

# Lecture Notes for **Machine Learning in Python**

Professor Eric Larson

## **History and Introduction to Recurrent Neural Networks**

# Agenda

- Intro to Recurrent Neural Network Architectures
  - RNNs, GRUs, LSTMs
  - Training for characters

# Lecture Agenda

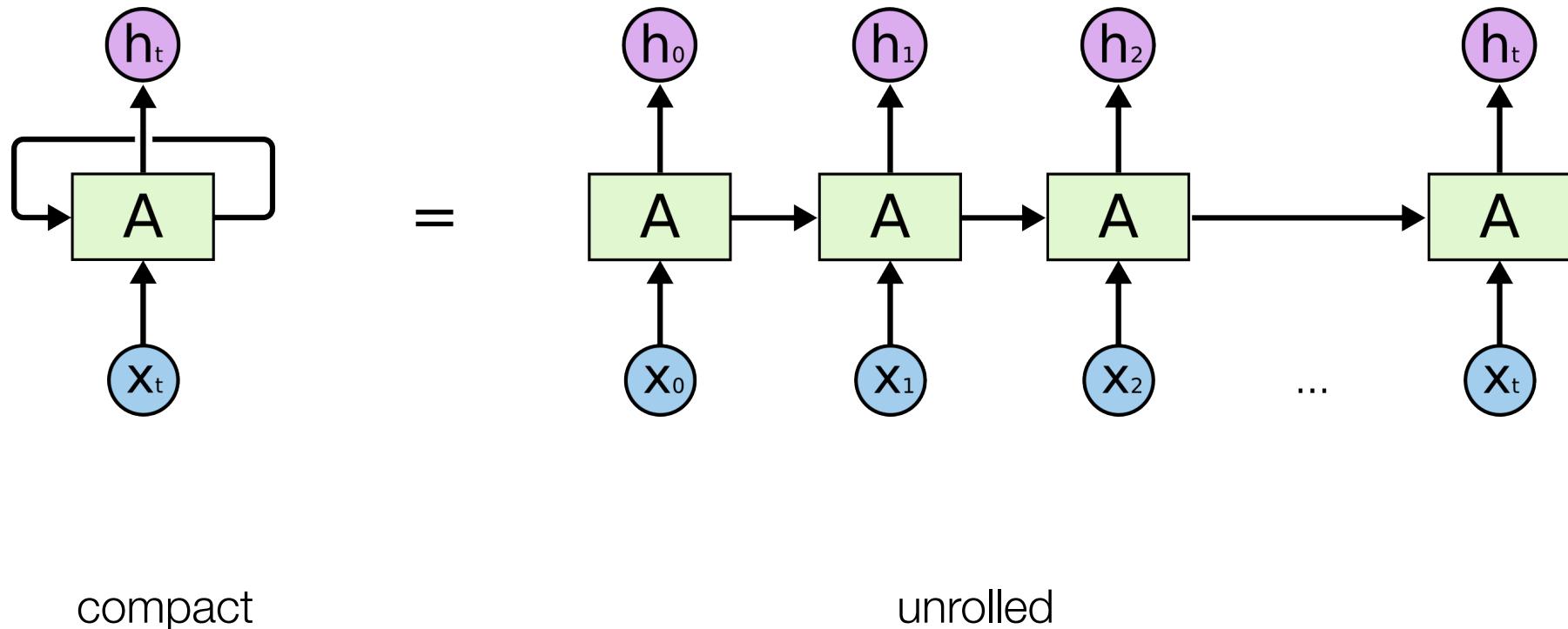
- Logistics
  - CNNs due date
  - RNNs due date
- Recurrent Networks (~three lecture agenda)
  - Overview
  - Problem Types
  - Embeddings
  - Types of RNNs
  - Demo A
  - CNNs and RNNs
  - Demo B
  - Ethical Use Case for RNNs
  - Course Retrospective

# History of Recurrent Neural Networks



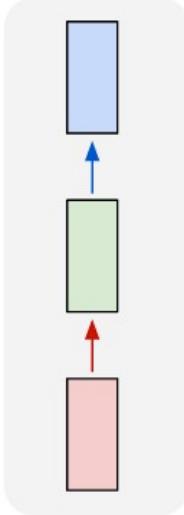
# Recurrent Networks: Main Idea

- equations for recurrent networks

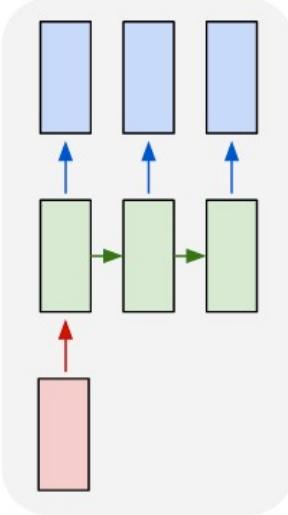


# Recurrent Networks: Problem Types

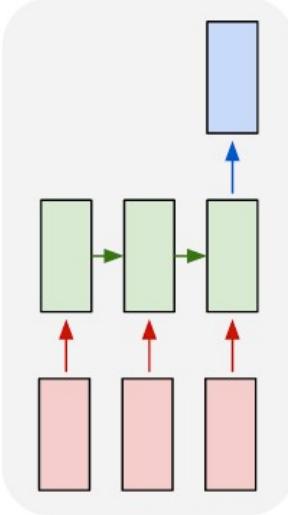
one to one



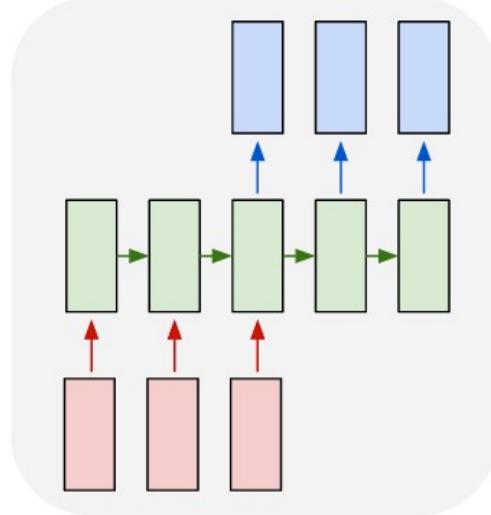
one to many



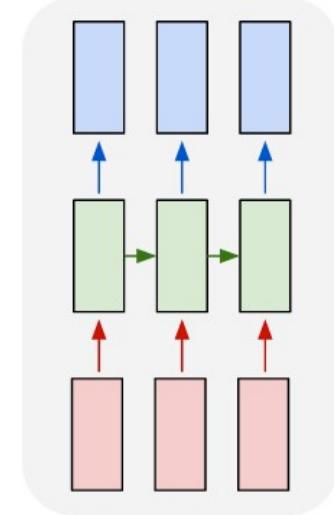
many to one



many to many

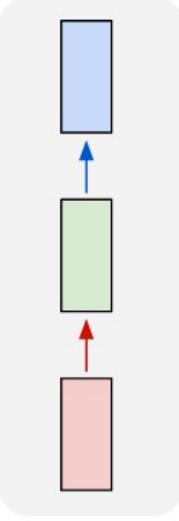


many to many

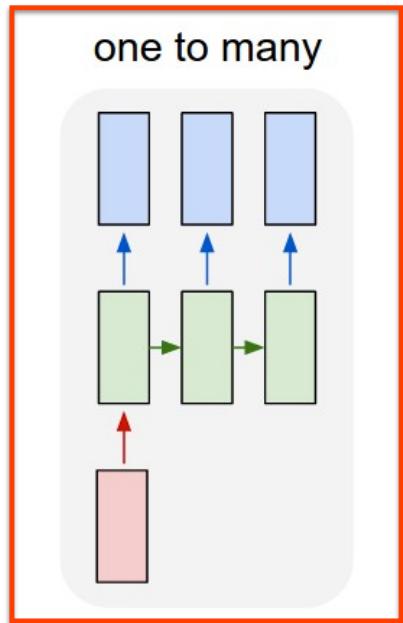


# Recurrent Networks: Problem Types

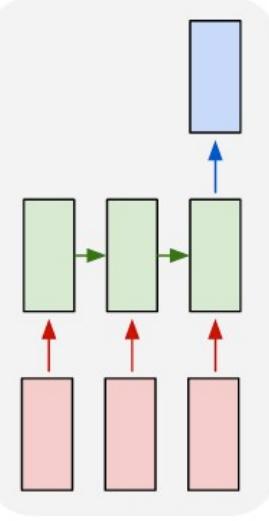
one to one



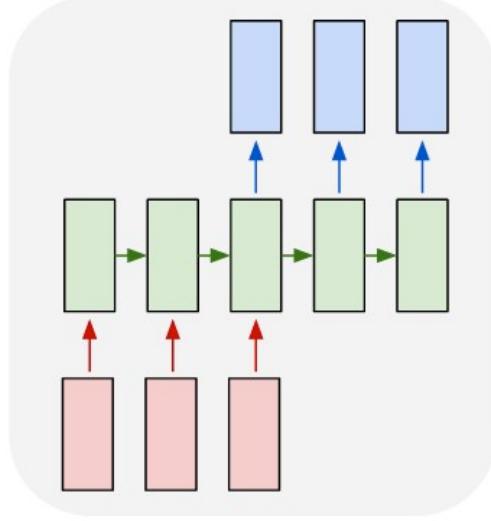
one to many



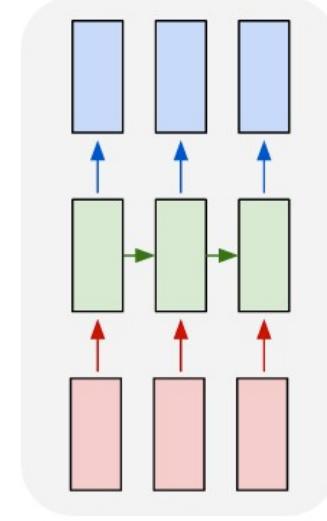
many to one



many to many



many to many



A person riding a motorcycle on a dirt road.



A group of young people playing a game of frisbee.



A herd of elephants walking across a dry grass field.

Two dogs play in the grass.



Two hockey players are fighting over the puck.



A close up of a cat laying on a couch.

A skateboarder does a trick on a ramp.



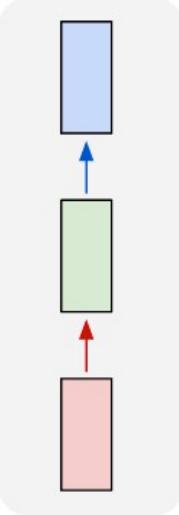
A little girl in a pink hat is blowing bubbles.



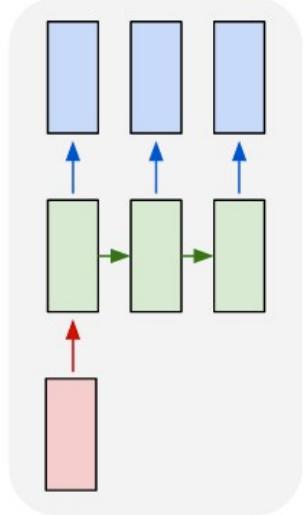
A red motorcycle parked on the side of the road.

# Recurrent Networks: Problem Types

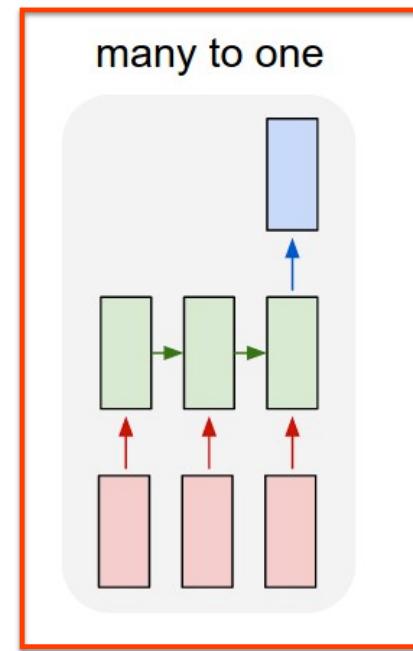
one to one



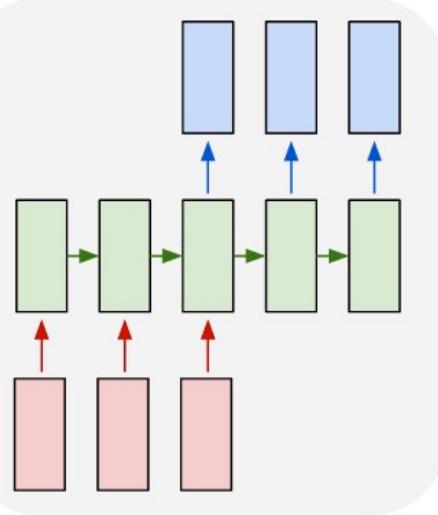
one to many



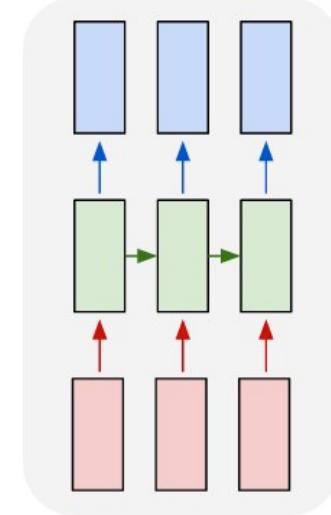
many to one



many to many



many to many



The movie is great.



The movie stars Mr. X



The movie is horrible.



# Recurrent Networks: Ontology

Eva Ingolf is a well known Icelandic violinist particularly recognized for her authoritative performances of solo works by J. S. Bach. She comes from a leading musical family and her father Ingólfur Guðbrandsson premiered many of the great choral works in Iceland and six of her sisters and brothers are professional musicians who have made an important contribution to the high quality of the musical life in the country. Eva Ingolf currently lives in New York City with her husband Kristinn Sv.

## **Artist**

Shaun Norris (born 14 May 1982) is a South African professional golfer. Norris plays on the Sunshine Tour where he has won twice. He won the inaugural Africa Open in 2008 and the Nashua Masters in 2011. He also began playing on the European Tour in 2011 after graduating from qualifying school.

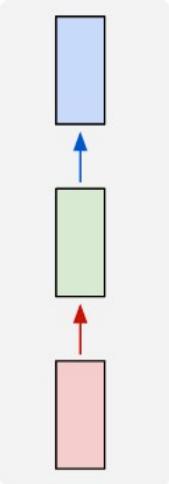
## **Athlete**

Palace Software was a British video game publisher and developer during the 1980s based in London England. It was notable for the Barbarian and Cauldron series of games for 8-bit home computer platforms in particular the ZX Spectrum Amstrad CPC and Commodore 64.

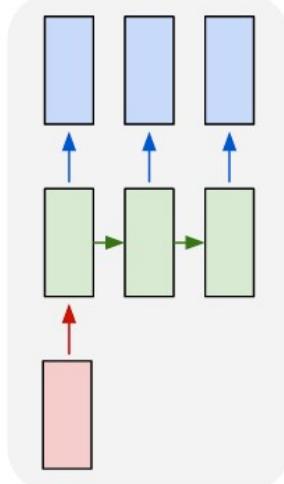
## **Company**

# Recurrent Networks: Problem Types

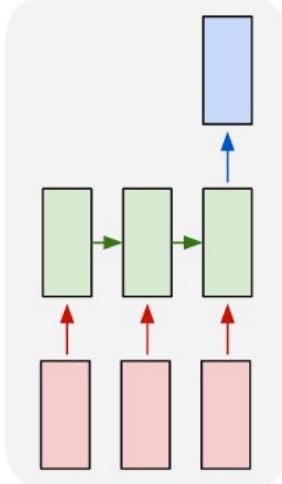
one to one



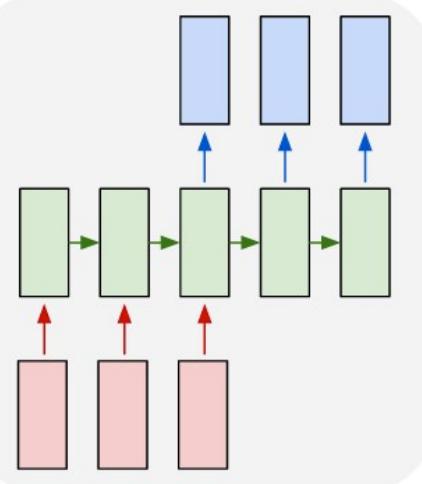
one to many



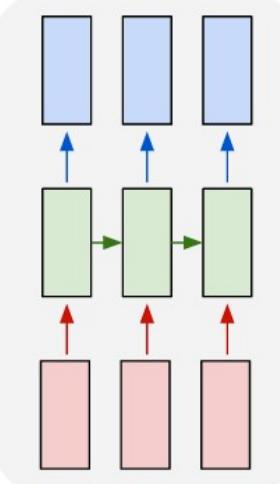
many to one



many to many



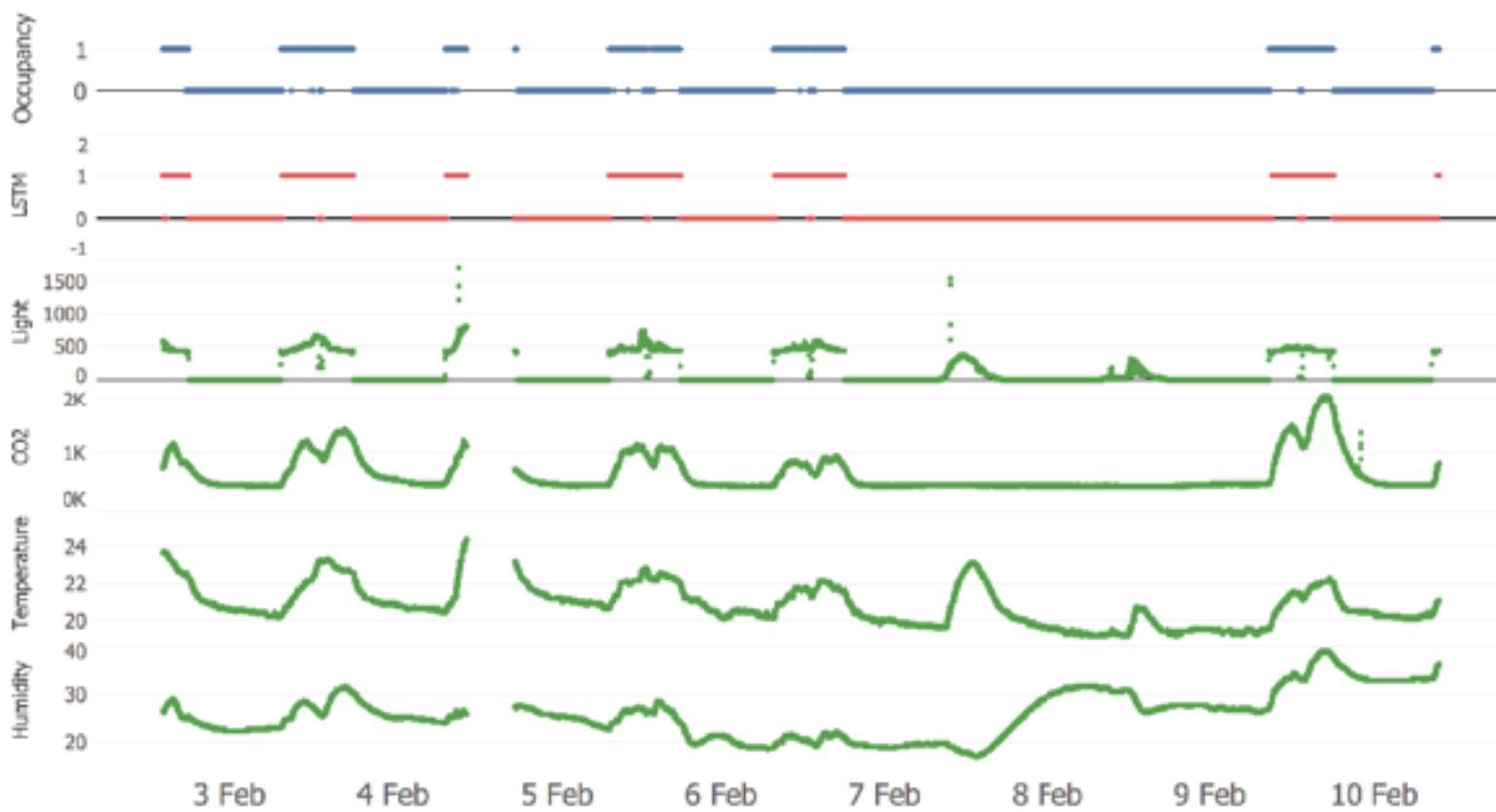
many to many



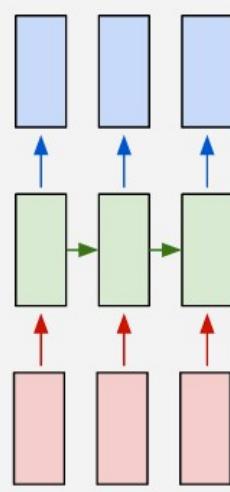
Das Wirtschaftswachstum hat sich in den letzten Jahren verlangsamt .  
Economic growth has slowed down in recent years .

La croissance économique s' est ralentie ces dernières années .

# Recurrent Networks: Problem Types



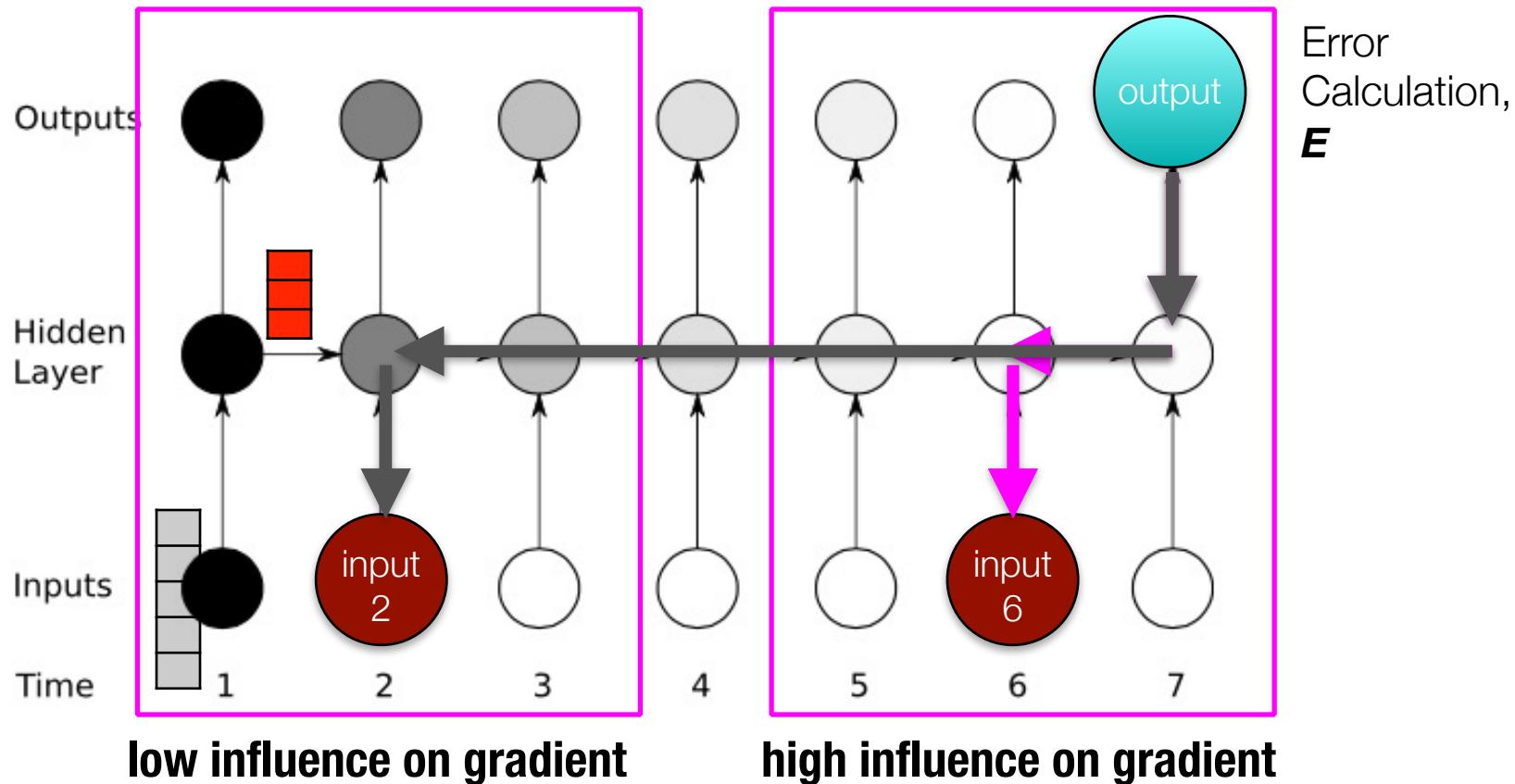
many to many



sequence to sequence

# Recurrent Networks, the Age Old Problem

- vanishing gradients: why are these a problem?



