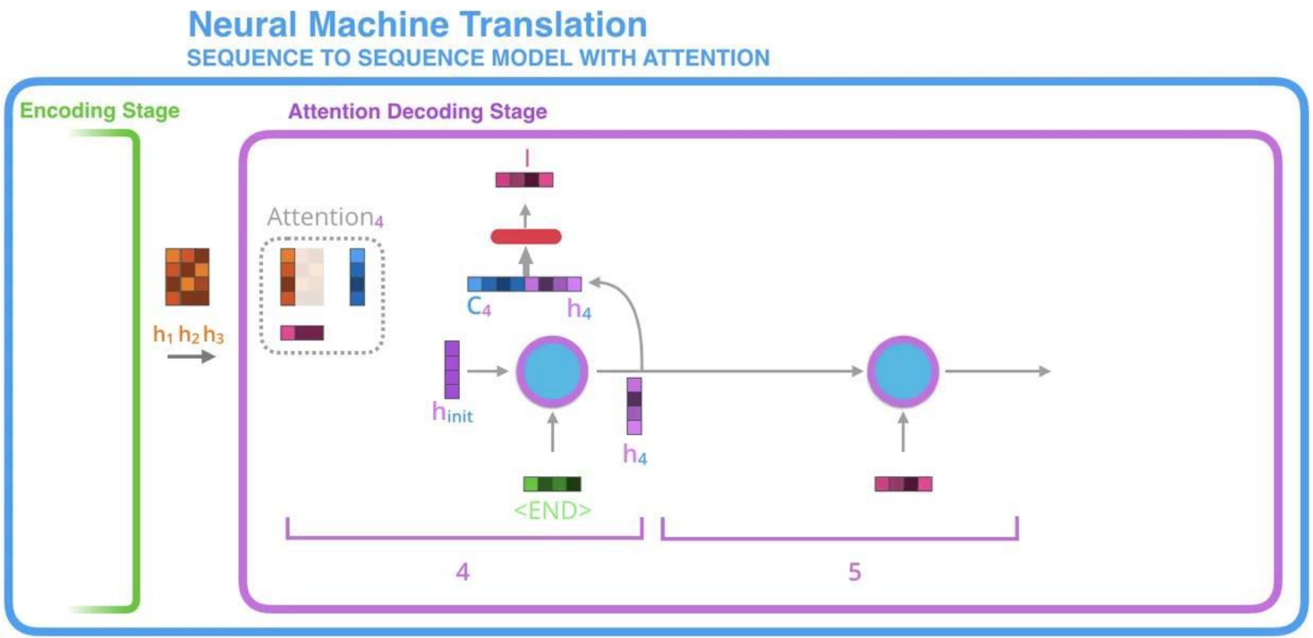
https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention)



https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention)



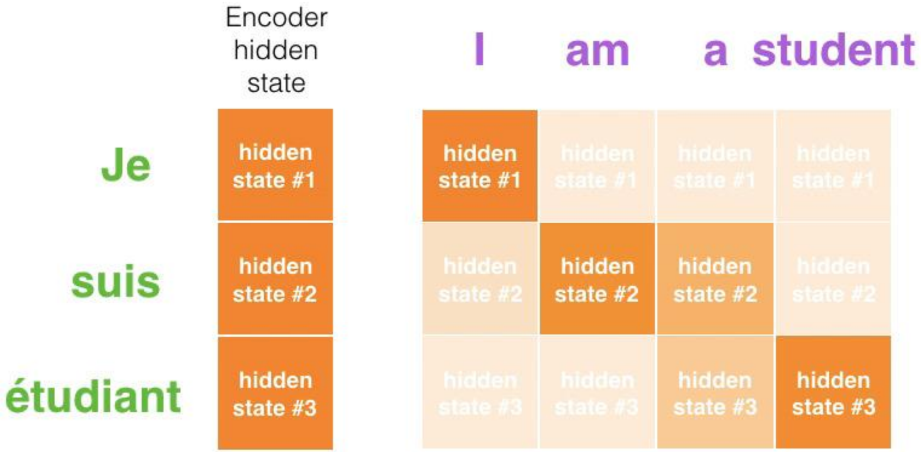
https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention)

Translations: Chinese (Simplified), Japanese, Korean, Russian

Watch: MIT's Deep Learning State of the Art lecture referencing this post

May 25th update: New graphics (RNN animation, word embedding graph), color coding, elaborated on the final attention example.