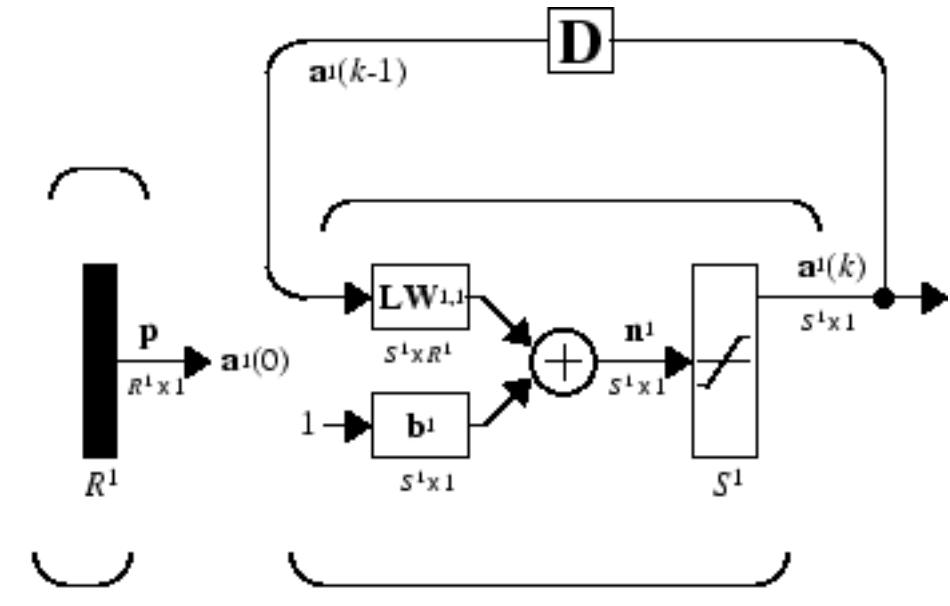
$$\frac{\partial E_t}{\partial S_{t-k}} = \frac{\partial E_t}{\partial S_t} \frac{\partial S_t}{\partial S_{t-k}} = \frac{\partial E_t}{\partial S_t} \left( \frac{\partial S_t}{\partial S_{t-1}} \frac{\partial S_{t-1}}{\partial S_{t-2}} \dots \frac{\partial S_{t-k+1}}{\partial S_{t-k}} \right) = \frac{\partial E_t}{\partial S_t} \prod_{i=1}^k \frac{\partial S_{t-i+1}}{\partial S_{t-i}}$$

# History of Recurrent Networks

- Hopfield Network, 1982



John Hopfield, Princeton



Initial conditions

Symmetric saturated linear layer

$\mathbf{a}^1(0) = \mathbf{p}$  and then for  $k = 1, 2, \dots$

$\mathbf{a}^1(k) = \text{satlims}(\mathbf{LW}_{1,1}\mathbf{a}^1(k-1)) + \mathbf{b}^1$

*Neural Network Design*, Hagan, Demuth, Beale, and De Jesus

# History of Recurrent Networks

- Elman/Jordan Networks, ~1988

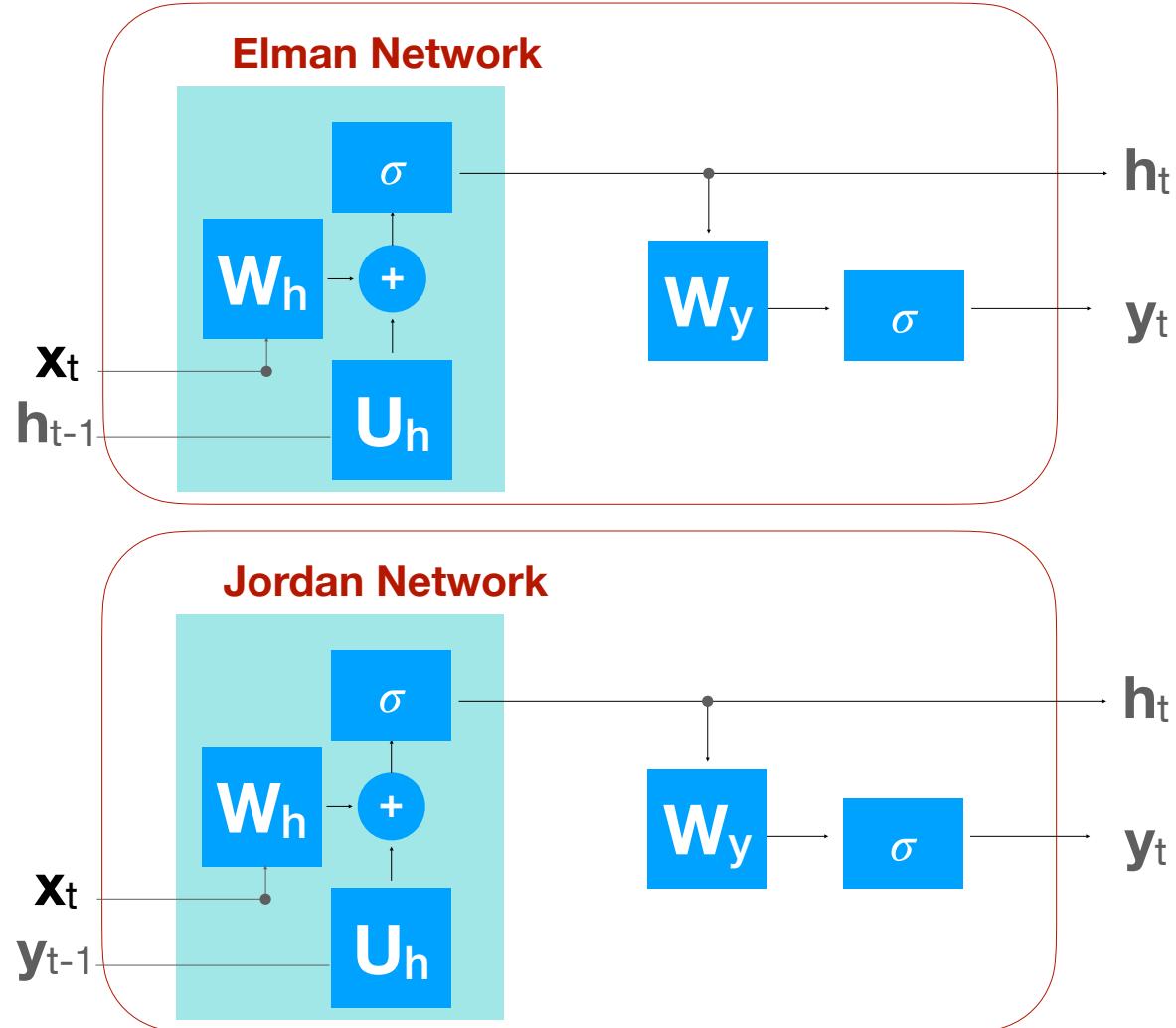
**Contribution:**  
Time Steps for Unrolling



Jeffrey Elman, UCSD



Michael Jordan, Berkeley



# History of Recurrent Networks

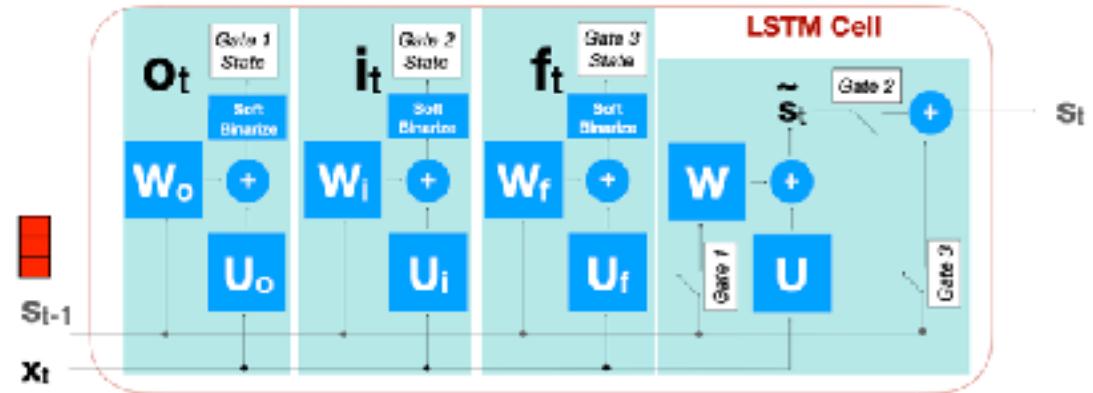
- Long Short Term Memory, ~1997 - 2010



Sepp Hochreiter, Many Universities



Jürgen Schmidhuber, Switzerland



*More on these later*

## Contribution:

Long Duration Memory  
State Vector separate from Output

# History of Recurrent Networks

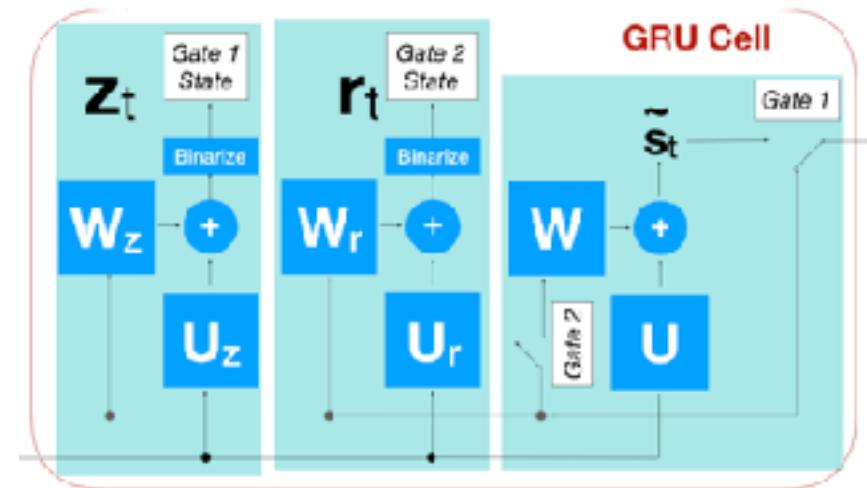
- Gated Recurrent Units, ~2014



Yoshua Bengio



Kyunghyun Cho, Professor at NYU



*More on these later*

## Contribution:

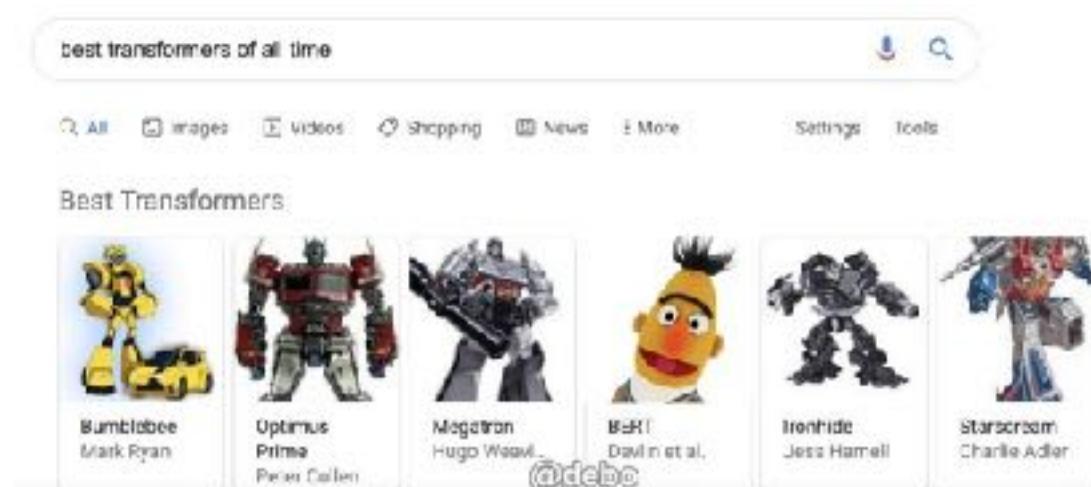
Forced Decision on State Vector

# Other big advances

- **Attention** (early 2017)
- 1D **Convolution** to Replace RNN (late 2017)
- Marriage of CNN and RNN
  - The **transformer** architecture (early 2018)
  - Self-attention (late 2018)
- **Multi-headed** attention in transformers (2018)
- **BERT, GPT-#, etc.** (2019)

This Course

NLP Course



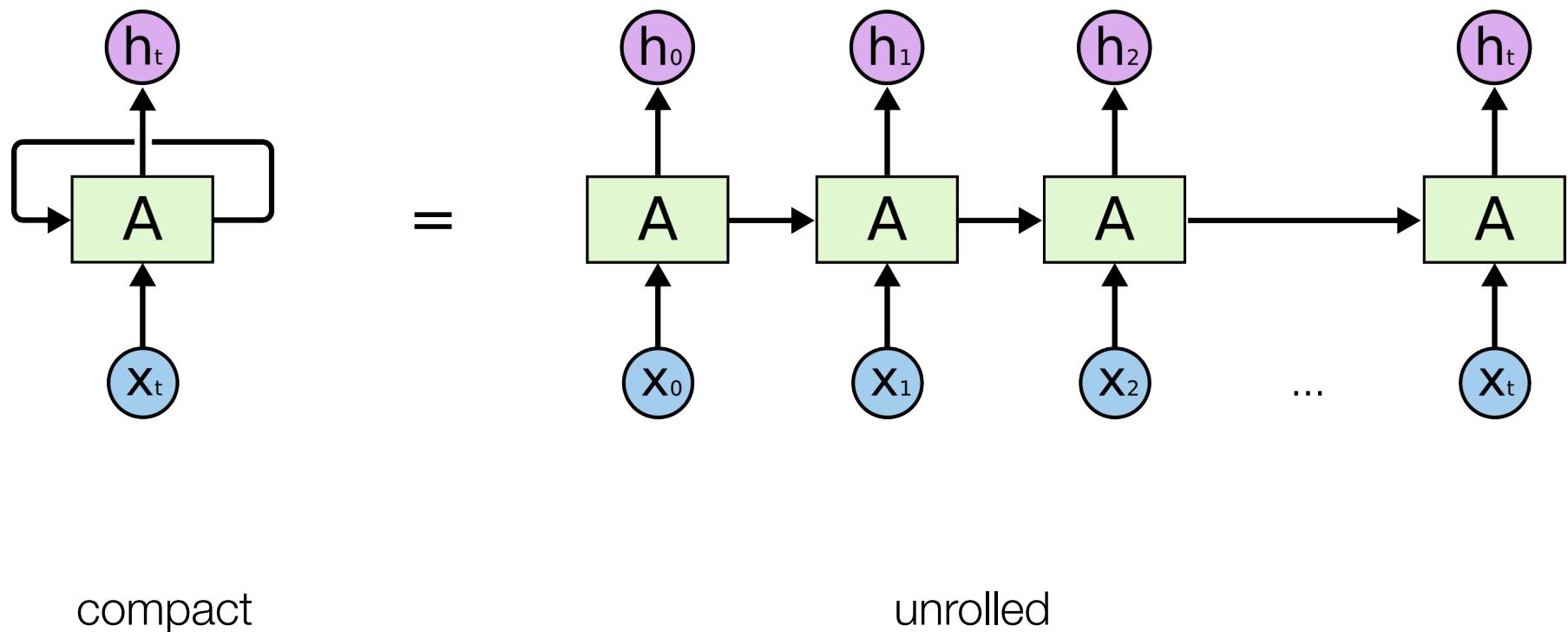
# Basics of Recurrent Neural Networks



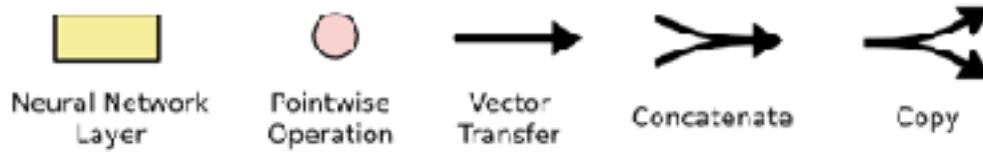
WHEN YOU TRAIN PREDICTIVE MODELS  
ON INPUT FROM YOUR USERS, IT CAN  
LEAK INFORMATION IN UNEXPECTED WAYS.

For now, put those architectures in long term memory. 😂

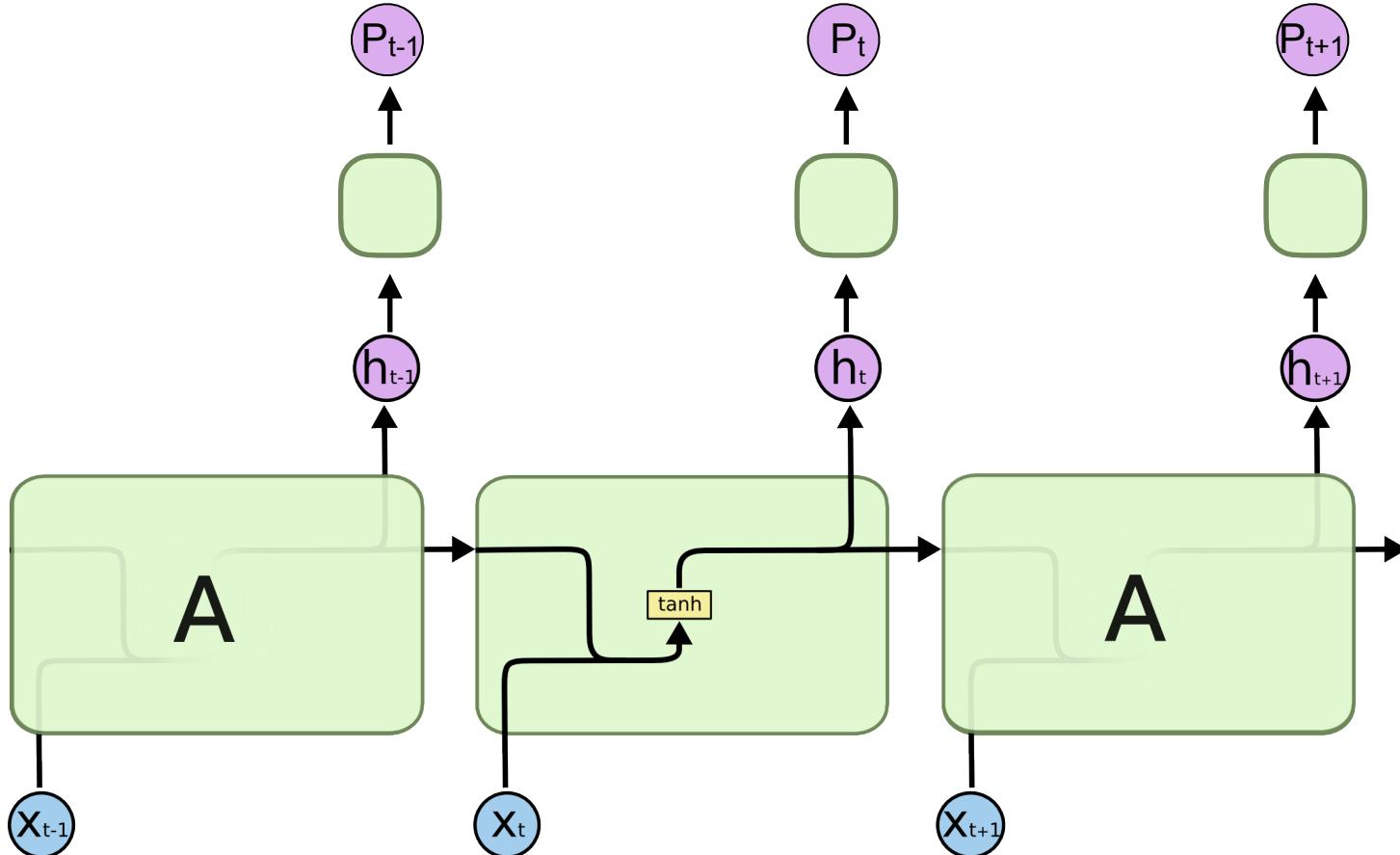
# Recurrent Networks: Main Idea



# Starting Basic



- basic RNN

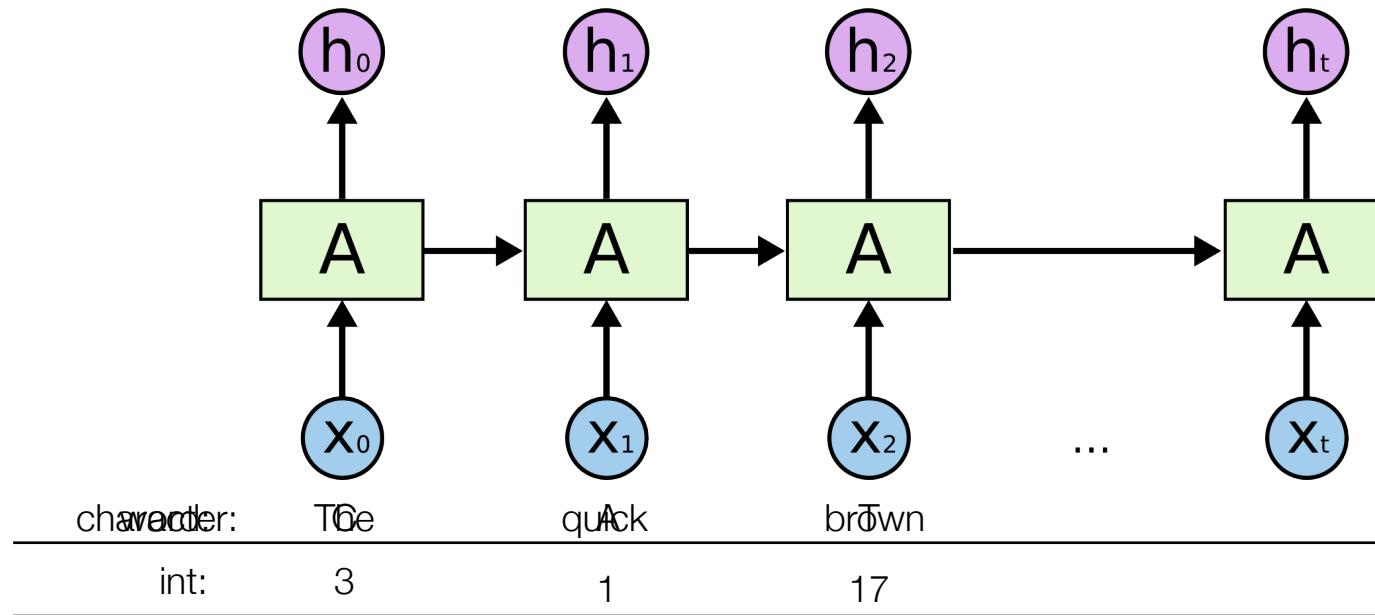


$$h_t = \tanh(W_A (X_t \oplus h_{t-1}) + b_A)$$

$$P_t = \text{softmax}(W_P h_t + b_P)$$

# Recurrent Networks: Representation

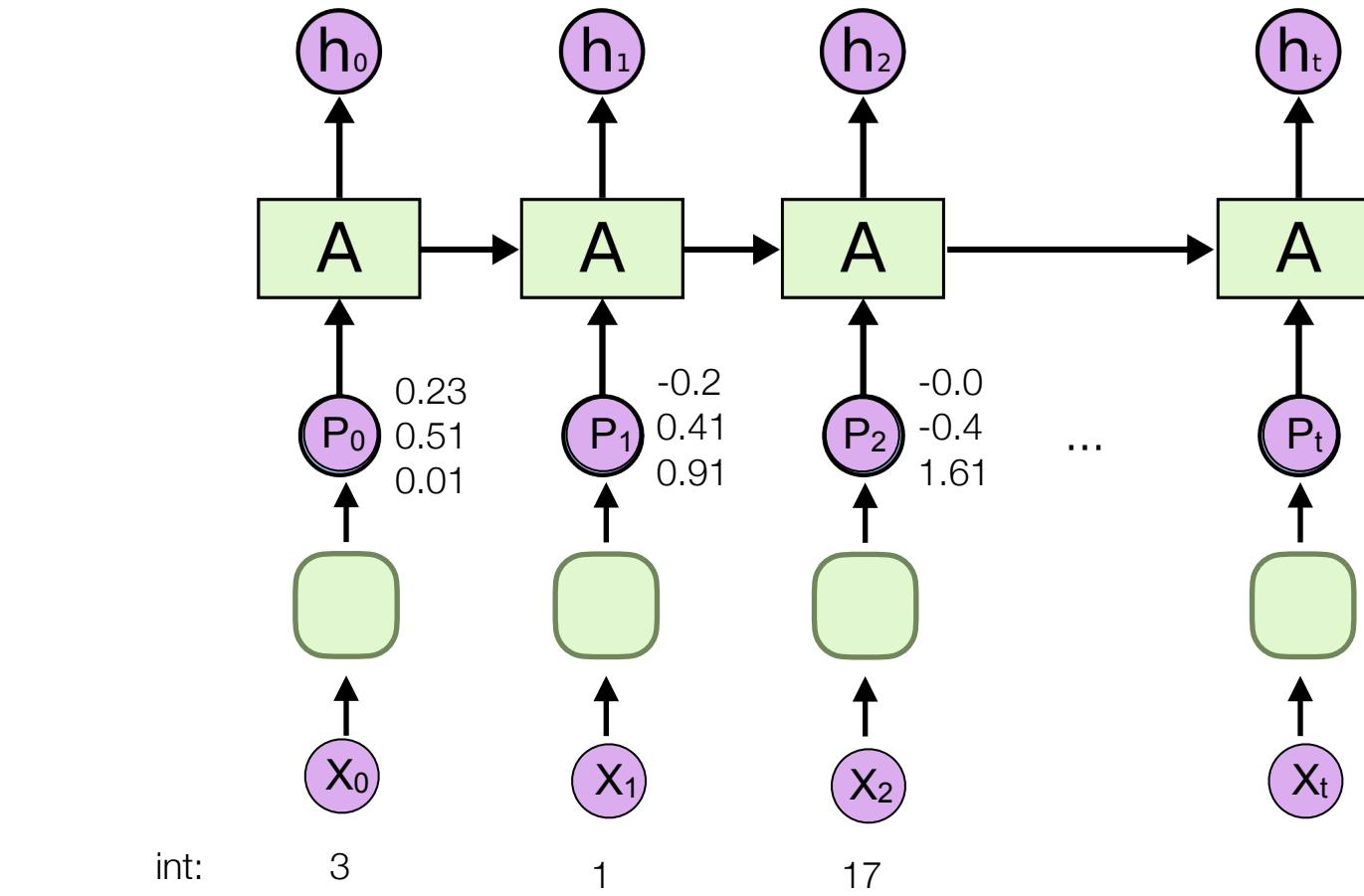
- python:



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

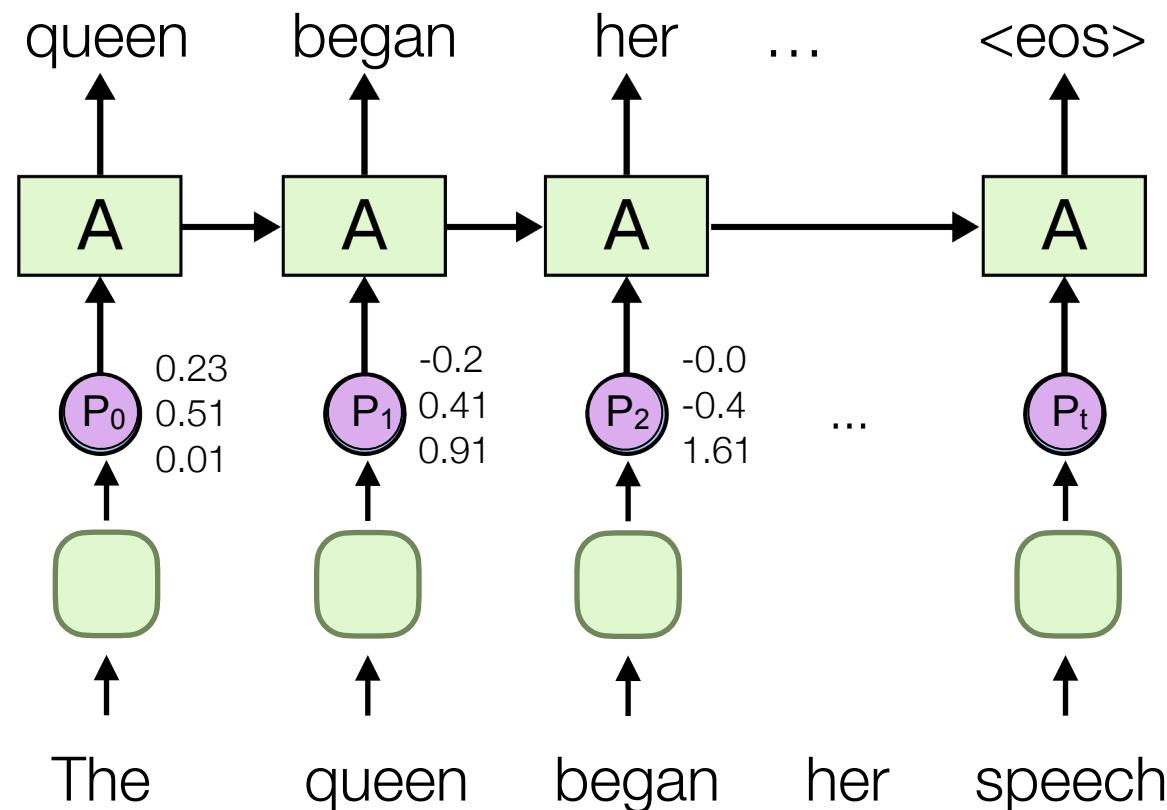
<http://r2rt.com/recurrent-neural-networks-in-tensorflow-i.html> 21

# Word Embeddings (like Wide/Deep)



# Word Embeddings: Training

- many training options exist
  - a popular option, next word prediction



# Word Embeddings

- Many are pre-trained for you!!

GloVe

## Highlights

### 1. Nearest neighbors

The Euclidean distance (or cosine similarity) between two word vectors provides an effective method for measuring the linguistic or semantic similarity of the corresponding words. Sometimes, the nearest neighbors according to this metric reveal rare but relevant words that lie outside an average human's vocabulary. For example, here are the closest words to the target word *frog*:

0. *frog*
1. *frogs*
2. *toad*
3. *litoria*
4. *leptodactylidae*
5. *rana*
6. *lizard*
7. *eleutherodactylus*



3. *litoria*



4. *leptodactylidae*



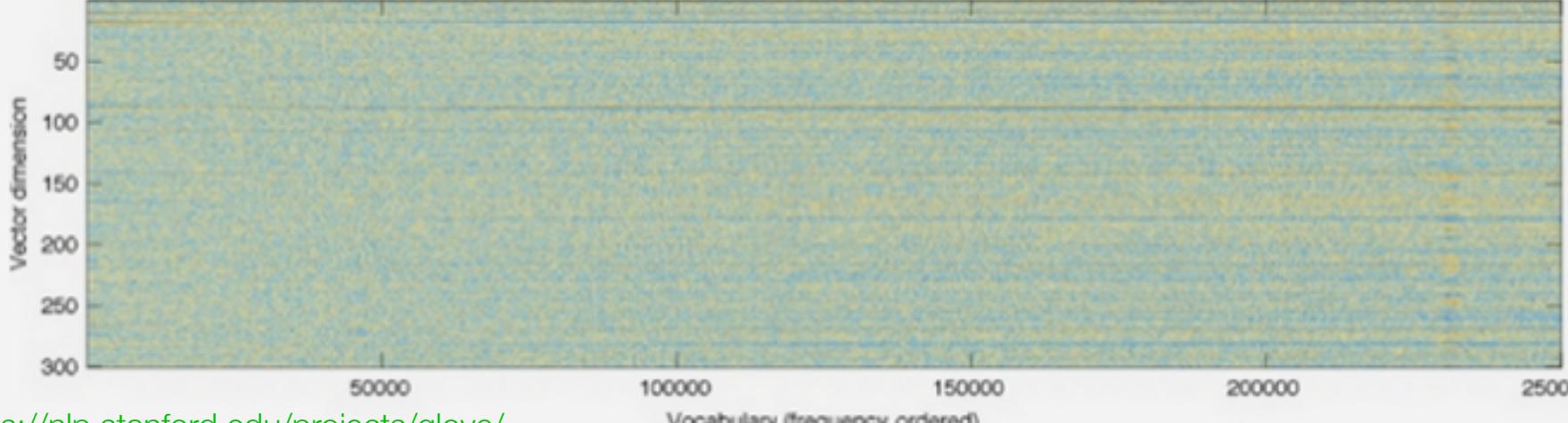
5. *rana*



7. *eleutherodactylus*

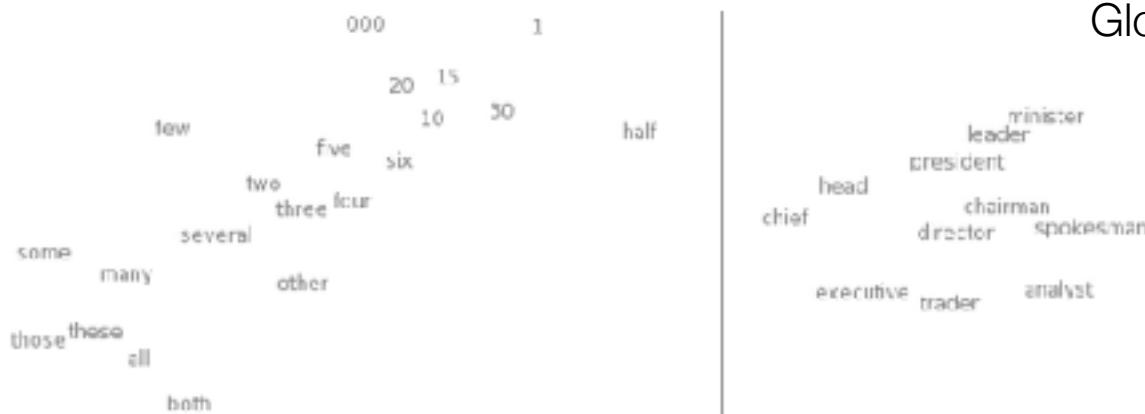
Global Vectors for Word Representation

GloVe produces word vectors with a marked banded structure that is evident upon visualization:



# Word Embeddings: proximity

GloVe



t-SNE visualizations of word embeddings. Left: Number Region; Right: Jobs Region. From Turian *et al.* (2010), see complete image.

FRANCE	JESUS	XBOX	REDDISH	SRAATCHED	MEGABITS
AUSTRIA	COD	AMIGA	GREENISH	NAILED	CCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/s
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	MHZ/s
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	psNUMBER	GREYISH	SCRAPED	KBIT/s
NORWAY	VISHNU	HD	GRAVISH	SCREWED	MEGAHRRTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/s
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

What words have embeddings closest to a given word? From Collobert *et al.* (2011)

<http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/>

The **chairman** called the **meeting** to order.

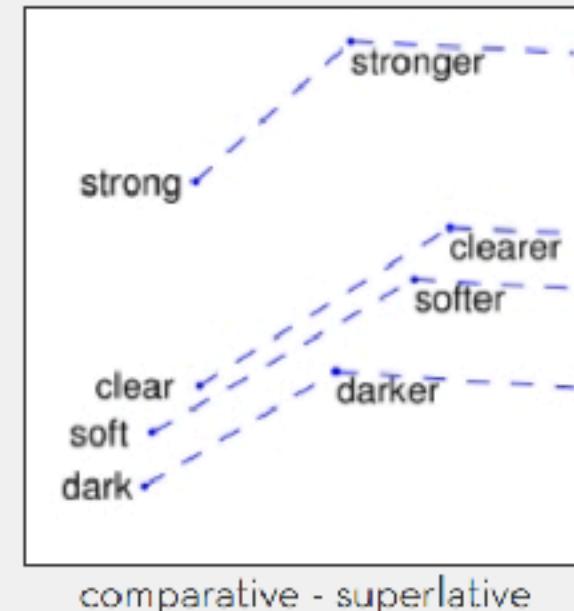
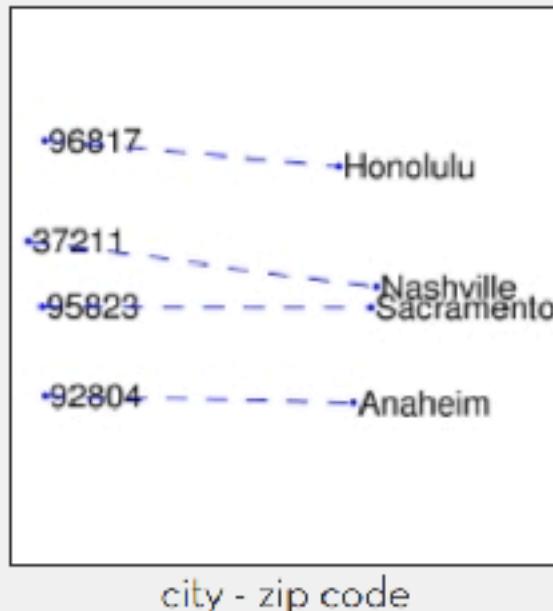
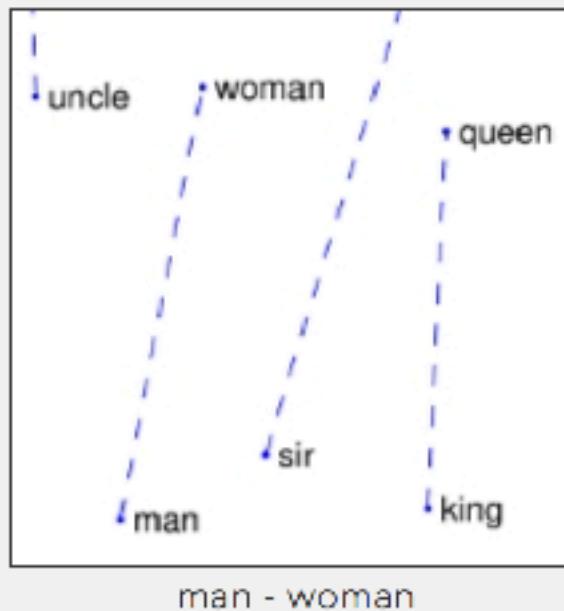
The **director** called the **conference** to order.

The **chief** called the **council** to order.

# Word Embeddings: Analogy

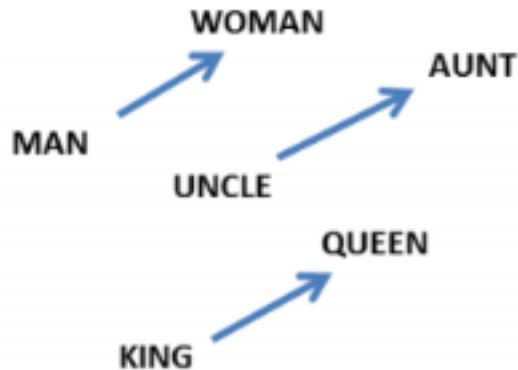
Glove

Global Vectors for Word Representation



each axis **might** encode a different type of relationship

# Word Embeddings: Analogy



From Mikolov *et al.*  
(2013a)

$$W(\text{"woman"}) - W(\text{"man"}) \simeq W(\text{"aunt"}) - W(\text{"uncle"})$$

$$W(\text{"woman"}) - W(\text{"man"}) \simeq W(\text{"queen"}) - W(\text{"king"})$$

## Glove

Global Vectors for Word Representation

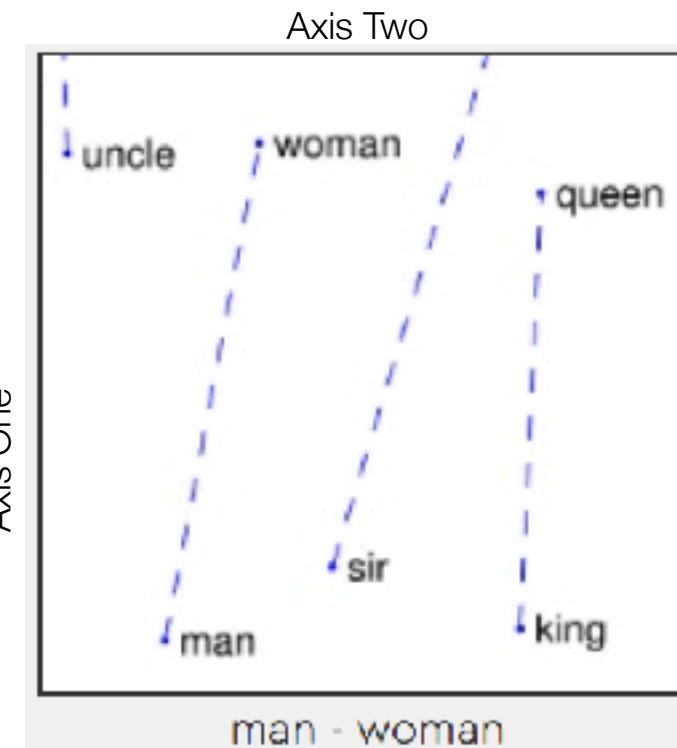
Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Relationship pairs in a word embedding. From Mikolov *et al.* (2013b).

<https://nlp.stanford.edu/projects/glove/> | <http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/>

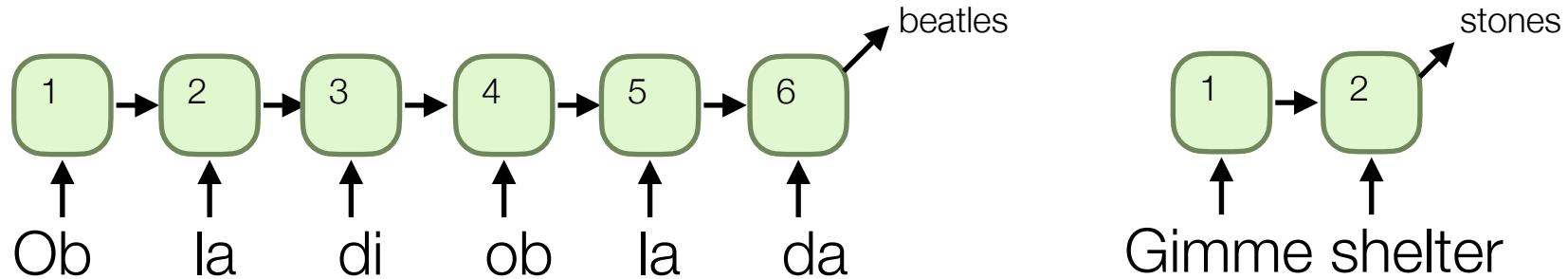
# Self Test: Analogy

- Each axis on the embedding plot below corresponds to:
- A. a weight inside the embedding layer
- B. an average of weights inside the embedding layer
- C. the average of the one hot encoding for a word
- D. an output of the embedding layer



# Practical Logistics: Sequence Length

- option A: dynamic length sequences



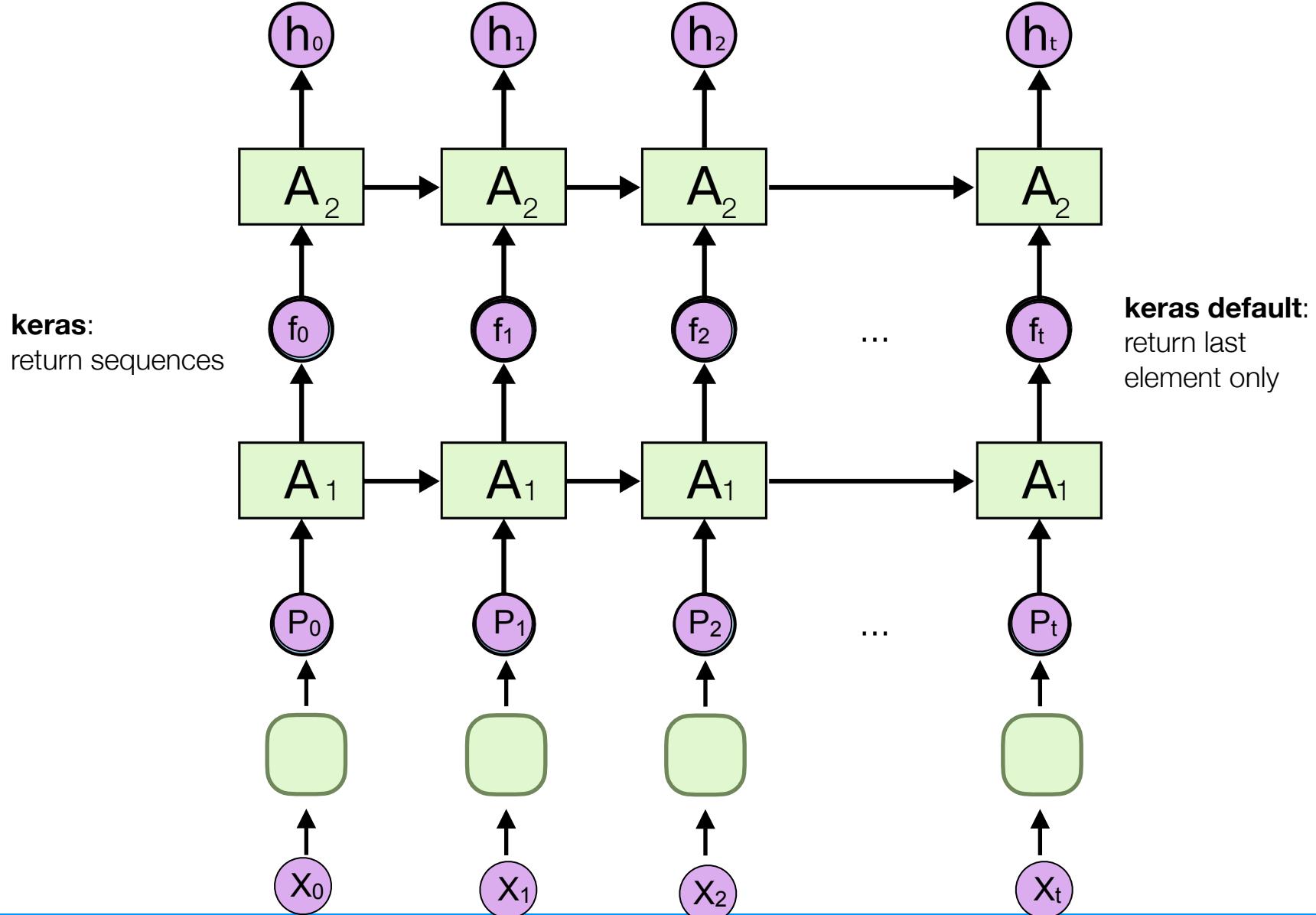
- option B: padding/clipping



- main difference:

**speed based on computation graph design**

# Sequence Stacking



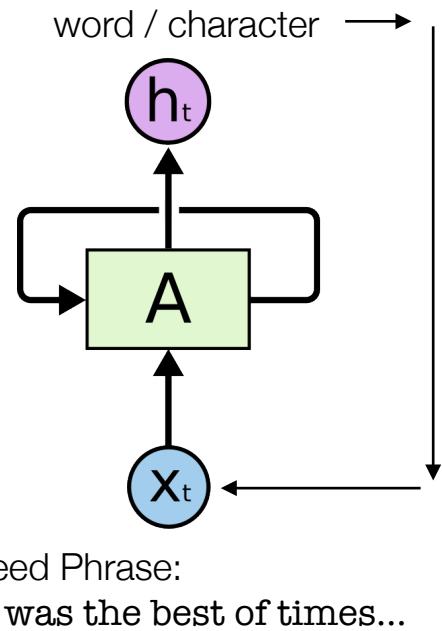
# Recurrent Generation



WHEN VISITING A NEW HOUSE, IT'S  
GOOD TO CHECK WHETHER THEY HAVE  
AN ALWAYS-ON DEVICE TRANSMITTING  
YOUR CONVERSATIONS SOMEWHERE.

# Generating Outputs

- Highly sophisticated steps:
  - train an RNN to generate the **next** word/character from the **current** word/character
  - train on a corpus of text
    - Shakespeare
    - Movie Scripts
    - Whatever!
  - seed with random word, feed output words as input to next node
  - rinse, repeat



# Training Data: Plays by Shakespeare

First Citizen:

Before we proceed any further, hear me speak.

All:

Speak, speak.

First Citizen:

You are all resolved rather to die than to famish?

All:

Resolved. resolved.

First Citizen:

First, you know Caius Marcius is chief enemy to the people.

All:

We know't, we know't.

First Citizen:

Let us kill him, and we'll have corn at our own price.

Is't a verdict?

All:

No more talking on't; let it be done: away, away!

Second Citizen:

One word, good citizens.

ervices he has done for his country?

First Citizen:

Very well; and could be content to give him good report fort, but that he pays himself with being proud.

Second Citizen:

Nay, but speak not maliciously.

First Citizen:

We are accounted poor citizens, the patricians g What authority surfeits on would relieve us: if would yield us but the superfluity, while it wer wholesome, we might guess they relieved us human but they think we are too dear: the leanness tha afflicts us, the object of our misery, is as an inventory to particularise their abundance; our sufferance is a gain to them Let us revenge this our pikes, ere we become rakes: for the gods kno speak this in hunger for bread, not in thirst fo

Second Citizen:

Would you proceed especially against Caius Marci

All:

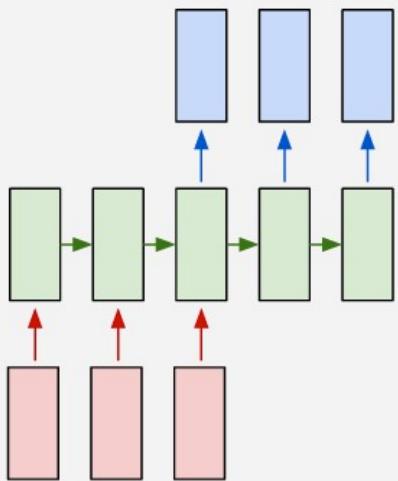
Against him first: he's a very dog to the common

Second Citizen:

Consider you what s

# Generating Shakespeare

many to many



ATOOOS

UIEAOUYOUZZZZZUZAAAYAYf n fsflflrurctuateot t ta's a wtut  
Whith then, a do makes and them and to sees,  
I wark on this ance may string take thou honon  
To sorriccorn of the bairer, whither, all  
I'd see if yiust the would a peid.

LARYNGLe:

To would she troust they fould.

PENMES:

Thou she so the havin to my shald woust of  
As tale we they all my forder have  
As to say heant thy wansing thag and  
Whis it thee shath his breact, I be and might, she  
Tirs you desarvishensed and see thee: shall,  
What he hath with that is all time,  
And sen the have would be sectiens, way thee,  
They are there to man shall with me to the mon,  
And mere fear would be the balte, as time an at  
And the say oun touth, thy way womers thee.

# Generating Movies

**more data:** star wars + star trek + tarantino + the matrix

DENT 'SUEENCK

Bartholomew of the TIE FIGHTERS are stunned. There is a crowd and armored  
switcheroos.

PICARD

(continuing)

Couns two dim is tired. In order to the sentence...

The sub bottle appears on the screen into a small shuttle shift of the  
ceiling. The DAMBA FETT splash fires and matches them into the top, transmit to stable high above  
upon their statels,  
falling from an alien shaft.

ANAKIN and OBI-WAN stand next to OBI-WAN down the control plate of smoke at the TIE fighter. They  
stare at the centre of the station loose into a comlink cover -- comes up to the General, the  
GENERAL HUNTAN AND FINNFURMBARD from the PICADOR to a beautiful Podracisly.

ENGINEER

Naboo from an army seventy medical  
security team area re-weilergular.

EXT.

THE MULTIVERSE —

# Movie written by algorithm turns out to be hilarious and intense

For *Sunspring*'s exclusive debut on Ars, we talked to the filmmakers about collaborating with an AI.