https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

https://jalammar.github.io/illustrated-transformer/

The Illustrated Transformer

Discussions: Hacker News (65 points, 4 comments), Reddit r/MachineLearning (29 points, 3 comments)

Translations: Chinese (Simplified), French, Japanese, Korean, Russian, Spanish

Watch: MIT's Deep Learning State of the Art lecture referencing this post

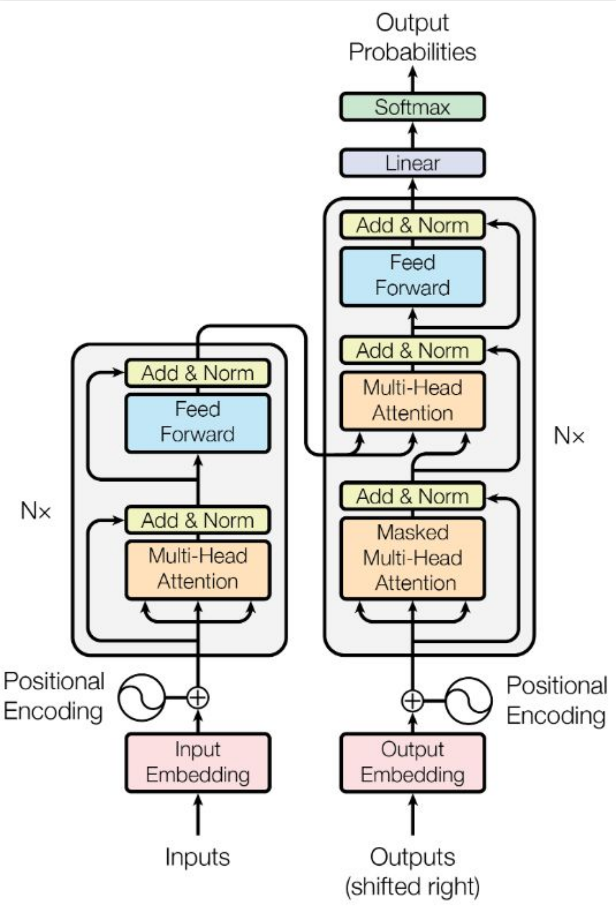
In the previous post, we looked at Attention – a ubiquitous method in modern deep learning models. Attention is a concept that helped improve the performance of neural machine translation applications. In this post, we will look at **The Transformer** – a model that uses attention to boost the speed with which these models can be trained. The Transformers outperforms the Google Neural Machine Translation model in specific tasks. The biggest benefit, however, comes from how The Transformer lends itself to parallelization. It is in fact Google Cloud's recommendation to use The Transformer as a reference model to use their Cloud TPU offering. So let's try to break the model apart and look at how it functions.

The Transformer was proposed in the paper Attention is All You Need. A TensorFlow implementation of it is available as a part of the Tensor2Tensor package. Harvard's NLP group created a guide annotating the paper with PyTorch implementation. In this post, we will attempt to oversimplify things a bit and introduce the concepts one by one to hopefully make it easier to understand to people without in-depth knowledge of the subject matter.