ANNALEE NEWITZ - 6/9/2016, 5:30 AM



# Next time

- Recurrent Networks
  - *Overview*
  - *Problem Types*
  - *Embeddings*
  - **Commonly Used RNNs Nodes**
  - **Demo A**
  - **CNNs and RNNs**
  - **Demo B**
  - **Ethics Case Study**
  - **Course Retrospective**

# Lecture Notes for **Machine Learning in Python**

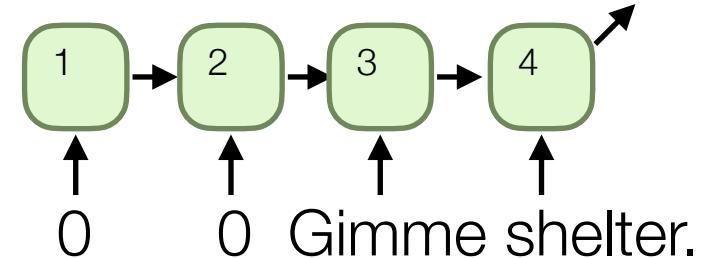
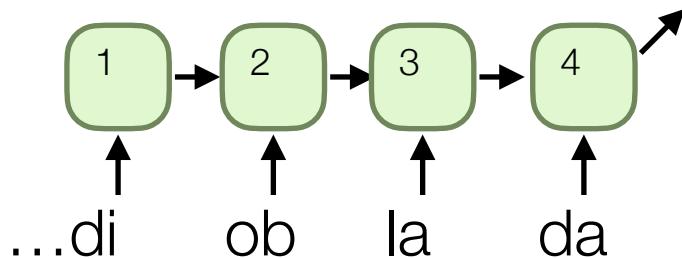
Professor Eric Larson  
**Lecture: RNN Demo**

# Lecture Agenda

- Logistics
  - RNNs due **During Finals Time**
- Recurrent Networks
  - Overview
  - *Problem Types*
  - *Embeddings*
  - *Types of RNNs*
  - **Demo A**
  - **CNNs and RNNs**
  - **Demo B**
  - **Ethics Case Study**
  - **Course Retrospective**

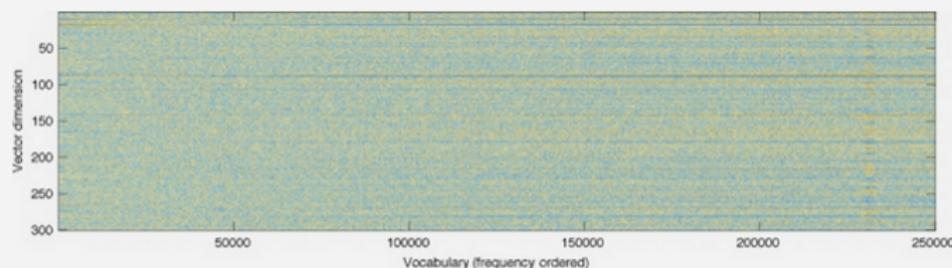
# Last Time

- padding/clipping



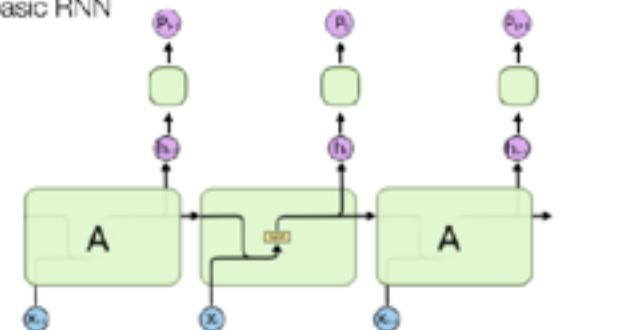
## Visualization

GloVe produces word vectors with a marked banded structure that is evident upon visualization:



## Recurrent Networks

- basic RNN

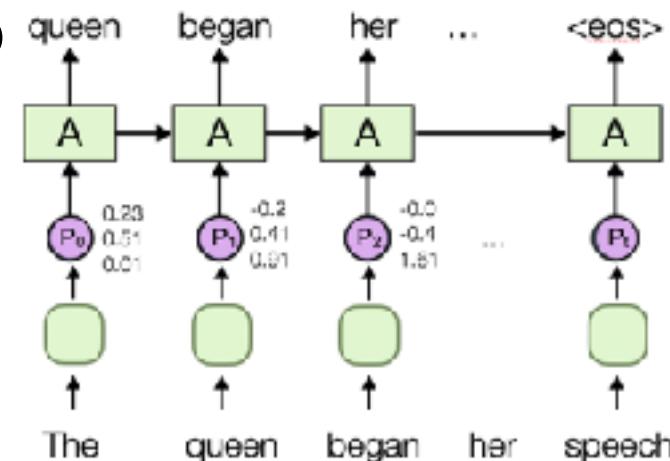


$$h_t = \tanh(W_A(X_t @ h_{t-1}) + b_A)$$
$$p_t = \text{softmax}(W_p h_t + b_p)$$

<https://colah.github.io/posts/2015-08-Understanding-RNNs/>

# Self Test

- T/F: In Recurrent Neural Networks that are “rolled out”, each RNN cell can be run in parallel.
  - A. **True**, state vectors can be added later
  - B. **True**, but parallelization must use forward backward (like Viterbi)
  - C. **False**, state vectors must be found sequentially
  - D. **False**, input changes due to sequential nature of  $X_t$

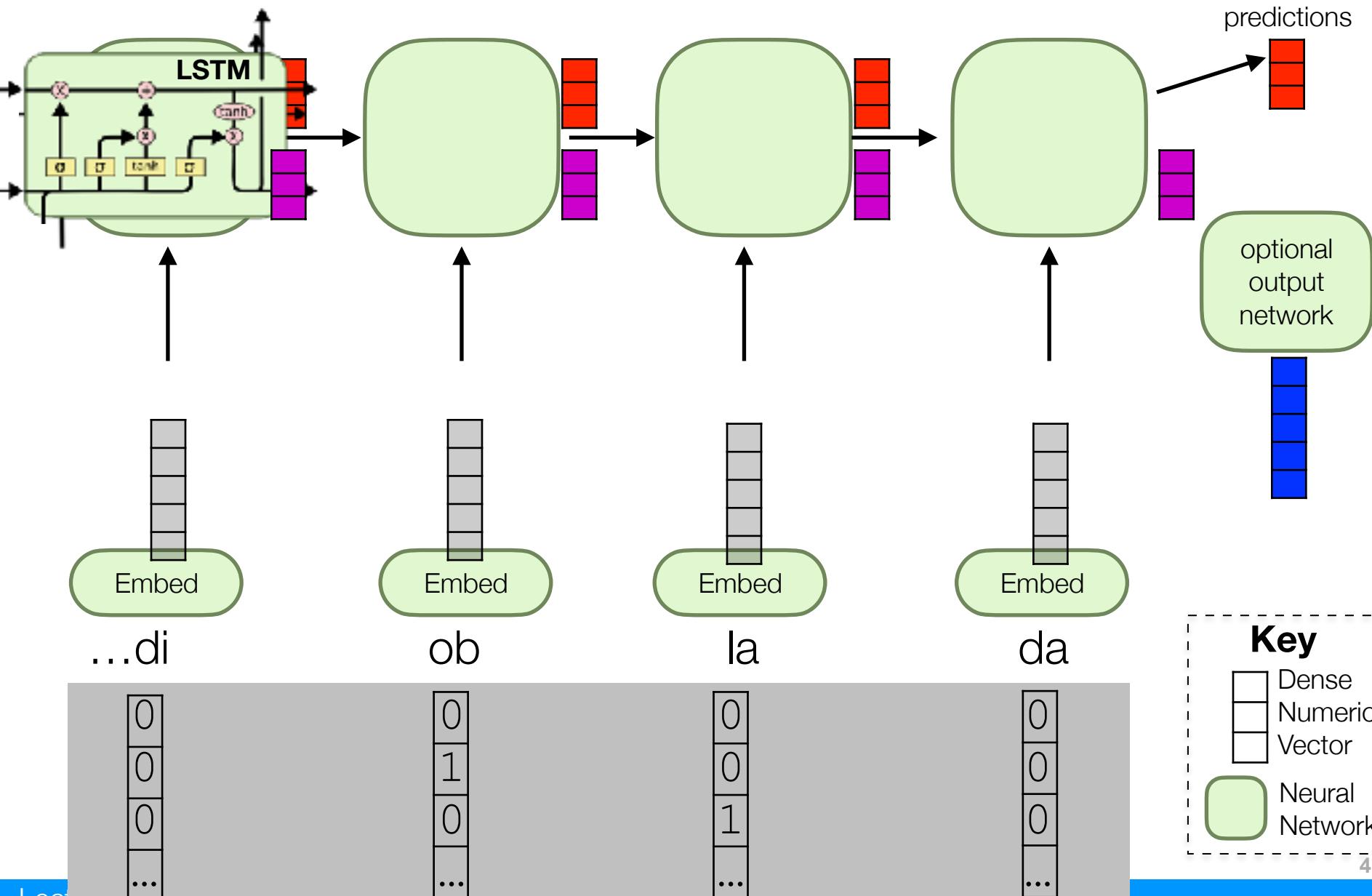


# Commonly Used RNN Nodes



I like this version better.

# General recurrent flow (many to one)



# Recurrent Networks: GRUs

- gated recurrent units

Selectivity controls, gates (**0 or 1**)

$$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r)$$

$$z_t = \sigma(W_z s_{t-1} + U_z x_t + b_z)$$

past state

current input

selectively remember

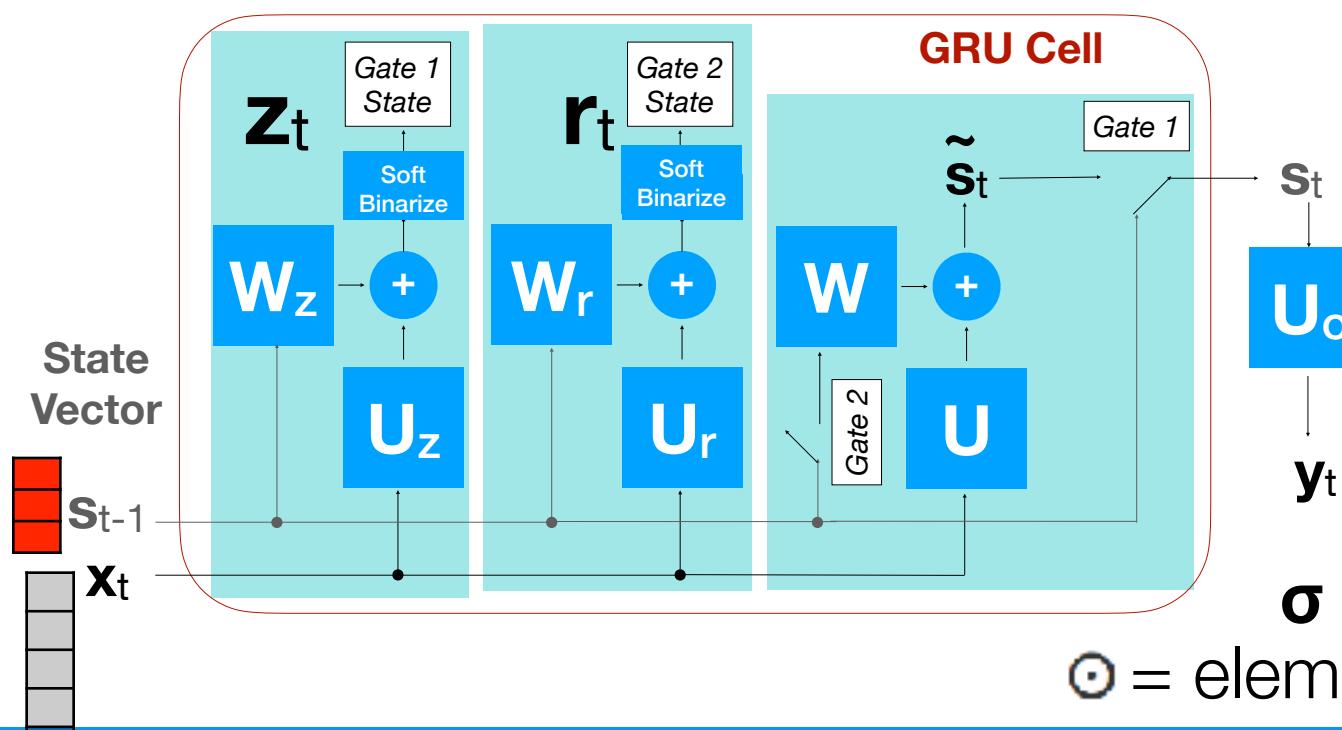
with influence

$$\tilde{s}_t = \phi(W(r_t \odot s_{t-1}) + U x_t + b)$$

$$s_t = z_t \odot s_{t-1} + (1 - z_t) \odot \tilde{s}_t$$

remember only past

OR remember with input



$\sigma$  = sigmoid

$\odot$  = elem. multiplication

# Self Test

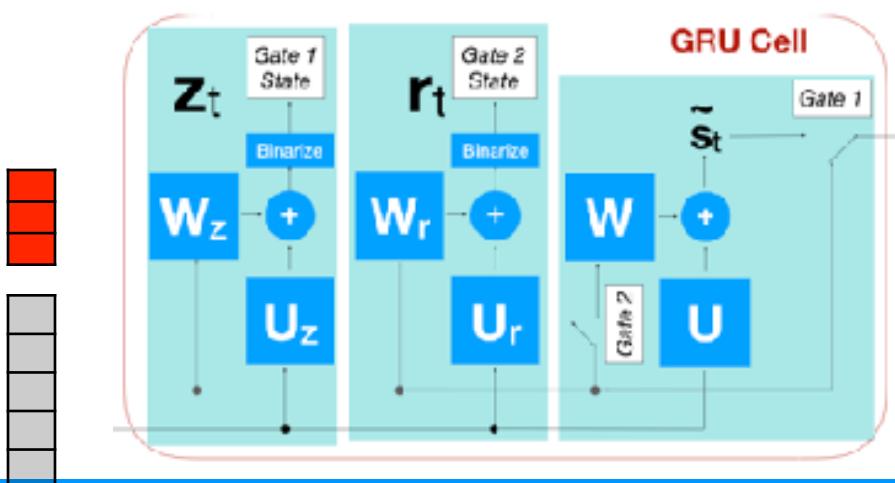
- What element of the GRU helps with vanishing and exploding gradients?
- A. derivative of  $\sigma$
- B. no activation function
- C. derivative of  $\phi$
- D.  $\phi$

$$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r)$$

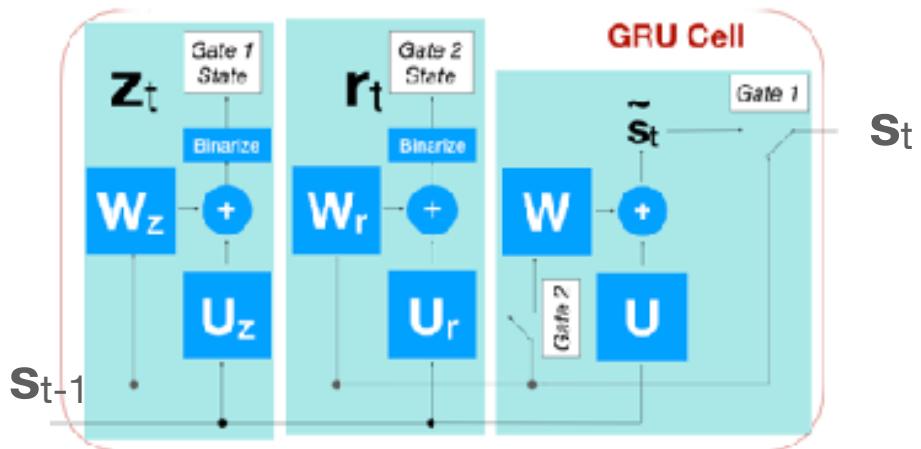
$$z_t = \sigma(W_z s_{t-1} + U_z x_t + b_z)$$

$$\tilde{s}_t = \phi(W(r_t \odot s_{t-1}) + Ux_t + b)$$

$$s_t = z_t \odot s_{t-1} + (1 - z_t) \odot \tilde{s}_t$$



# Derivative of GRU



$$r_t = \sigma(W_r s_{t-1} + U_r x_t + b_r)$$
$$z_t = \sigma(W_z s_{t-1} + U_z x_t + b_z)$$
$$\tilde{s}_t = \phi(W(r_t \odot s_{t-1}) + Ux_t + b)$$
$$s_t = z_t \odot s_{t-1} + (1 - z_t) \odot \tilde{s}_t$$

To back propagate, we need sensitivity of state vector, w.r.t previous state

$$\frac{\partial s_t}{\partial s_{t-1}} = (\partial z_t / \partial s_{t-1}) + (\partial s_{t-1} / \partial z_t) + \frac{\partial \tilde{s}_t}{\partial s_{t-1}} - (\partial z_t / \partial \tilde{s}_t) - (\partial \tilde{s}_t / \partial z_t)$$

Product Rule

Product Rule

likely vanish

could vanish, depending on  $\phi$

likely vanish

could vanish, depending on  $\phi$

hard to vanish unless  $z_t = 0$

# Recurrent Networks: Gen 1 LSTM

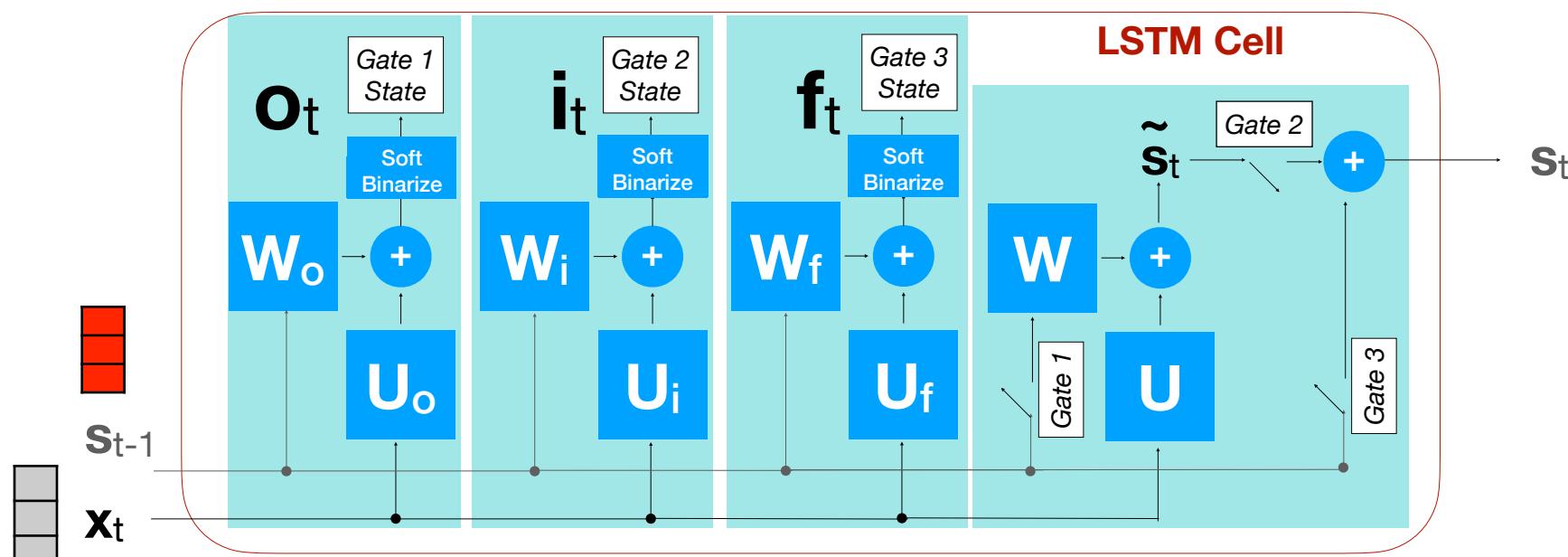
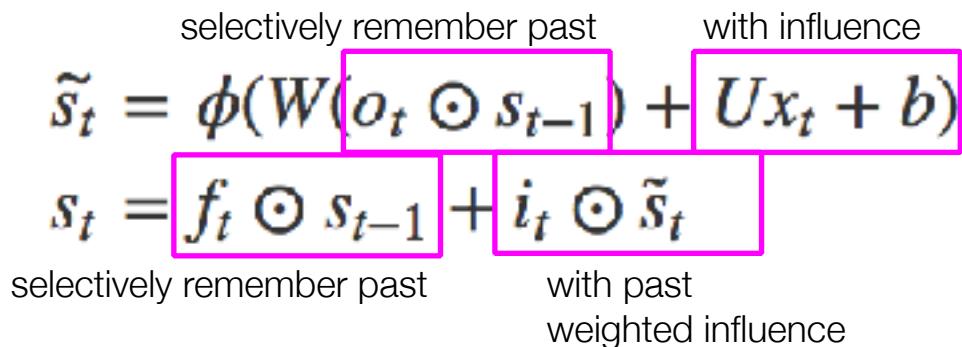
- LSTM prototype

## Selectivity controls (**gates**, 0 or 1)

$$o_t = \sigma(W_o s_{t-1} + U_o x_t + b_o)$$

$$i_t = \sigma(W_i s_{t-1} + U_i x_t + b_i)$$

$$f_t = \sigma(W_f s_{t-1} + U_f x_t + b_f)$$



# Recurrent Networks: Gen 2 LSTM

- LSTM in TensorFlow
- Selectivity controls (**gates, 0 or 1**)

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$

explicit remembering state

$$\tilde{c}_t = \phi(W h_{t-1} + U x_t + b)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

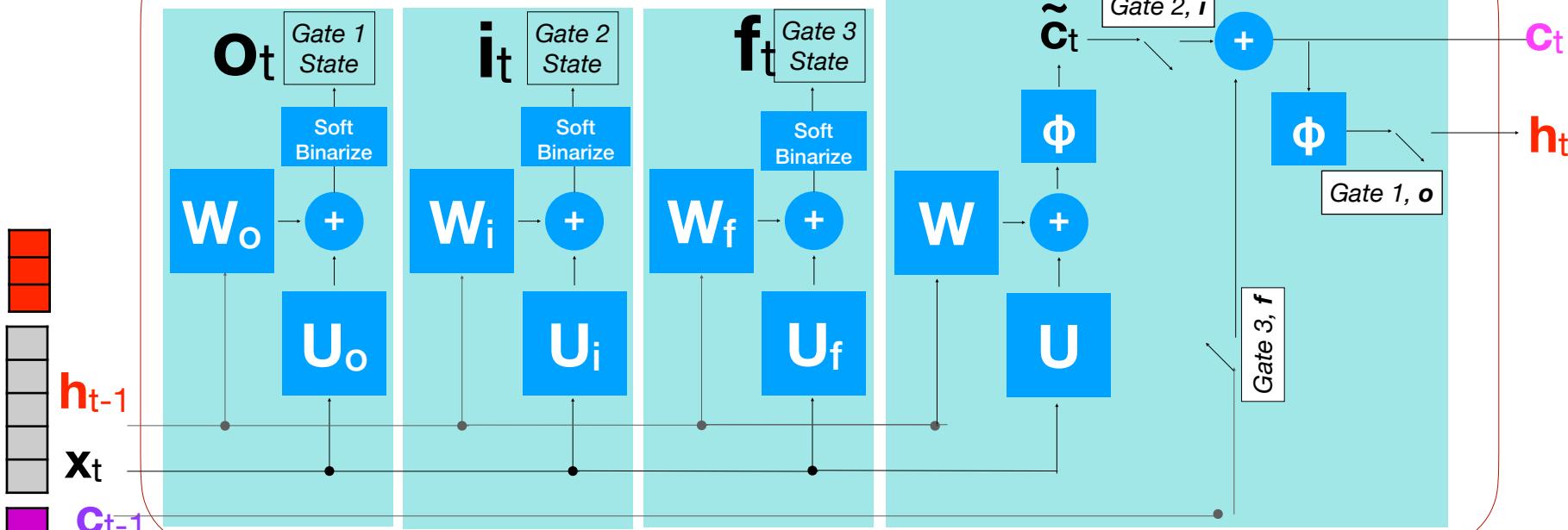
remember previous state

update with output,  $h_t$

$$h_t = o_t \odot \phi(c_t)$$

get next ht for selecting gates

LSTM Cell, Tensorflow



# LSTM Dropout

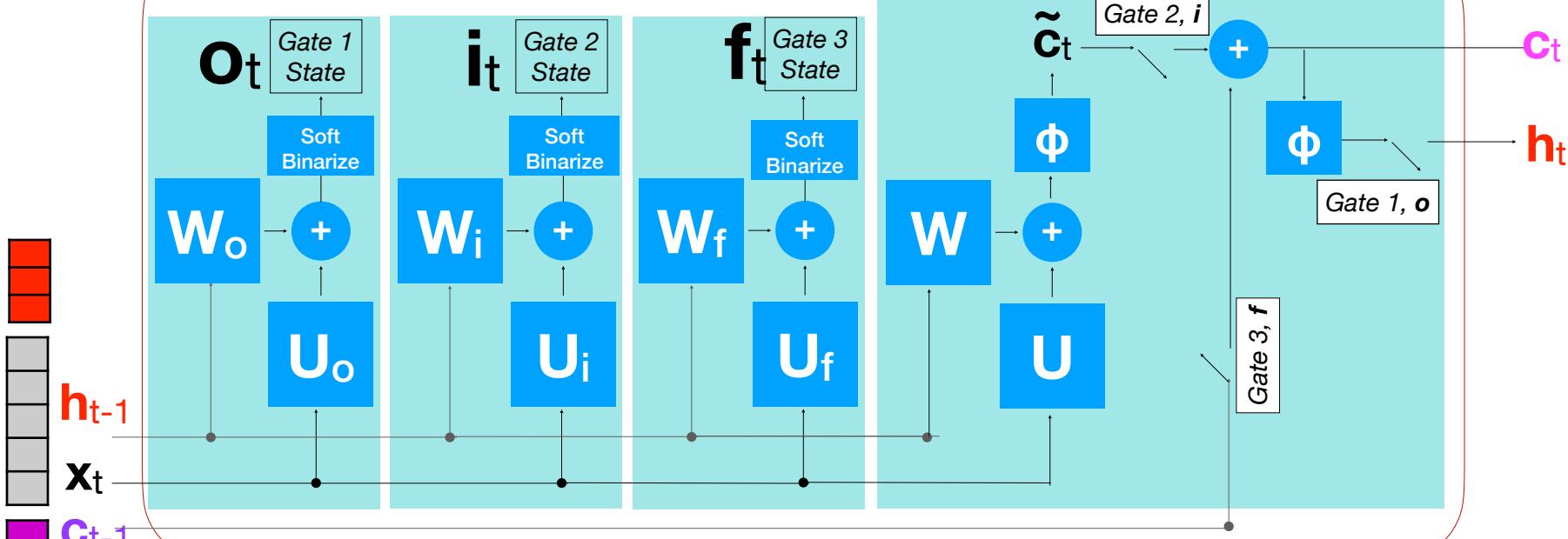
$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$
$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$
$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$

Recurrent  
Dropout

Input  
Dropout

The days of  
training **without**  
using **dropout** are  
**over.**

## LSTM Cell, Tensorflow



# What to choose?

- There is no hard and fast rule
  - try both
  - basic LSTM has had great success
  - GRU also sometimes is easier to train
  - you will see many variations
    - peephole LSTM
    - hierarchical LSTM
    - and many more...

Many to one:  
Simple RNNs  
GRUs  
LSTMs



More examples:

<https://github.com/tensorflow/tensorflow/tree/r0.11/tensorflow/examples/skflow>

<http://r2rt.com/recurrent-neural-networks-in-tensorflow-i.html>

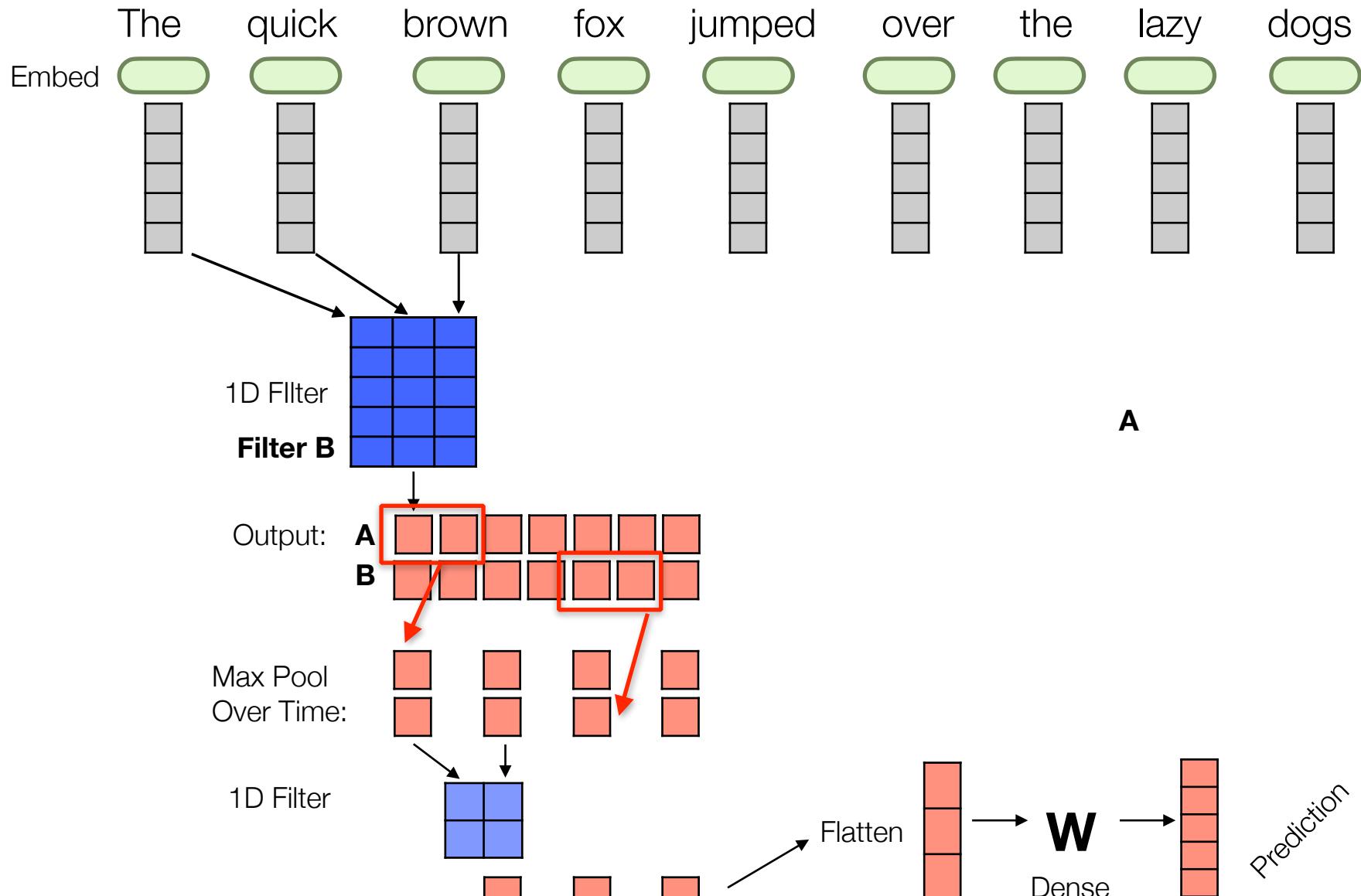
<http://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/>

Seq2Seq:

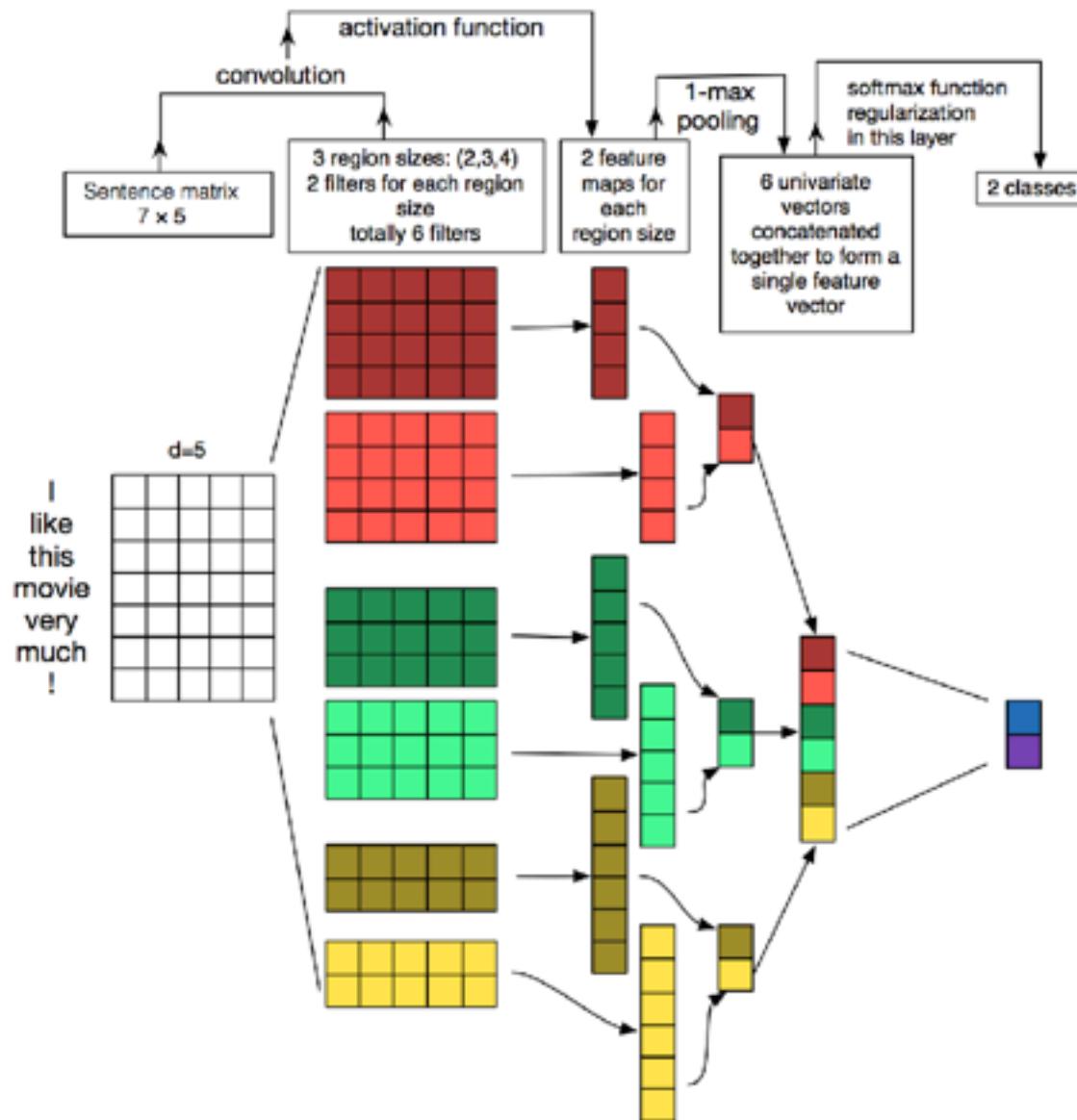
[https://github.com/tensorflow/tensorflow/blob/r0.11/tensorflow/examples/skflow/neural\\_translator\\_word.py](https://github.com/tensorflow/tensorflow/blob/r0.11/tensorflow/examples/skflow/neural_translator_word.py)

# CNNs and RNNs

is an RNN similar to a CNN?



# CNNs with Multiple Region Sizes



Back to the CNN



More examples:

<http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/>

Seq2Seq:

[https://github.com/tensorflow/tensorflow/blob/r0.11/tensorflow/examples/skflow/neural\\_translation\\_word.py](https://github.com/tensorflow/tensorflow/blob/r0.11/tensorflow/examples/skflow/neural_translation_word.py)

# Ethics and Bias Case Study in NLP



**Janelle Shane** @JanelleCShane · 1d

Predictive policing algorithms don't predict who commits crime. They predict who the police will arrest.

So of course the algorithm points toward people that are already overpoliced - it's trying to predict racism. Don't explicitly tell it race & it will just use other proxies.



**Timnit Gebru** ✅  
@timnitGebru

I'm sick of this framing. Tired of it. Many people have tried to explain, many scholars. Listen to us. You can't just reduce harms caused by ML to dataset bias.



**Yann LeCun** @ylecun · 19h

ML systems are biased when data is biased. This face upsampling system makes everyone look white because the network was pretrained on FlickrFaceHQ, which mainly contains white people pics....

# Word Embeddings: Analogy?

$$W(\text{"woman"}) - W(\text{"man"}) \approx W(\text{"aunt"}) - W(\text{"uncle"})$$

$$W(\text{"woman"}) - W(\text{"man"}) \approx W(\text{"queen"}) - W(\text{"king"})$$

$$\overrightarrow{\text{man}} - \overrightarrow{\text{woman}} \approx \overrightarrow{\text{computer programmer}} - \overrightarrow{\text{homemaker}}.$$

Trained on  
New York Times



<https://nlp.stanford.edu/projects/glove/>

## Extreme *she* occupations

- |                 |                       |                        |
|-----------------|-----------------------|------------------------|
| 1. homemaker    | 2. nurse              | 3. receptionist        |
| 4. librarian    | 5. socialite          | 6. hairdresser         |
| 7. nanny        | 8. bookkeeper         | 9. stylist             |
| 10. housekeeper | 11. interior designer | 12. guidance counselor |

## Extreme *he* occupations

- |                |                   |                |
|----------------|-------------------|----------------|
| 1. maestro     | 2. skipper        | 3. protege     |
| 4. philosopher | 5. captain        | 6. architect   |
| 7. financier   | 8. warrior        | 9. broadcaster |
| 10. magician   | 11. fighter pilot | 12. boss       |

Bolukbasi et al., NeurIPS 2016

<https://arxiv.org/pdf/1607.06520.pdf>

# ConceptNet

## en cooking dinner

An English term in ConceptNet 5.7

Source: Open Mind Common Sense contributors

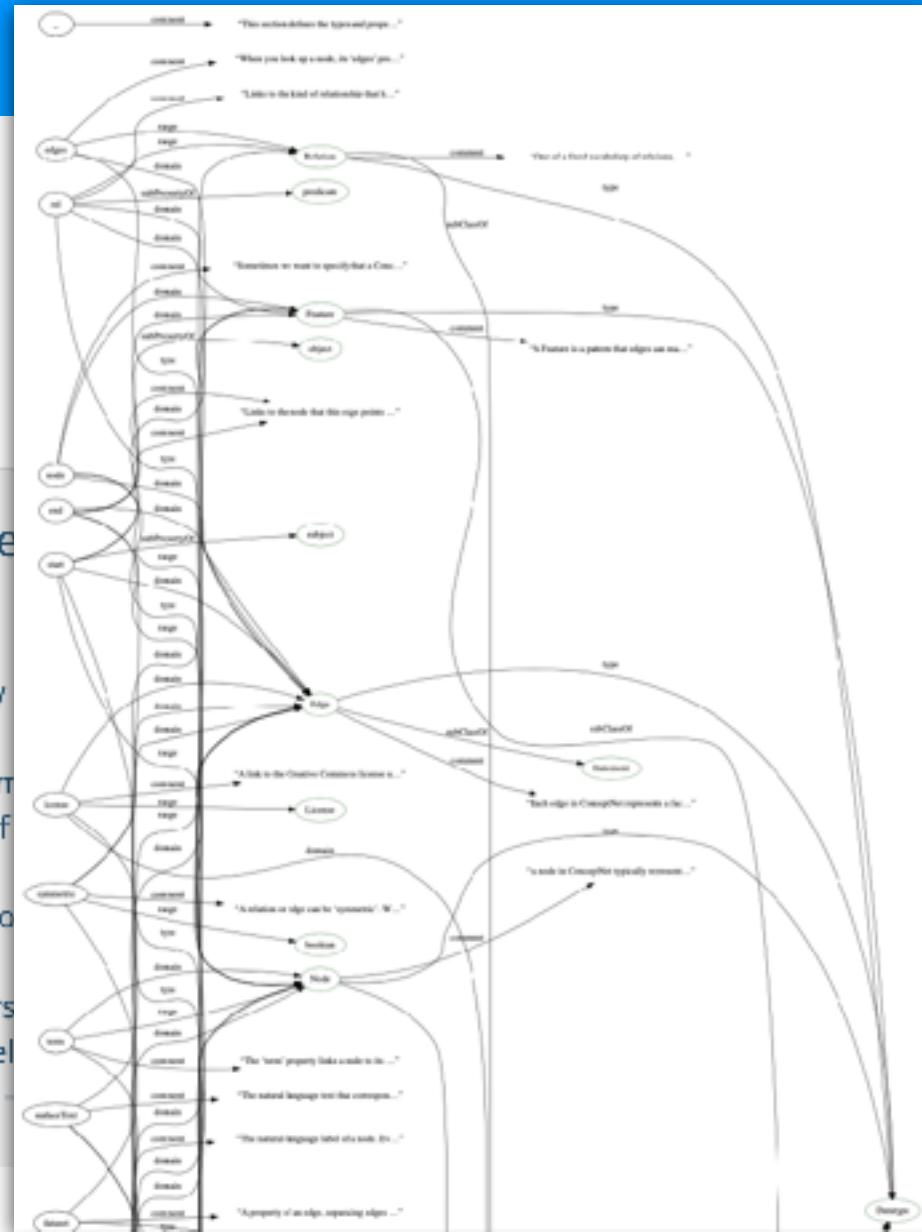
[View this term in the API](#)

cooking dinner is a  
subevent of...

- en boiling water →
- en it burns →
- en preheat the oven →
- en taste the food →
- en boil salt water →
- en boil water →
- en brown the hamburger →
- en chop a vegetable →
- en defrost →
- en a fire →
- en the fire alarm might go off →

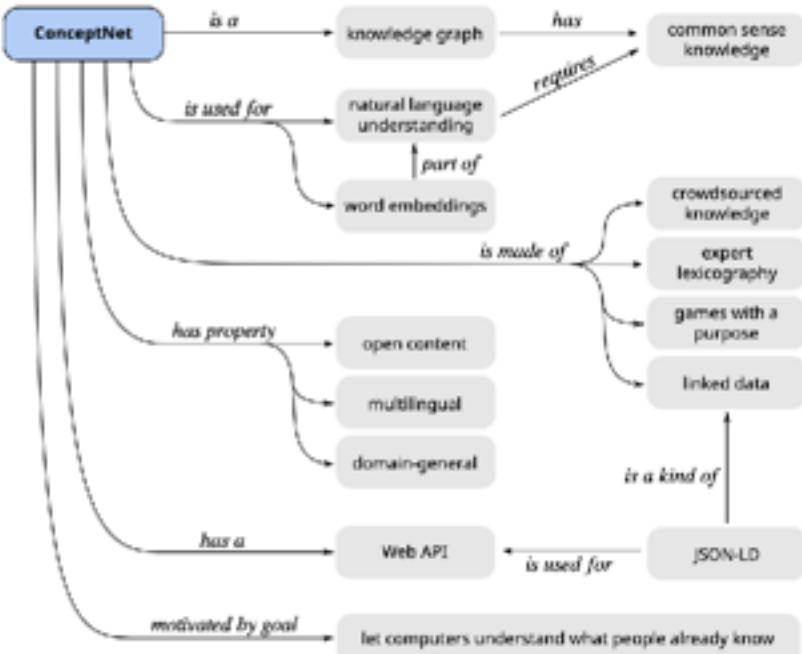
cooking dinner  
for...

- en feeding a family
- en TO EAT →
- en entertaining com
- en feeding yourself
- en anyone →
- en avoiding fast foo
- en being a cook →
- en caring for others
- en cheering yourself
- en creative people
- en eating →



# ConceptNet Numberbatch

- Create with a Knowledge Graph (from multiple sources with relations like *UsedFor*, *PartOf*, etc.)
- Based on this KG, perturb existing embeddings (like GloVe) to optimize:



$$\Psi(Q) = \sum_{i=1}^n \left[ \alpha_i \|q_i - \hat{q}_i\|^2 + \sum_{(i,j) \in E} \beta_{ij} \|q_i - q_j\|^2 \right]$$

Annotations below the equation explain the terms:

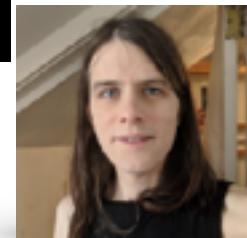
- new embed**: Points to  $\hat{q}_i$  in the first term.
- old embed**: Points to  $q_i$  in the first term.
- neighbors from KG**: Points to the second term  $\sum_{(i,j) \in E} \beta_{ij} \|q_i - q_j\|^2$ .
- (keep similar to original)**: A double-headed arrow below the first term indicates it aims to keep the new embedding similar to the old one.
- (make similar according to other knowledge)**: A double-headed arrow below the second term indicates it aims to make the new embedding similar to its neighbors based on the Knowledge Graph.

- Easy to optimize the objective by averaging neighbors in the ConceptNet KG
- Multiple embeddings achieved by merging through “retrofitting” which projects onto a shared matrix space (with SVD)

ConceptNet 5.5: An Open Multilingual Graph of General Knowledge, Speer et al., 2017



# How to Make a Racist AI without Really Trying



Robyn Speer, 2017

<http://blog.conceptnet.io/posts/2017/how-to-make-a-racist-ai-without-really-trying/>

**Debiasing: Man is to Computer  
Programmer as Woman is to  
Homemaker? Debiasing Word  
Embeddings**

Bolukbasi et al., NeurIPS 2016

<https://arxiv.org/pdf/1607.06520.pdf>

**ConceptNet 5.5: An Open  
Multilingual Graph of General  
Knowledge**

Speer et al., AAAI 2017

<https://arxiv.org/pdf/1612.03975.pdf>



Rachael Tatman @rctatman · 18h

I first got interested in ethics in NLP/ML because I was asking "does this system work well for everyone". It's a good question, but there's a more important important one:

Who is being harmed and who is benefiting from this system existing in the first place?



François Chollet · 11h  
When faced with tech ethics problems, you can either ask hard questions, seek solutions, and take responsibility, or you can lazily jump on the nearest intellectual shortcut that looks like it will save you.

# Town Hall

## Timnit Gebru

A lot of times, people are talking about bias in the sense of equalizing performance across groups. They're not thinking about the underlying foundation, whether a task should exist in the first place, who creates it, who will deploy it on which population, who owns the data, and how is it used?

The root of these problems is not only technological. It's social. Using technology with this underlying social foundation often advances the worst possible things that are happening. In order for technology not to do that, you have to work on the underlying foundation as well. You can't just close your eyes and say: "Oh, whatever, the foundation, I'm a scientist. All I'm going to do is math."

cial  
; entire  
en and  
is

7h  
is it take?  
e, books  
m experts  
se,  
cover  
munity  
matter.

ut ML

ployed...

# Lecture Notes for **Machine Learning in Python**

Professor Eric Larson  
**Seq-2-Seq and Transformers**

Archived

# Lecture Agenda

- ❑ Logistics
  - ❑ RNNs due **During Finals Time**
- ❑ Agenda
  - ❑ Sequence to sequence
  - ❑ Transformers

# Last Time

- LSTM prototype

Selectivity controls (**gates, 0 or 1**)

$$o_t = \sigma(W_o s_{t-1} + U_o x_t + b_o)$$

$$i_t = \sigma(W_i s_{t-1} + U_i x_t + b_i)$$

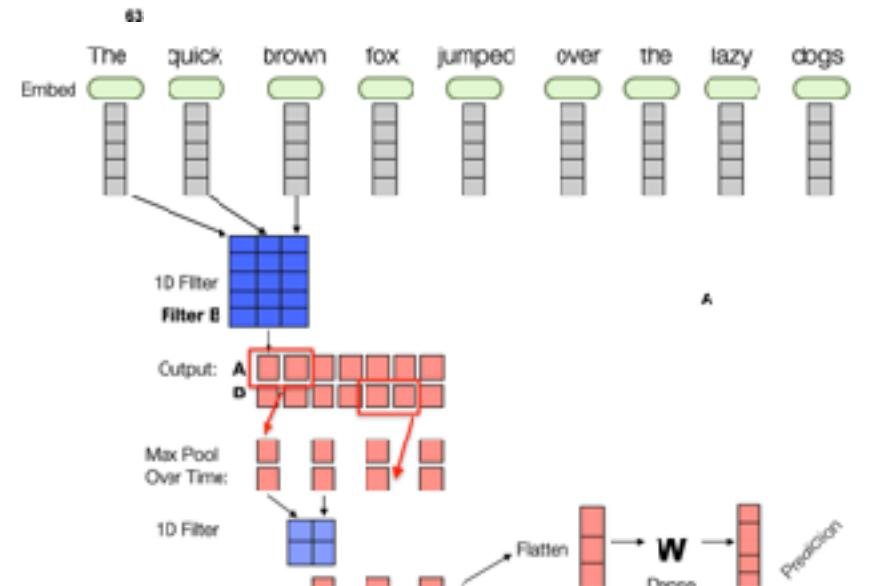
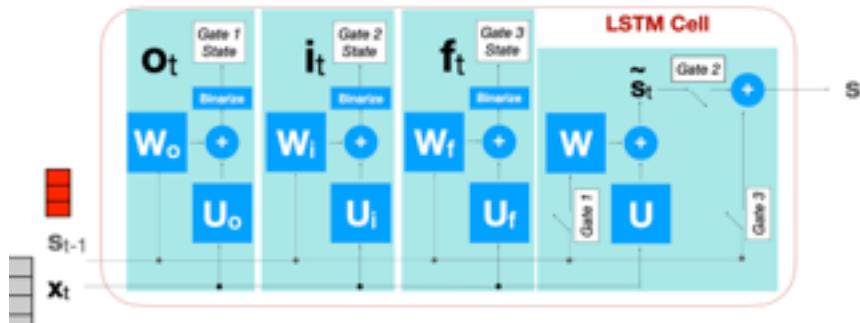
$$f_t = \sigma(W_f s_{t-1} + U_f x_t + b_f)$$