2020 Update: I've created a "Narrated Transformer" video which is a gentler approach to the topic:

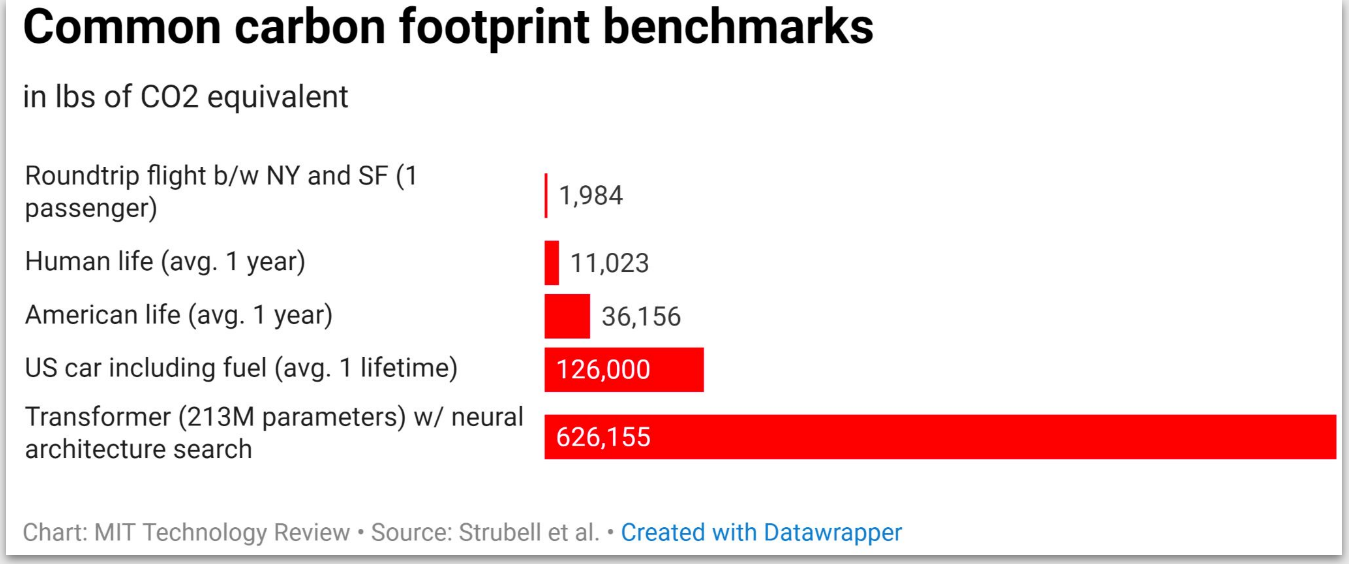
Transformer

- Transformer Architecture
  - Encoder & Decoder
  - Input & output embedding
  - Positional encoding
  - Self-attention
  - Multi-head attention
  - Masked multi-head attention
  - Residual connections
  - Layer Normalization
  - Feedforward



https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Transformer



ML CO2 Impact
https://mlco2.github.io/impact

CodeCarbon
https://github.com/mlco2/codecarbon

https://www.technologyreview.com/2019/06/06/239031/training-a-single-ai-model-can-emit-as-much-carbon-as-five-cars-in-their-lifetimes/

Transformers

ACM Conference on Fairness, Accountability and Transparency (FAccT) 2021

Year	Model	# of Parameters	Dataset Size
2019	BERT [39]	3.4E+08	16GB
2019	DistilBERT [113]	6.60E+07	16GB
2019	ALBERT [70]	2.23E+08	16GB
2019	XLNet (Large) [150]	3.40E+08	126GB
2020	ERNIE-GEN (Large) [145]	3.40E+08	16GB
2019	RoBERTa (Large) [74]	3.55E+08	161GB
2019	MegatronLM [122]	8.30E+09	174GB
2020	T5-11B [107]	1.10E+10	745GB
2020	T-NLG [112]	1.70E+10	174GB
2020	GPT-3 [25]	1.75E+11	570GB
2020	GShard [73]	6.00E+11	–
2021	Switch-C [43]	1.57E+12	745GB

Table 1: Overview of recent large language models

https://dl.acm.org/doi/pdf/10.1145/3442188.3445922

On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? 🦜

Emily M. Bender*
ebender@uw.edu
University of Washington
Seattle, WA, USA

Timnit Gebru*
timnit@blackinai.org
Black in AI
Palo Alto, CA, USA

Angelina McMillan-Major
aymm@uw.edu
University of Washington
Seattle, WA, USA

Shmargaret Shmitchell
shmargaret.shmitchell@gmail.com
The Aether

ABSTRACT

The past 3 years of work in NLP have been characterized by the development and deployment of ever larger language models, especially for English. BERT, its variants, GPT-2/3, and others, most recently Switch-C, have pushed the boundaries of the possible both through architectural innovations and through sheer size. Using these pretrained models and the methodology of fine-tuning them for specific tasks, researchers have extended the state of the art on a wide array of tasks as measured by leaderboards on specific benchmarks for English. In this paper, we take a step back and ask: How big is too big? What are the possible risks associated with this technology and what paths are available for mitigating those risks? We provide recommendations including weighing the environmental and financial costs first, investing resources into curating and carefully documenting datasets rather than ingesting everything on the web, carrying out pre-development exercises evaluating how the planned approach fits into research and development goals and supports stakeholder values, and encouraging research directions beyond ever larger language models.

CCS CONCEPTS

• **Computing methodologies** → **Natural language processing**.

ACM Reference Format:

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the Dangers of Stochastic Parrots: Can Language

alone, we have seen the emergence of BERT and its variants [39, 70, 74, 113, 146], GPT-2 [106], T-NLG [112], GPT-3 [25], and most recently Switch-C [43], with institutions seemingly competing to produce ever larger LMs. While investigating properties of LMs and how they change with size holds scientific interest, and large LMs have shown improvements on various tasks (§2), we ask whether enough thought has been put into the potential risks associated with developing them and strategies to mitigate these risks.