$$\tilde{s}_t = \phi(W(o_t \odot s_{t-1}) + Ux_t + b)$$

selectively remember past  
with influence

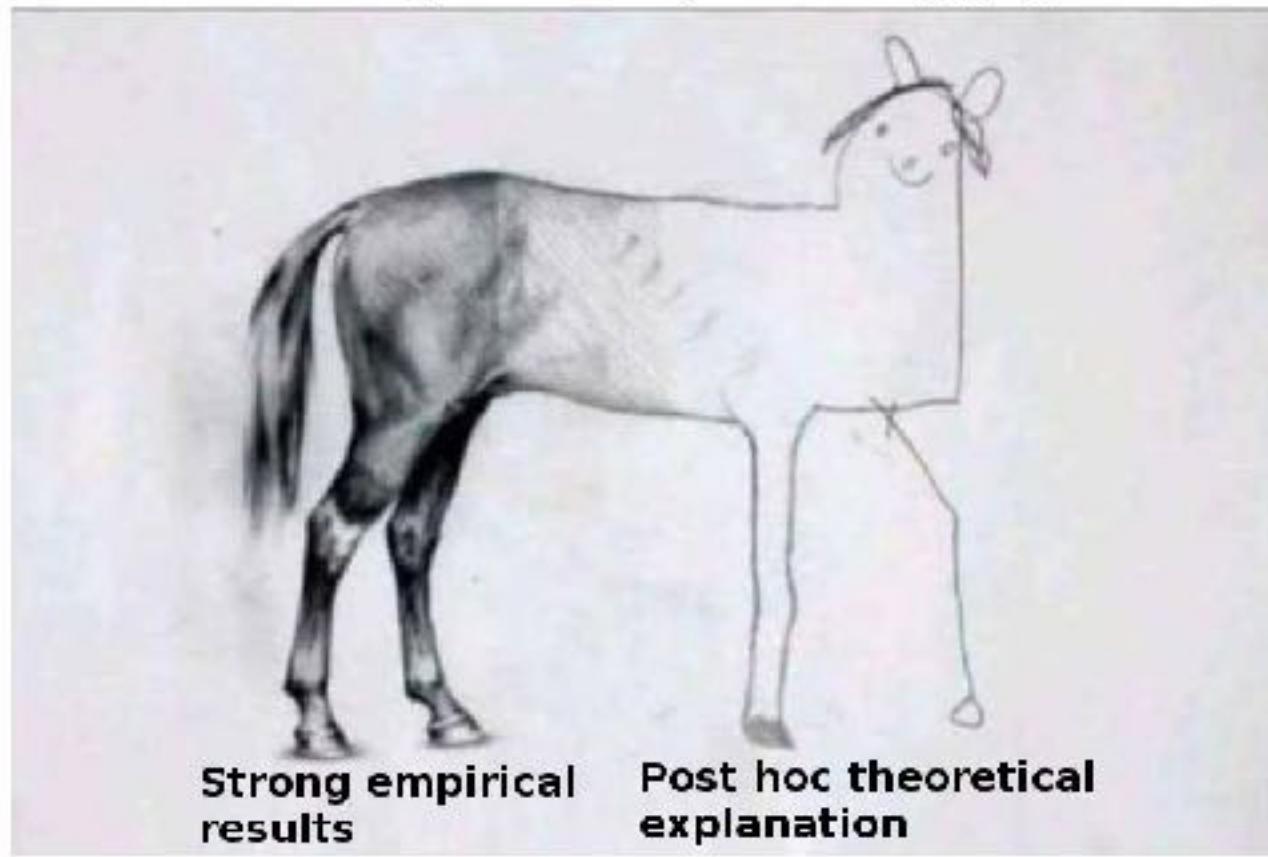
$$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s}_t$$

selectively remember past  
with past weighted influence



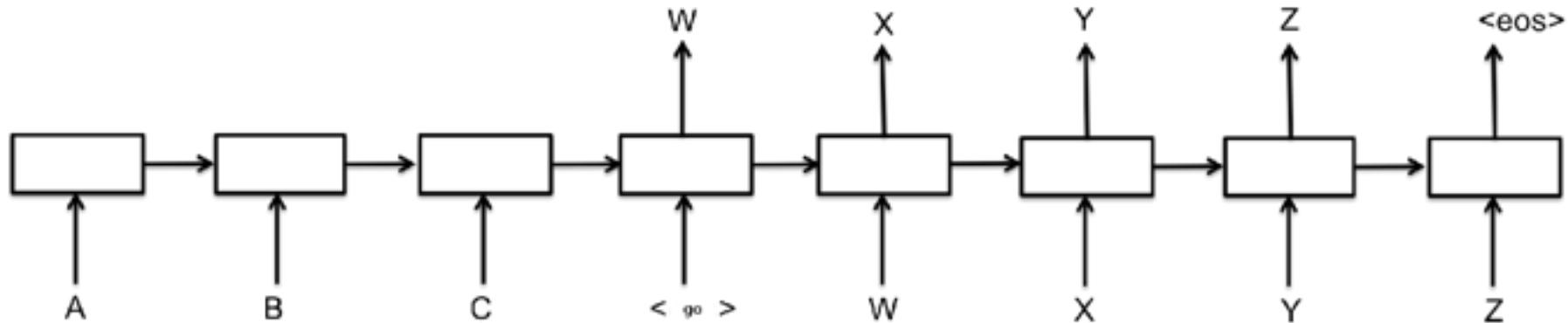
# Sequence to Sequence

Anatomy of a deep learning paper



# Modeling Sequence to Sequence

Need to translate outputs of unknown size.



- Additional Vocabulary Special Casing:
  - <UNKNOWN>, for unknown input or characters not included in vocabulary
  - <EOS>, end of sentence
  - <GO>, start output sequence
  - <DONTCARE>, outputs before <GO> command

Sutskever et al. Sequence to Sequence Learning with Neural Networks, arXiv. 2014  
<https://arxiv.org/pdf/1409.3215.pdf>

# Modeling Sequence to Sequence

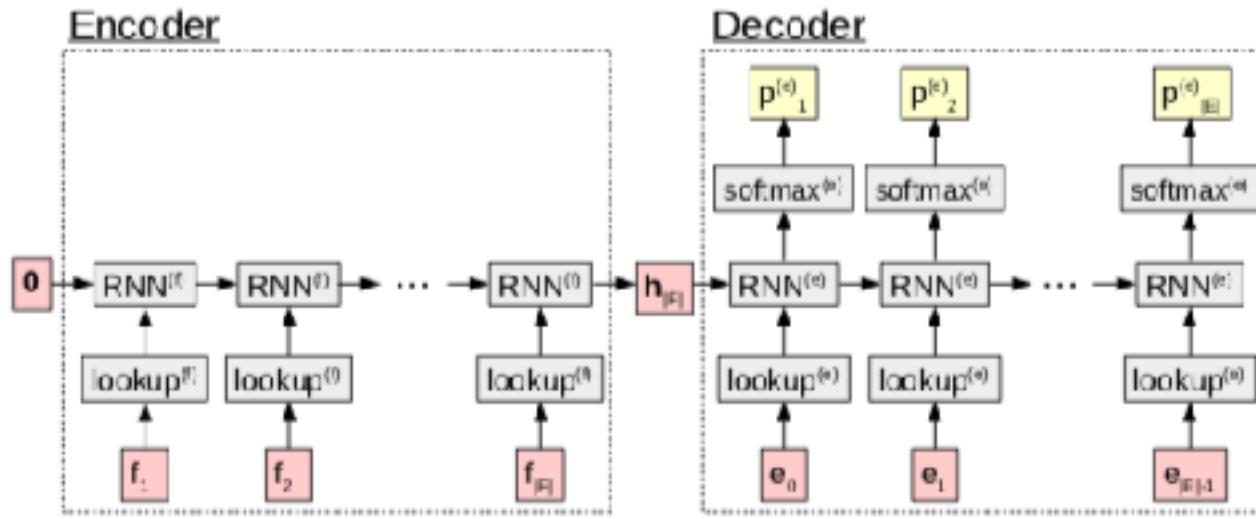


Figure 21: A computation graph of the encoder-decoder model.

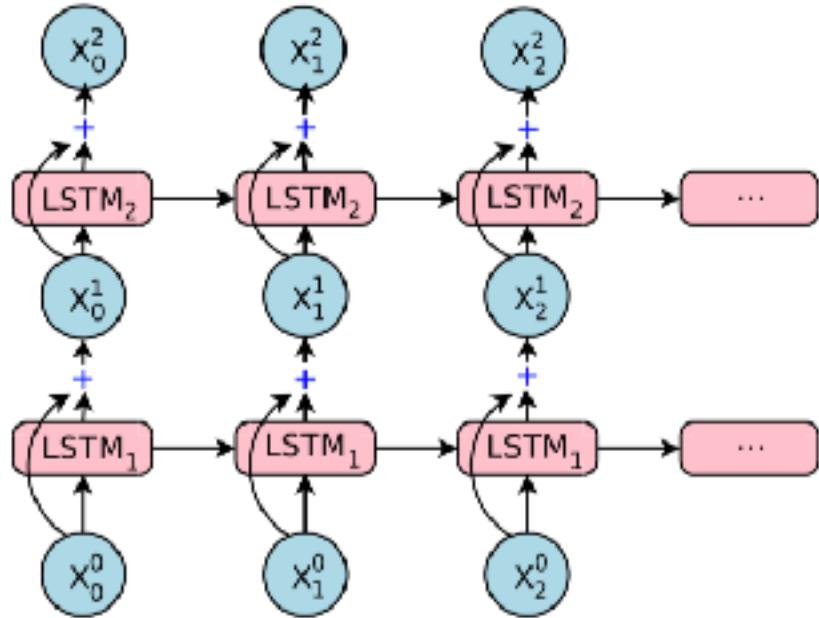
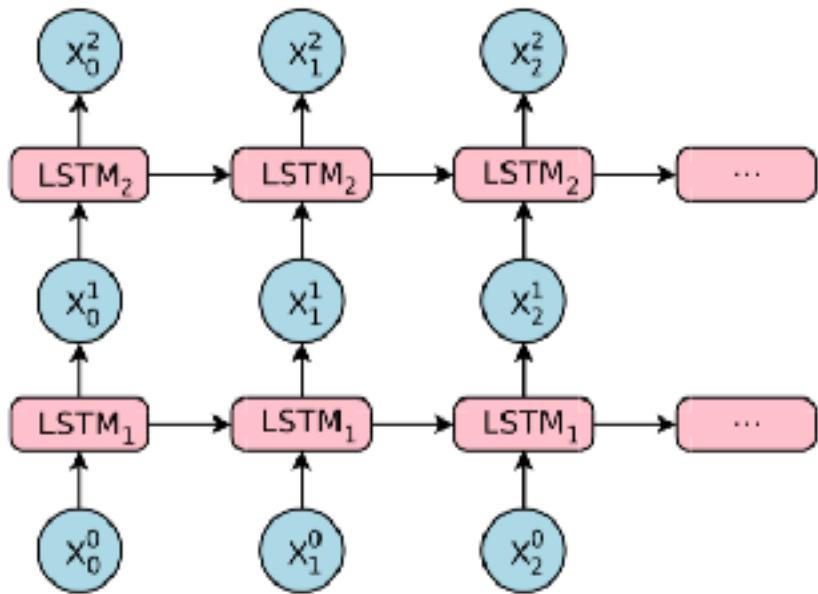
- **Training Process:** Give actual decoded letters for predicting next token
- **Decoding Process** can alter reliability of results:
  - Greedy Search, always choose most likely “next” symbol, seed
  - Keep list of “best” predictions for seeding (i.e., Beam Search)

Graham Neubig. 2017  
Neural Machine Translation and  
Sequence-to-sequence Models: A Tutorial  
<https://arxiv.org/pdf/1703.01619.pdf>

[https://github.com/m2dsupsdclclass/lectures-labs/blob/master/labs/07\\_seq2seq/Translation\\_of\\_Numeric\\_Phrases\\_with\\_Seq2Seq\\_rendered.ipynb](https://github.com/m2dsupsdclclass/lectures-labs/blob/master/labs/07_seq2seq/Translation_of_Numeric_Phrases_with_Seq2Seq_rendered.ipynb)

# GNMT: Residuals

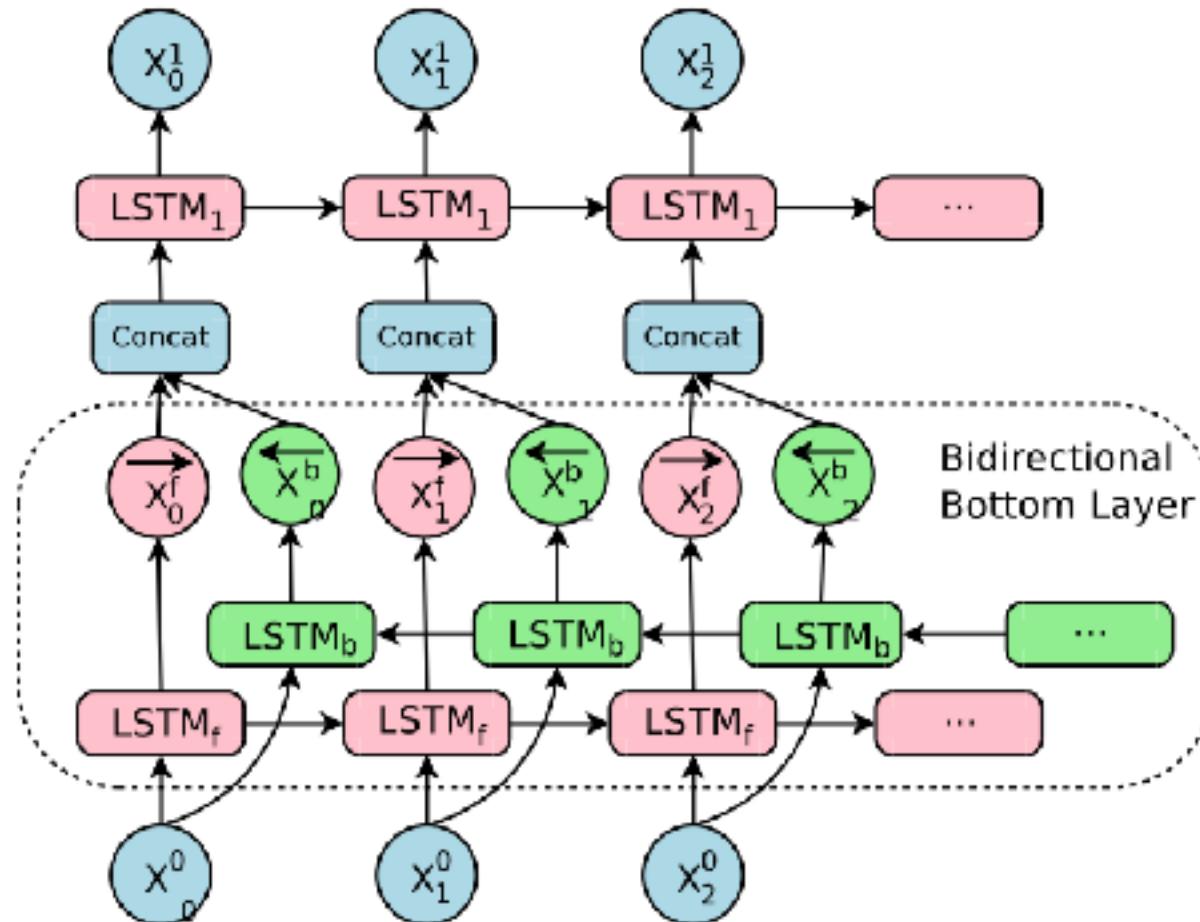
- Google, 2016



Google's Neural Machine Translation:  
<https://arxiv.org/pdf/1609.08144.pdf>

# GNMT: Bidirectionality

- Google, 2016



Google Neural Machine Translation:  
<https://arxiv.org/pdf/1609.08144.pdf>

# GNMT: Attention

- Google, 2016

$$s_t = \text{AttentionFunction}(\mathbf{y}_{i-1}, \mathbf{x}_t) \quad \forall t, \quad 1 \leq t \leq M$$

$$p_t = \exp(s_t) / \sum_{t=1}^M \exp(s_t) \quad \forall t, \quad 1 \leq t \leq M$$

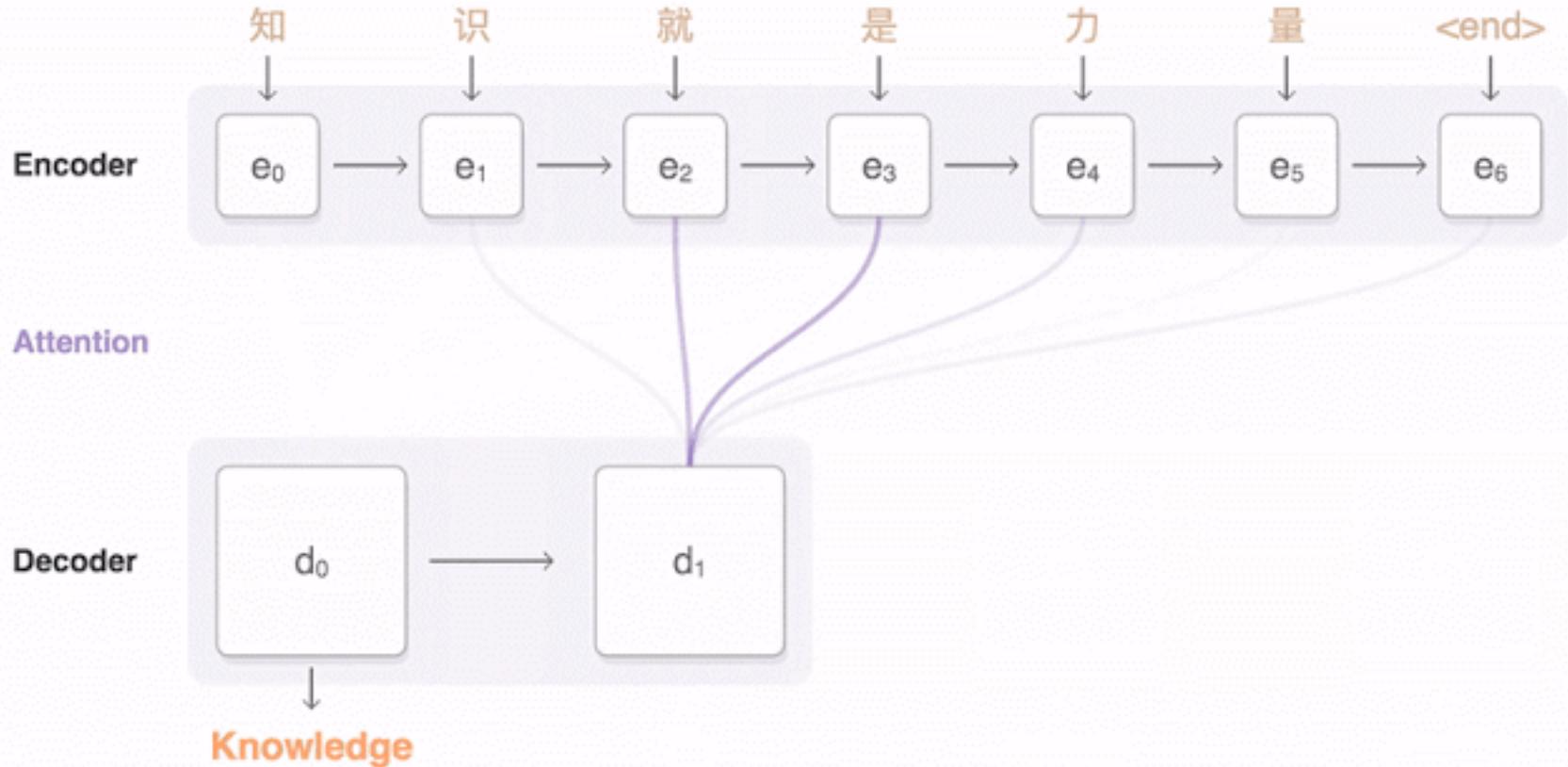
$$\mathbf{a}_i = \sum_{t=1}^M p_t \cdot \mathbf{x}_t$$

where  $\mathbf{x}_t$  is state of the  $t^{th}$  encoder  
 $\mathbf{y}_{i-1}$  is the state of the previous decoder  
and  $\mathbf{a}_i$  is the input for the  $i^{th}$  decoder

# GNMT: Attention

- Google, 2016

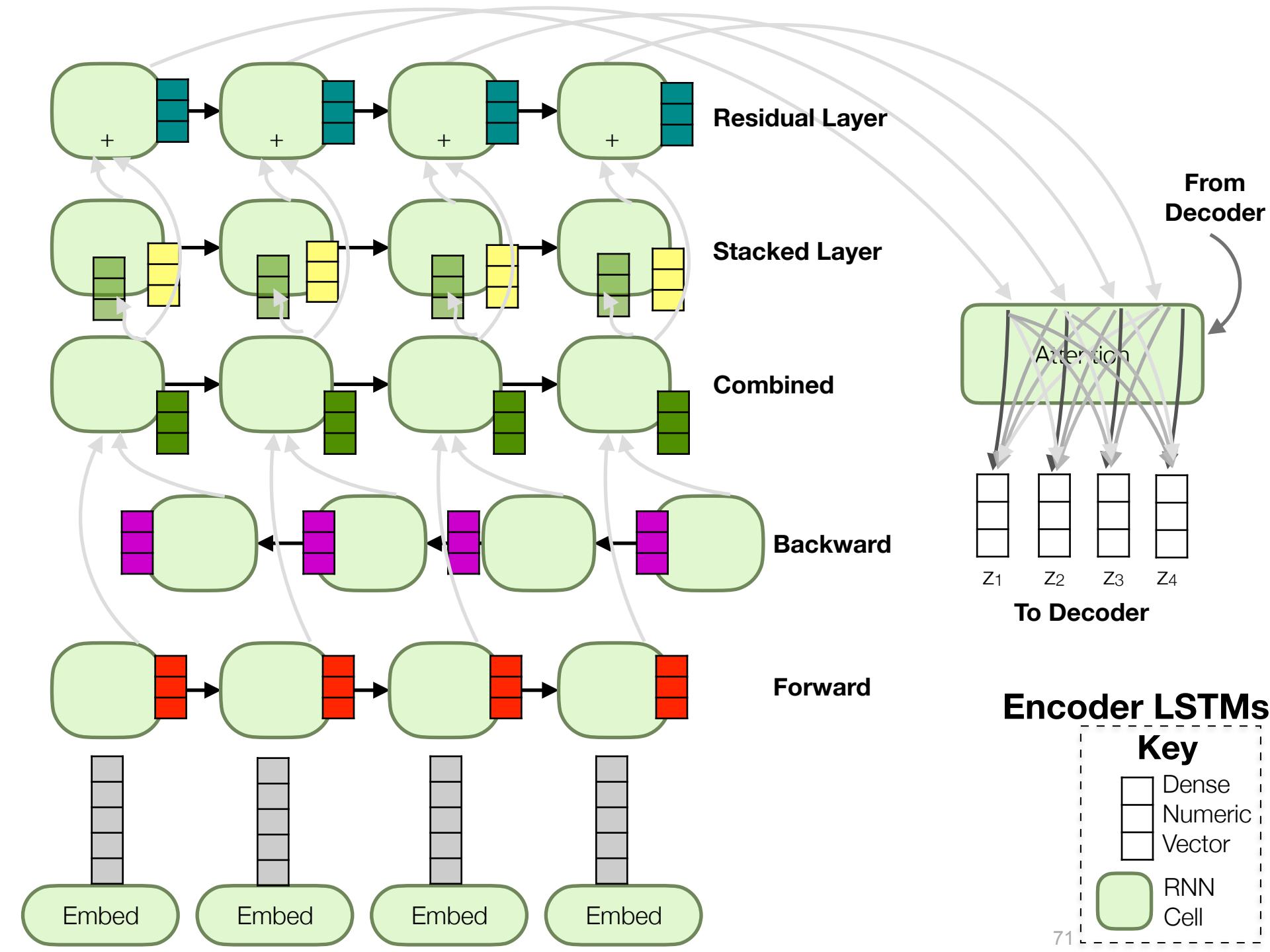
$$s_t = \text{AttentionFunction}(\mathbf{y}_{t-1}, \mathbf{x}_t) \quad \forall t, \quad 1 \leq t \leq M$$
$$p_t = \exp(s_t) / \sum_{t=1}^M \exp(s_t) \quad \forall t, \quad 1 \leq t \leq M$$
$$\mathbf{a}_t = \sum_{t=1}^M p_t \cdot \mathbf{x}_t$$



Google Neural Machine Translation:

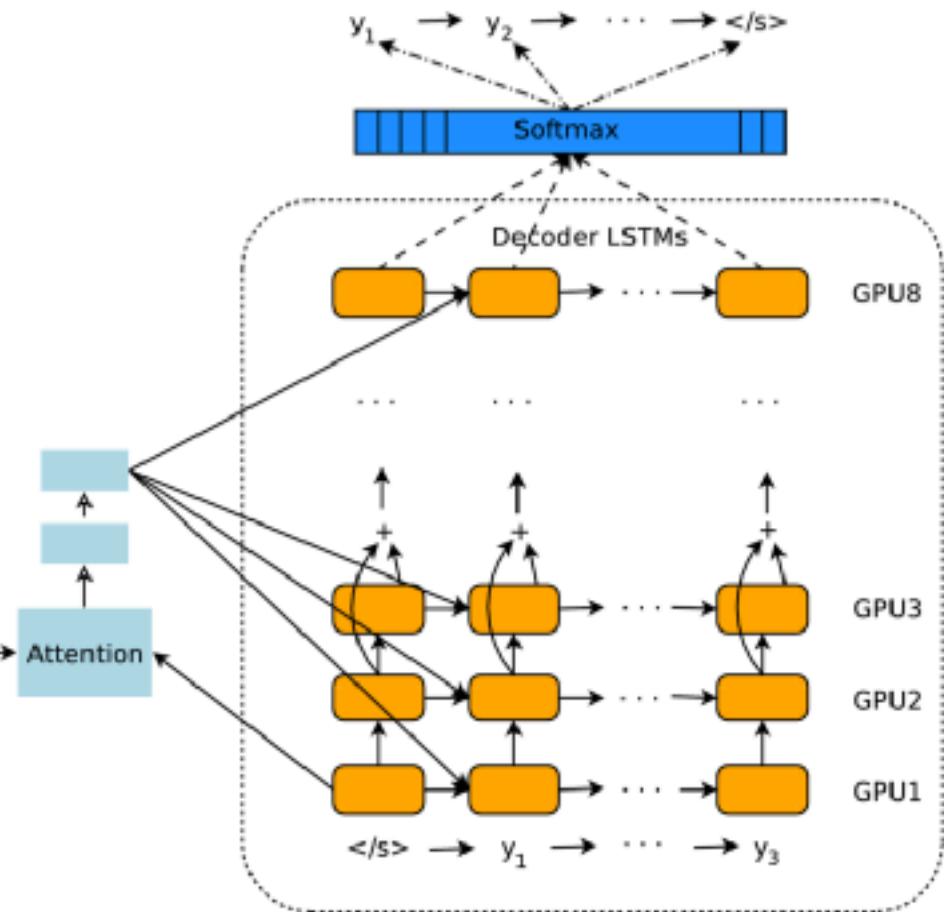
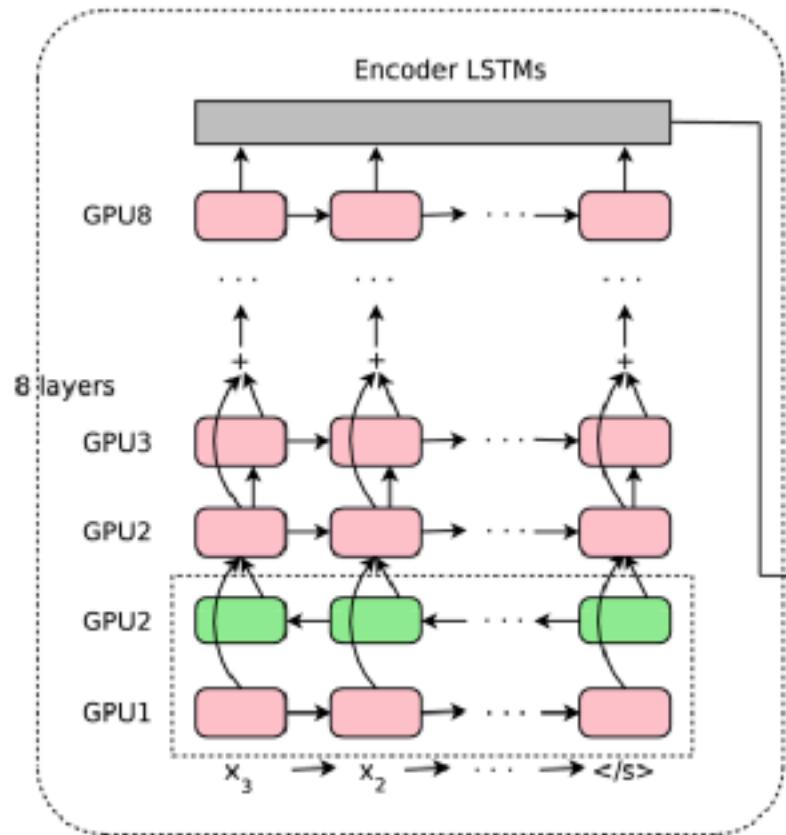
<https://arxiv.org/pdf/1609.08144.pdf>

<https://medium.com/@Synced/history-and-frontier-of-the-neural-machine-translation-dc981d25422d>



# GNMT: Putting it All together

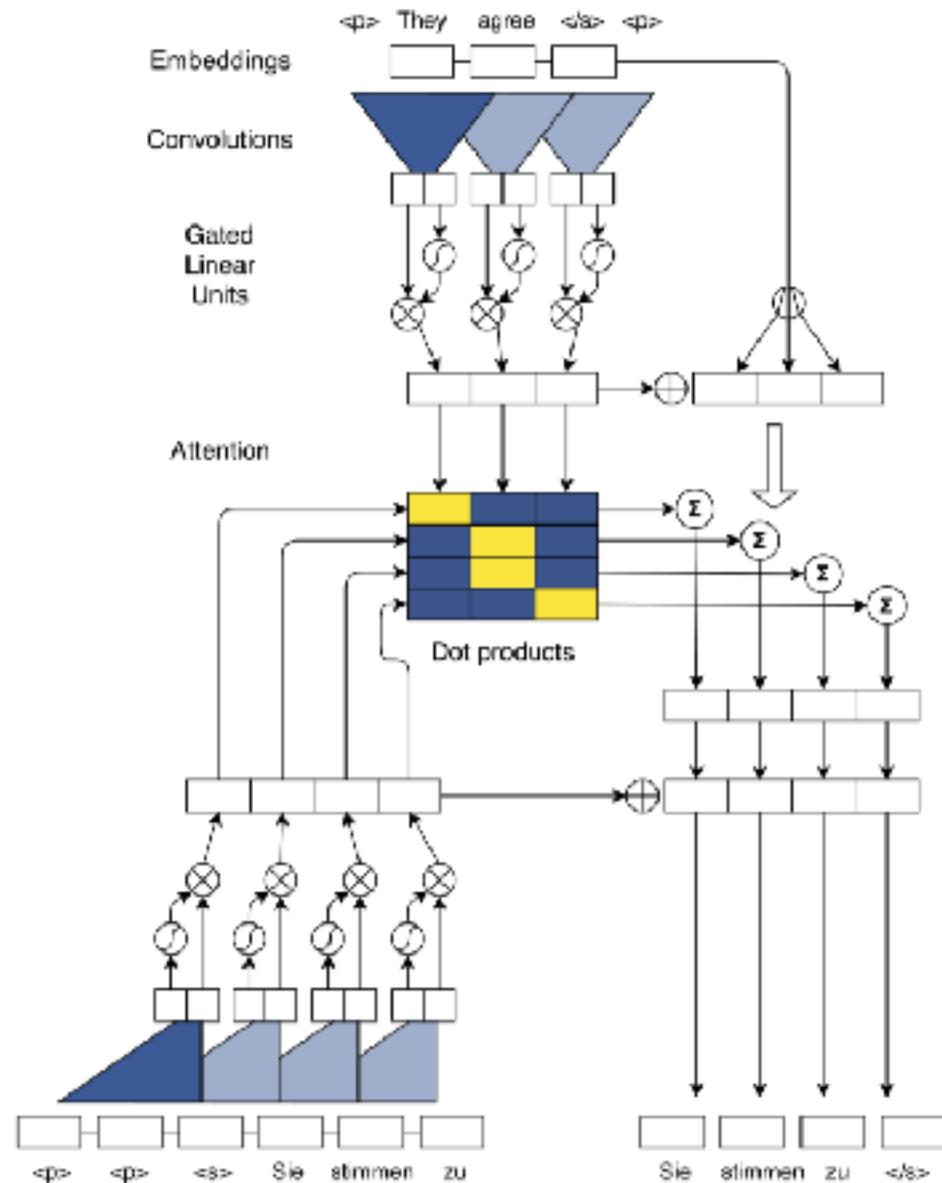
- Google, 2016



Google Neural Machine Translation:  
<https://arxiv.org/pdf/1609.08144.pdf>

# CNNs and RNNs

- Can translation also be done using only CNNs?
  - Yes, Facebook AI already did it,
  - 9 times faster than GNMT
  - Similar Performance
  - July, 2017



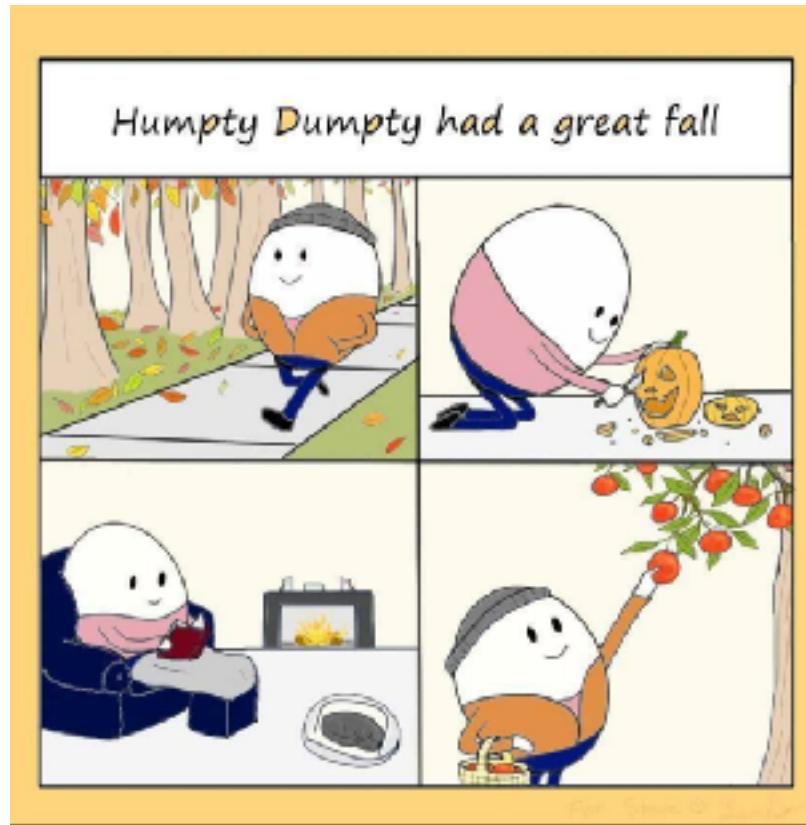
<https://arxiv.org/pdf/1705.03122.pdf>

... from Olivier Grisel



[https://github.com/m2dsupsdlclass/lectures-labs/blob/master/labs/07\\_seq2seq/  
Translation of Numeric Phrases with Seq2Seq rendered.ipynb](https://github.com/m2dsupsdlclass/lectures-labs/blob/master/labs/07_seq2seq/Translation%20of%20Numeric%20Phrases%20with%20Seq2Seq%20rendered.ipynb)

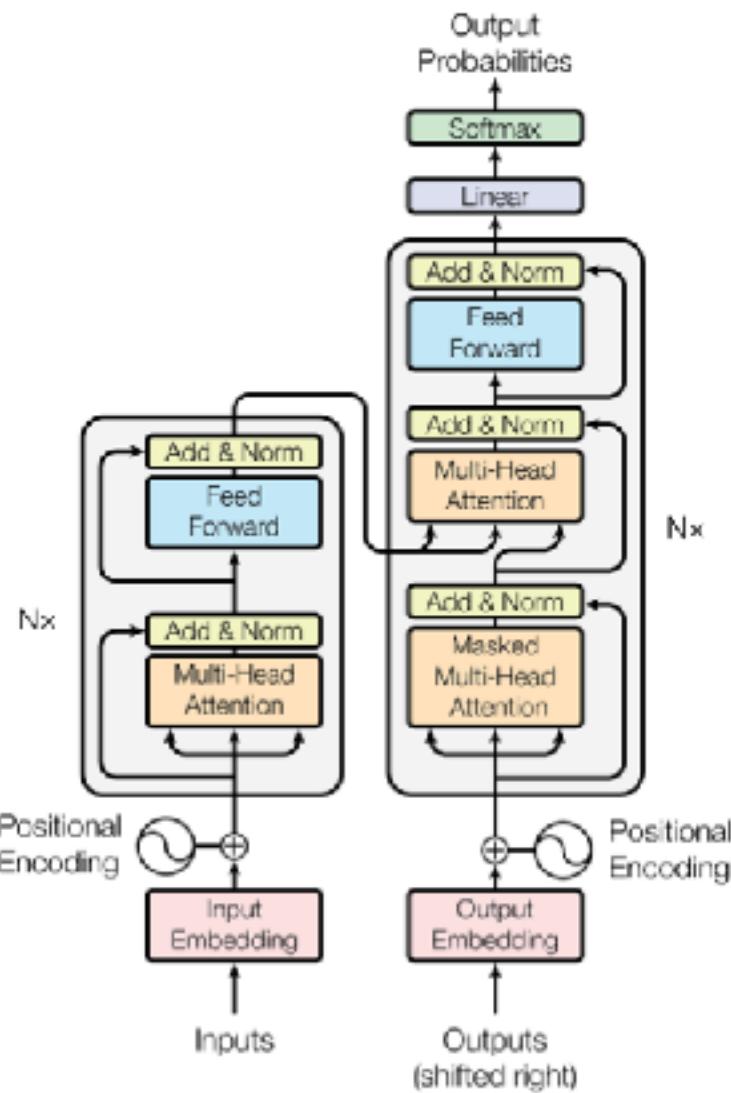
# Transformers



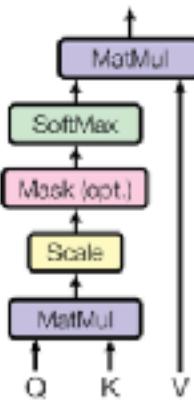
# Attention is All You Need

- Well, it's a good paper title, but not exactly accurate
- Problem: recurrent networks are not inherently parallelized or efficient at remembering
- Convolution needs many examples from all different word positions (after flattening)
- Filters are not resilient to long-term relationships
- Transformer Solution:
  - Build attention into model from the **beginning**
  - Compare all words to each other through **multi-headed** attention
  - Define a notion of "**position**" in the sentence

# Transformer

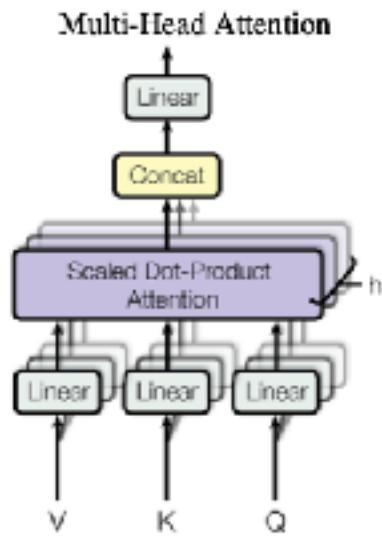


Scaled Dot-Product Attention



for each word

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



more than one  
Q,K,V use in document

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

where  $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$

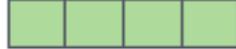
# Transformer: in more detail

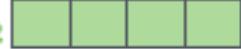
Input

Thinking

Machines

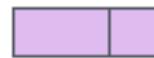
Embedding

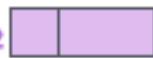
$x_1$  

$x_2$  

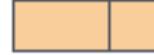
Queries

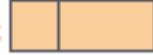
Outputs of Matrix Multiplications:

$q_1$  

$q_2$  

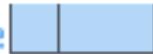
Keys

$k_1$  

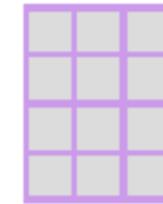
$k_2$  

Values

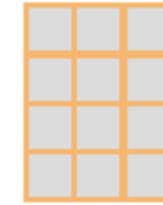
$v_1$  

$v_2$  

Learned Matrices



$W^Q$



$W^K$



$W^V$

# Transformer: in more detail

Input

Embedding

Queries

Keys

Values

Score

Divide by 8 ( $\sqrt{d_k}$ )

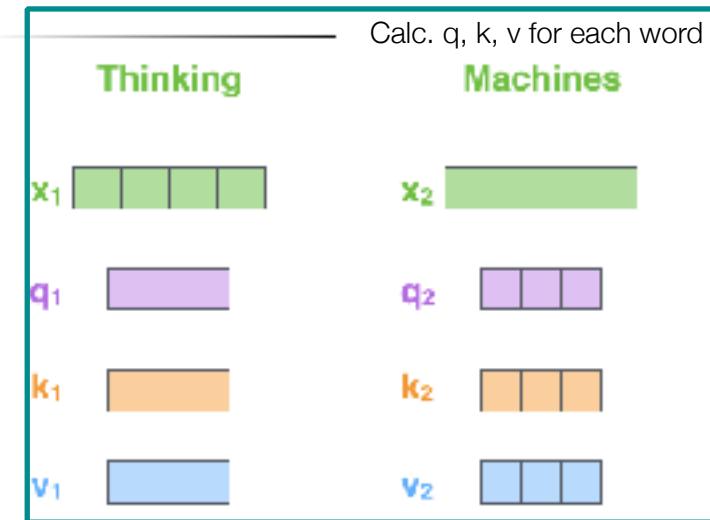
Softmax

Softmax

X

Value

Sum



$$q_1 \cdot k_1 = 112 \quad q_1 \cdot k_2 = 96$$

14

12

0.88

0.12

Calc weights for  $z_1$



weighted sum for all words in document



Straight forward to do this operation in matrix form:

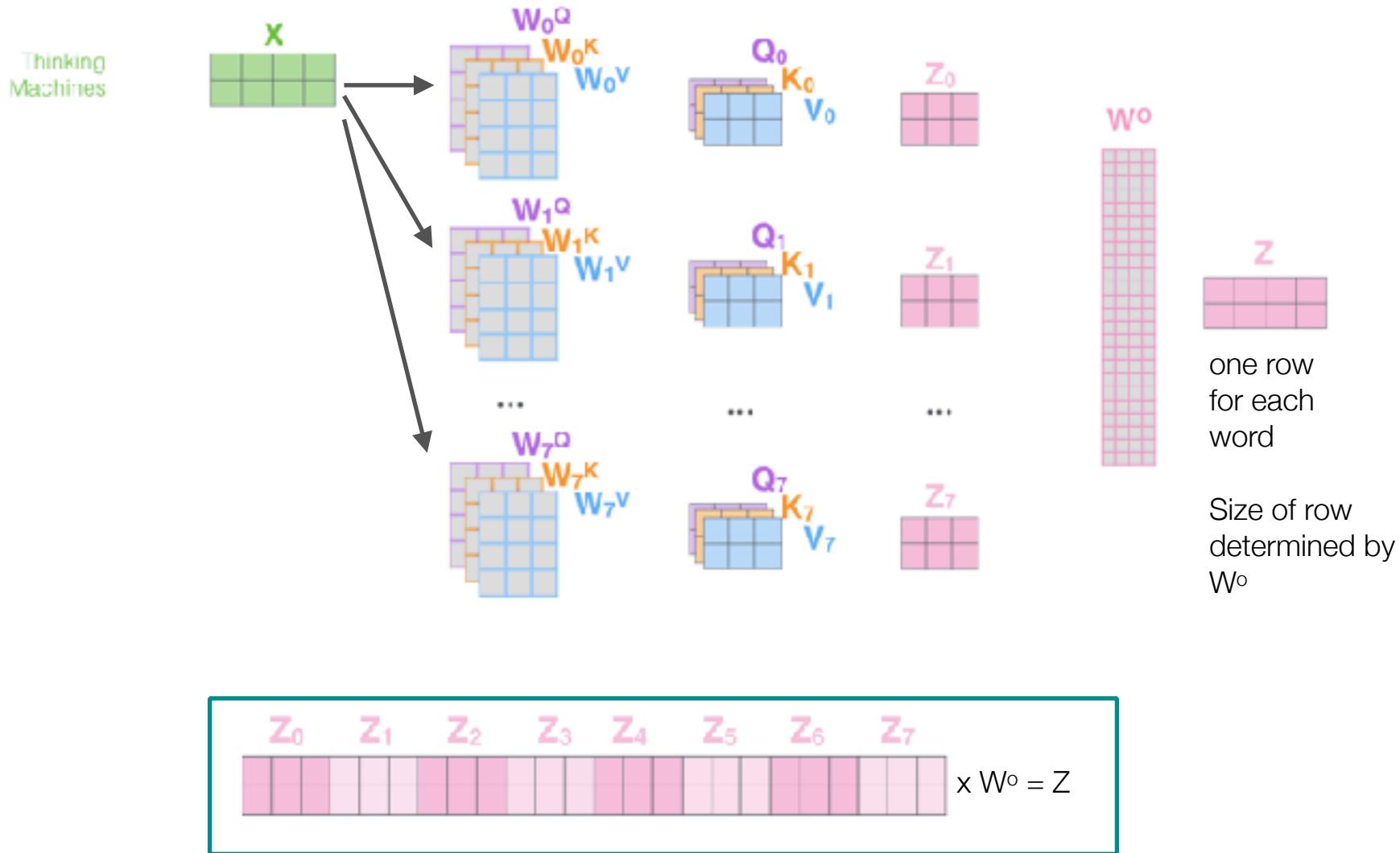
$$\text{Thinking Machines} \times \begin{pmatrix} w_q \\ w_k \\ w_v \end{pmatrix} = \begin{pmatrix} q \\ k \\ v \end{pmatrix}$$

$$\text{Thinking Machines} \times \begin{pmatrix} w_q \\ w_k \\ w_v \end{pmatrix} = \begin{pmatrix} q \\ k \\ v \end{pmatrix}$$

$$\text{Thinking Machines} \times \begin{pmatrix} w_q \\ w_k \\ w_v \end{pmatrix} = \begin{pmatrix} q \\ k \\ v \end{pmatrix}$$

$$\text{softmax}\left(\frac{\begin{pmatrix} q \\ k \\ v \end{pmatrix} \times \begin{pmatrix} K^T \\ V^T \end{pmatrix}}{\sqrt{d_k}}\right) = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$$

# Transformer: Multi-headed Attention



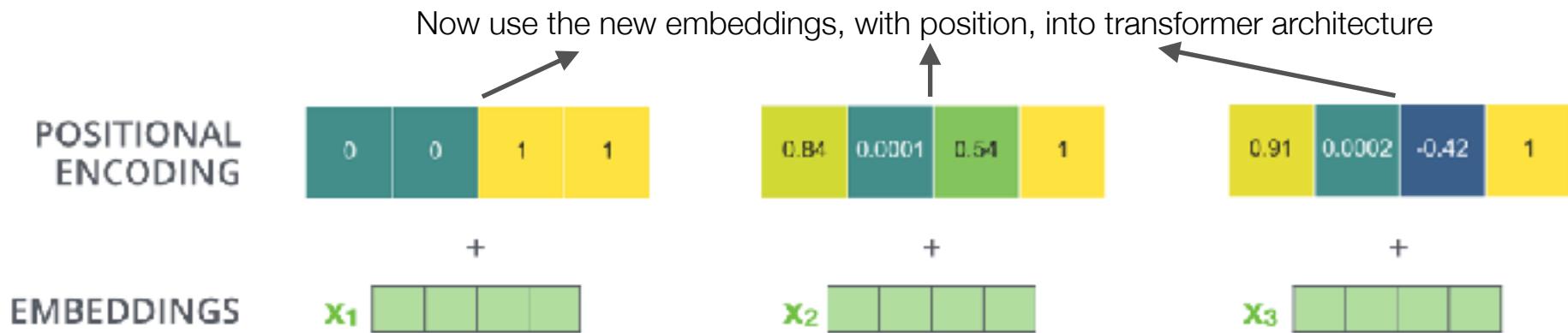
Excellent Blog on Transformers: <http://jalammar.github.io/illustrated-transformer/>

# Transformer: Positional Encoding

- Objective: add notion of position to embedding
- Attempt in paper: add sin/cos to embedding
- But could be anything that encodes position

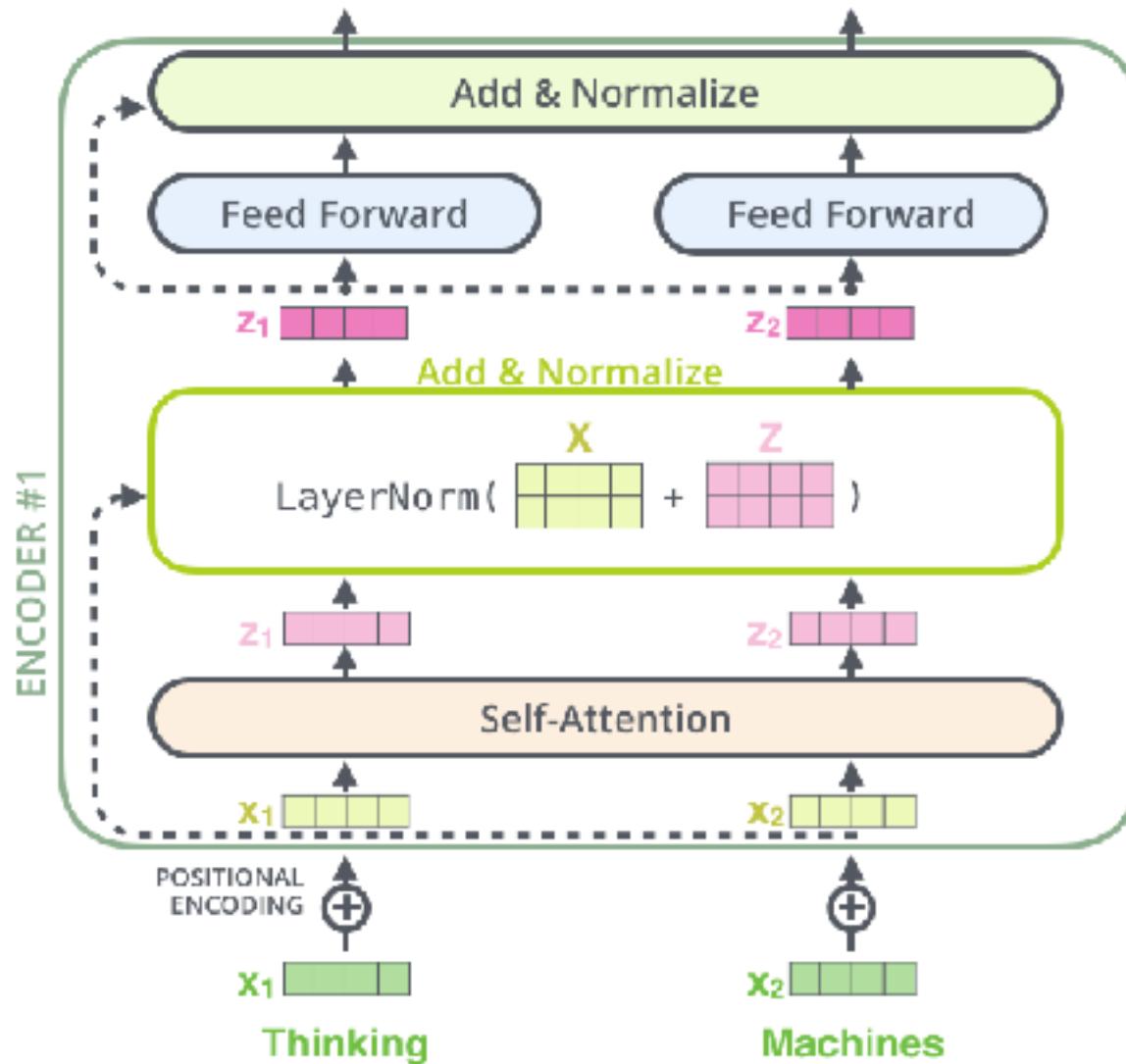
$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$