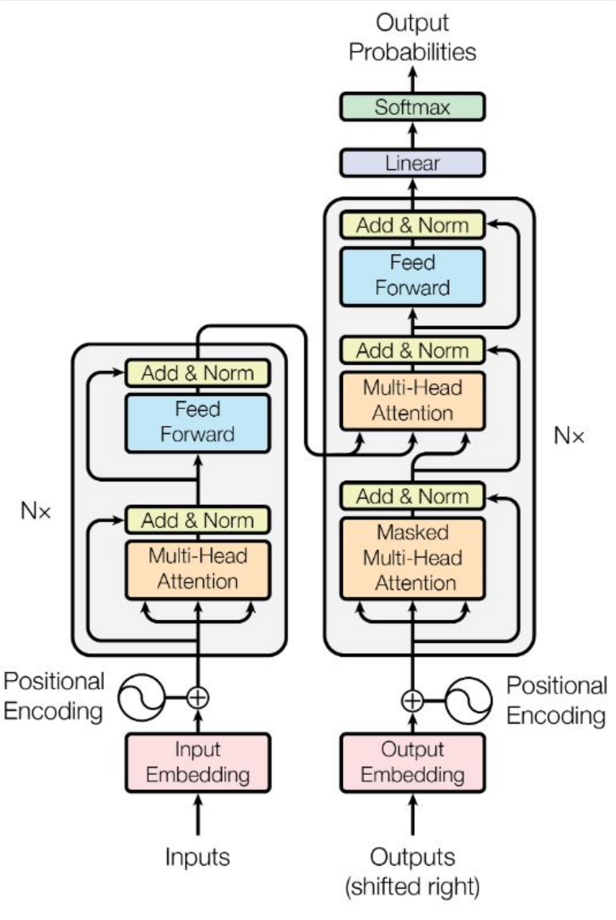
We first consider environmental risks. Echoing a line of recent work outlining the environmental and financial costs of deep learning systems [129], we encourage the research community to prioritize these impacts. One way this can be done is by reporting costs and evaluating works based on the amount of resources they consume [57]. As we outline in §3, increasing the environmental and financial costs of these models doubly punishes marginalized communities that are least likely to benefit from the progress achieved by large LMs and most likely to be harmed by negative environmental consequences of its resource consumption. At the scale we are discussing (outlined in §2), the first consideration should be the environmental cost.

Just as environmental impact scales with model size, so does the difficulty of understanding what is in the training data. In §4, we discuss how large datasets based on texts from the Internet overrepresent hegemonic viewpoints and encode biases potentially damaging to marginalized populations. In collecting ever larger datasets we risk incurring documentation debt. We recommend

https://dl.acm.org/doi/pdf/10.1145/3442188.3445922

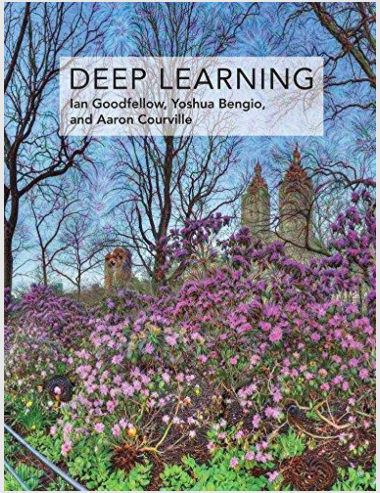
Transformer

- Transformer Architecture
  - Encoder & Decoder
  - Input & output embedding
  - Positional encoding
  - Self-attention
  - Multi-head attention
  - Masked multi-head attention
  - Residual connections
  - Layer Normalization
  - Feedforward



https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

References



"Deep Learning", I. Goodfellow and Y. Bengio and A. Courville, https://www.deeplearningbook.org, 2016. Cap. 10

"Recurrent Neural Networks | MIT 6.S191", https://youtu.be/SEnXr6v2ifU **(Fev., 2020)**

"A friendly introduction to Recurrent Neural Networks", Luis Serrano, https://youtu.be/UNmqTiOnRf

"Attention, please! A survey of Neural Attention Models in Deep Learning", A. Correia and E. Colombini, https://arxiv.org/pdf/2103.16775.pdf