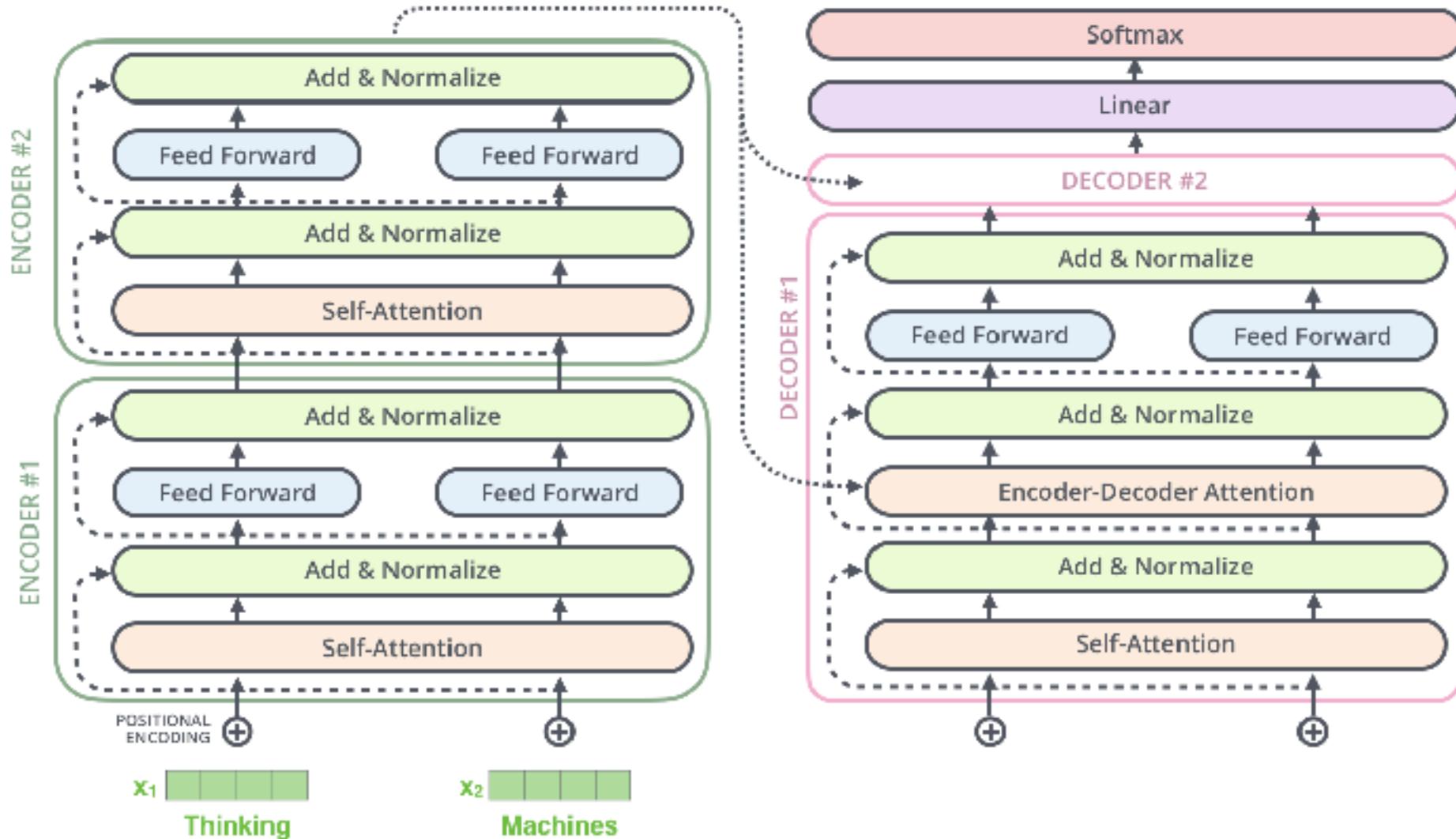
**Hypothesis:** Now the word proximity is encoded in the embedding matrix, with other pertinent information. Well, it does help... so it could be true that this is a good way to do it.

# Transformer: Residual Connections



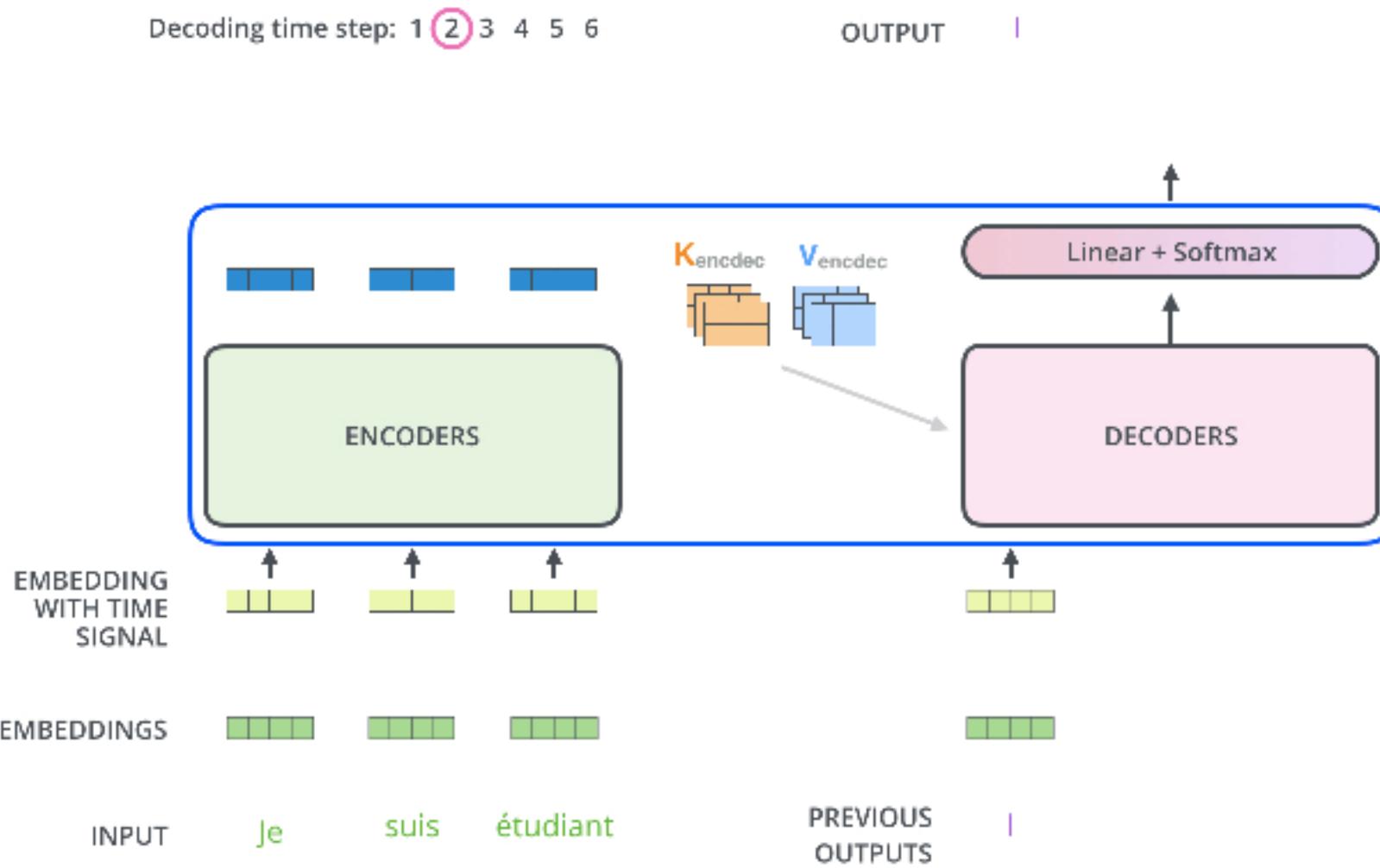
Excellent Blog on Transformers: <http://jalammar.github.io/illustrated-transformer/>

# Transformer: Putting it all together



Excellent Blog on Transformers: <http://jalammar.github.io/illustrated-transformer/>

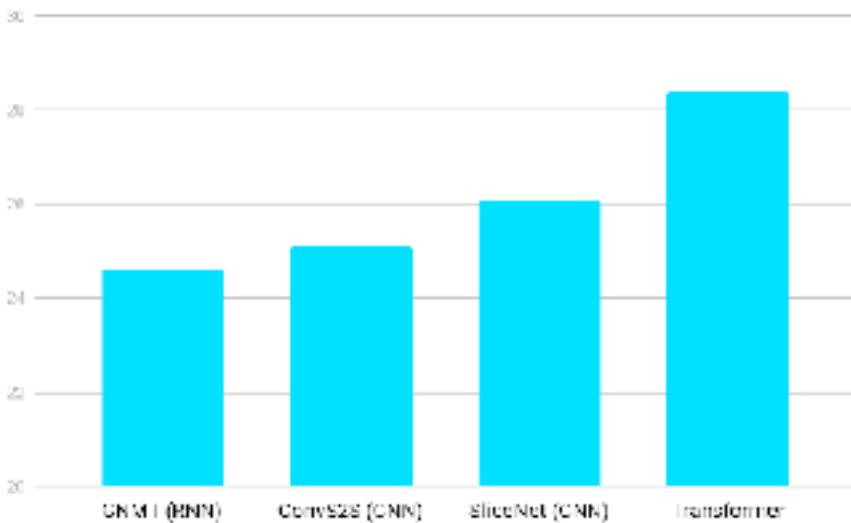
# Transformer: Putting it all together



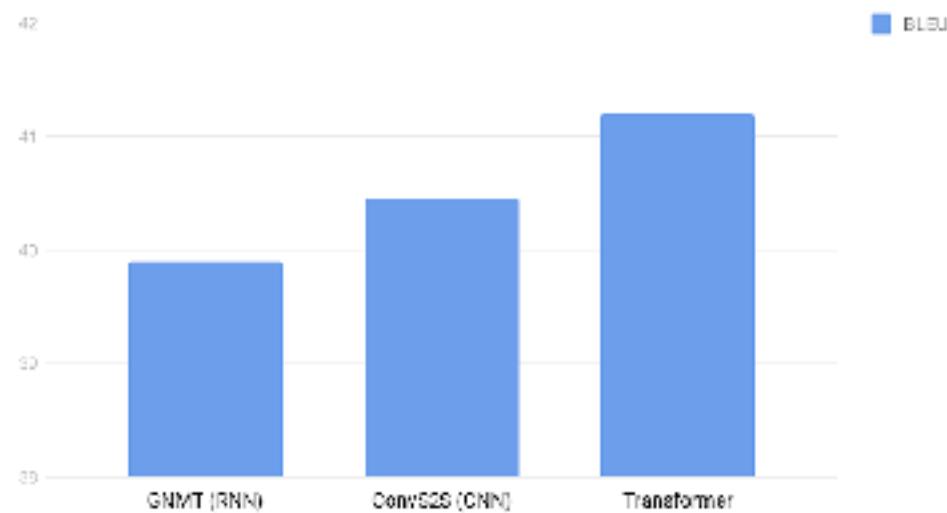
Excellent Blog on Transformers: <http://jalammar.github.io/illustrated-transformer/>

# Results

English German Translation quality



English French Translation Quality



<https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html>

## Implementations:

- Not Native to Keras or Tensorflow, but many Open Source Implementations Exist
- Is Native to PyTorch

# Next time

- ❑ Class Retrospective

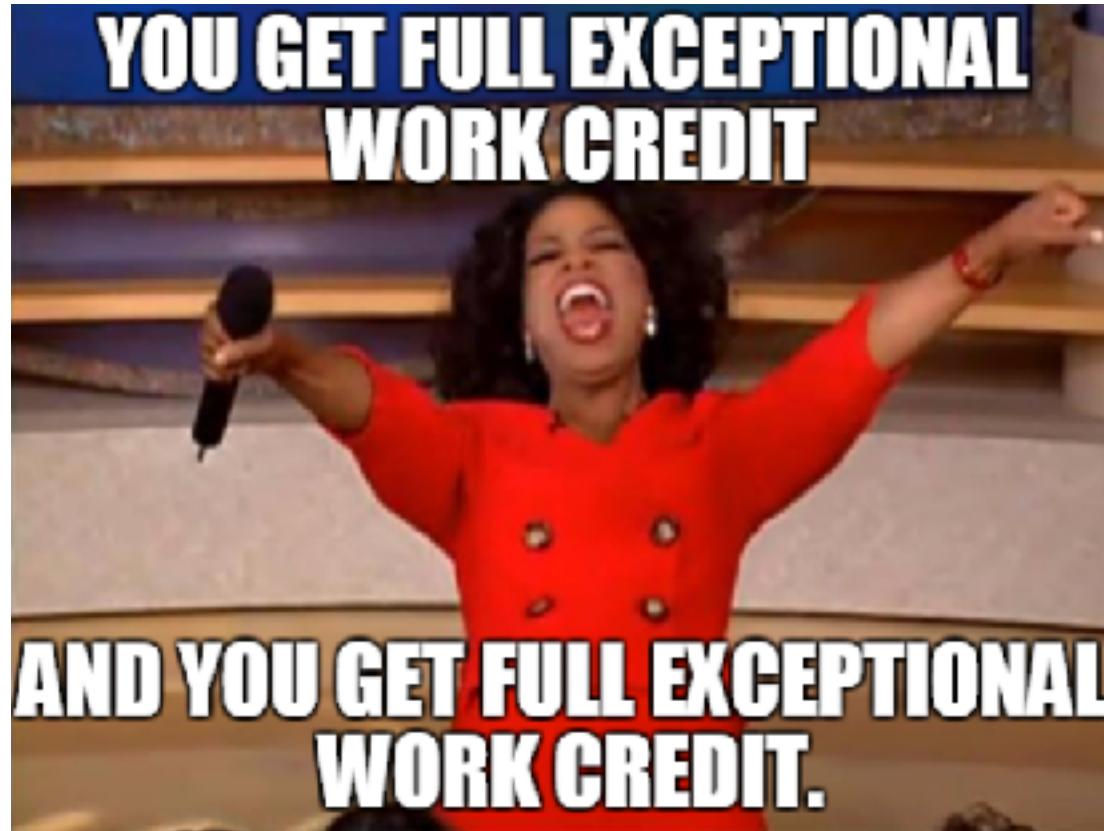
# Lecture Notes for **Machine Learning in Python**

Professor Eric Larson  
**Final Lecture: Retrospective**

# Lecture Agenda

- ❑ Logistics
  - ❑ RNNs due **During Finals Time**
- ❑ Agenda
  - ❑ Retrospective

# Town Hall



Courtesy of Daniel F.P. Garcia and Oprah

# Course Retrospective

## Leading ML researchers issue statement of support for JMLR

- AI winters exist
- machine learning
- repeat)**

- Formal methods

- At the end of

- Open source** advancement

[http://ww](http://www.cs.berkeley.edu/~jordan/mlj_resignation.html)

From: Michael Jordan [mailto:jordan@CS.Berkeley.EDU]  
 Sent: Monday, October 08, 2001 5:33 PM  
 Subject: Letter of resignation from Machine Learning journal

Dear colleagues in machine learning,

The forty people whose names appear below have resigned from the Editorial Board of the Machine Learning Journal (MLJ). We would like to make our resignations public, to explain the rationale for our action, and to indicate some of the implications that we see for members of the machine learning community worldwide.

The machine learning community has come of age during a period of enormous change in the way that research publications are circulated. Fifteen years ago research papers did not circulate easily, and as with other research communities we were fortunate that a viable commercial publishing model was in place so that the fledgling MLJ could begin to circulate. The needs of the community, principally those of seeing our published papers circulate as widely and rapidly as possible, and the business model of commercial publishers were in harmony.

Times have changed. Articles now circulate easily via the Internet, but unfortunately MLJ publications are under restricted access. Universities and research centers can pay a yearly fee of \$1050 US to obtain unrestricted access to MLJ articles (and individuals can pay \$120 US). While these fees provide access for institutions and individuals who can afford them, we feel that they also have the effect of limiting contact between the current machine learning community and the potentially much larger community of researchers worldwide whose participation in our field should be the fruit of the modern Internet.

None of the revenue stream from the journal makes its way back to authors, and in this context authors should expect a particularly favorable return on their intellectual contribution---they should expect a service that maximizes the distribution of their work. We see little benefit accruing to our community from a mechanism that ensures revenue for a third party by restricting the communication channel between authors and readers.

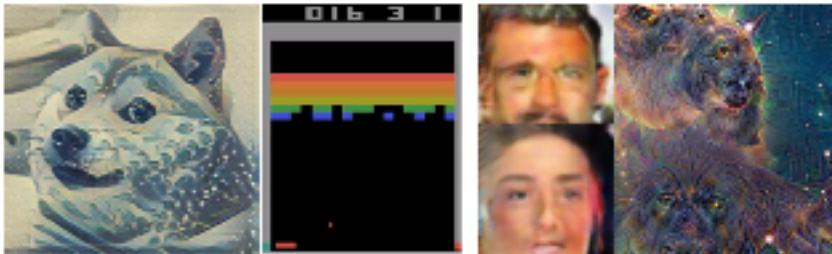
Chris Atkeson  
 Peter Bartlett  
 Andrew Barto  
 Jonathan Baxter  
 Yoshua Bengio  
 Kristin Bennett  
 Chris Bishop  
 Justin Boyan  
 Carla Brodley  
 Claire Cardie  
 William Cohen  
 Peter Dayan  
 Tom Dietterich  
 Jerome Friedman  
 Nir Friedman  
 Zoubin Ghahramani  
 David Heckerman  
 Geoffrey Hinton  
 Yoav Hirsh  
 Tommi Jaakkola  
 Michael Jordan  
 Leslie Kaelbling  
 Daphne Koller  
 John Lafferty  
 Sridhar Mahadevan  
 Marina Meila  
 Andrew McCallum  
 Tom Mitchell  
 Stuart Russell  
 Lawrence Saul  
 Bernhard Schoelkopf  
 John Shawe-Taylor  
 Yoram Singer  
 Satinder Singh  
 Padhraic Smyth  
 Richard Sutton  
 Sebastian Thrun  
 Manfred Warmuth  
 Chris Williams  
 Robert Williamson

# Topics review

- Data **munging** in pandas and numpy
- Data **visualization** in jupyter with matplotlib, pandas, seaborn, and plotly
- Data preprocessing: **dim reduction**, images, text, categorical features, **embeddings**
- **Linear models**: linear regression, logistic regression, simple neural networks
- **Optimization** strategies: Gradient ascent, Quasi-Newton
- **Back propagation** in MLP (from scratch)
- Tensorflow/Keras for **wide and deep networks**
- **Convolutional** neural networks
- **Recurrent** neural networks

# Topics Not Covered

- Visualizing Deep Convolutional Networks
- Fully Convolutional Networks
- Transfer/Multi-Task Learning
- Style Transfer
- Generative Adversarial Networks
- Reinforcement Learning



## Syllabus for CSE8321: Machine Learning and Neural Networks

### Course Schedule

Week	Lecture A	Lecture B	Last Date
1	Lecture: Course Introduction and Syllabus	Lecture: Basics of Neural Networks	
2	Student IP presentation (Reviewing: Early days of Deep Learning; Chapter 2017) Student IP discussion: Leader vs. TBD Lecture: CNN Visualization Overview	Lecture: CNN Architecture Methods Reading: Chapter 5, Section 4	
3	Lecture: CNNs: Class Invariance	Lecture: Image Style Transfer Overview Reading: Chapter 6, Sections 2 and 3	Homework: CNNs
4	Student IP presentation 1: Reviewing: A New GAN Algorithm For In-Roll Video Style Transfer and Content Mixing Lecture: GANs and Content Mixing	Student IP presentation 1: Reviewing: An Overview of GANs For Learning To Generate Pictures (Arxiv 2016) Lecture: "Hyperparameter Tuning: Survey Methods and Optimal Convergence"	
5	Lecture/Demo: Image Style Transfer in PyTorch	Lecture: Transfer Learning in CNNs Reading: Chapter 5, Section 2 and 3	Last Test: Style Transfer
6	Lecture/Demo: Semantic Image-to-Monochrome	Lecture: Multi-modal learning (Cross-Net)	
7	Student IP presentation 2: Reviewing: Deep Multimodal Learning: An Overview of Recent Advances and Future Challenges Lecture: GANs: Towards More Stable Training	Student IP presentation 2: Reviewing: An Overview of GANs For Learning To Generate Pictures (Arxiv 2016) Lecture: "Hyperparameter Tuning: Survey Methods and Optimal Convergence"	
8	Lecture: Generative Adversarial Networks Overview Reading: Chapter 6, Sections 1, 2 and 3	Student IP presentation 3: Reviewing: Generative Hypermodels—Less-is-more with Deep Convolutional GANs, by Radford J 2018 Lecture: "Hyperparameter Tuning: Survey Methods and Optimal Convergence"	Last Test: Multi-modal and Multi-task
9	Student IP presentation 3: Reviewing: Generative Hypermodels—Less-is-more with Deep Convolutional GANs, by Radford J 2018 Lecture: GANs: Hyperparameter Tuning	Lecture Demo: GAN training on Monochrome	
10	Lecture: Deep Reinforcement Learning Overview	Lecture: G-Dueling	Last Test: GANs
11	Lecture: Trust Region Policy Optimization	Lecture: Trust Region Policy Optimization: Part 1	
12	Student IP presentation 4: Reviewing: Playing Atari with Deep Reinforcement Learning (PDRL) Lecture/Demo: OpenAI Gym: Part 1	Lecture Demo: Training Deep Reinforcement Learning in Finance	

## Syllabus for CSE8321: Machine Learning and Neural Networks

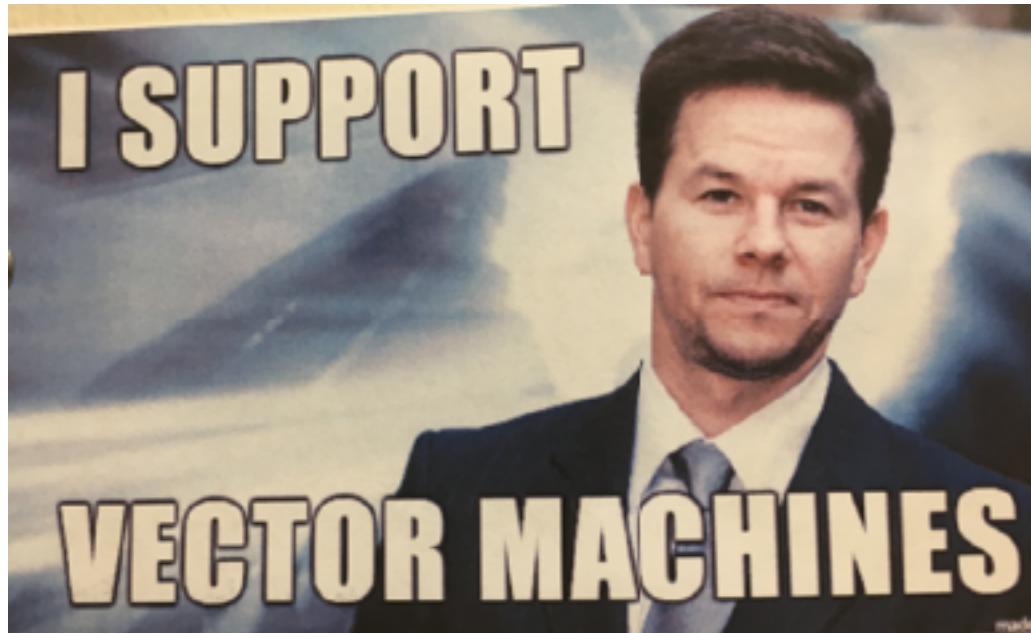
### Overview

This course extends basic knowledge of the use of neural networks in machine learning beyond simple prediction, especially targeted outputs that are generation or alteration of images, text, and audio. This course emphasizes topics of neural networks in the “deep learning” subdomain. This course will survey of important topics and current areas of research, including transfer learning, multi-task and multi-modal learning, image style transfer, neural network visualization, deep convolutional generative adversarial networks, and deep reinforcement learning. For grading, students are expected to complete smaller team-based projects throughout the semester, present one research paper in a 15-20 minute group presentation (covering topics in the course), and complete a comprehensive final project that involves a number of different deep learning architectures.

# Thank you for a great semester!

- but it could **have been better** somehow, right?
  - how could you learn better?
  - what should **not be cut** or **changed**?
  - SVMs were cut from class to allow better coverage of Deep Learning
  - More RNNs? Less RNNs?
  - More convolutional approaches/depth?
  - More APIs? Turi / Pytorch?

# Thank You for an Excellent Semester!



Courtesy of Omar Roa

# Backup slides

# TensorFlow

<http://r2rt.com/recurrent-neural-networks-in-tensorflow-i.html>

```
with tf.variable_scope('rnn_cell'):
    W = tf.get_variable('W', [num_classes + state_size, state_size])
    b = tf.get_variable('b', [state_size], initializer=tf.constant_initializer(0.0))

def rnn_cell(rnn_input, state):
    with tf.variable_scope('rnn_cell', reuse=True):
        W = tf.get_variable('W', [num_classes + state_size, state_size])
        b = tf.get_variable('b', [state_size], initializer=tf.constant_initializer(0.0))
    return tf.tanh(tf.matmul(tf.concat([rnn_input, state]), W) + b)

state = init_state
rnn_outputs = []
for rnn_input in rnn_inputs:
    state = rnn_cell(rnn_input, state)
    rnn_outputs.append(state)
final_state = rnn_outputs[-1]

#logits and predictions
with tf.variable_scope('softmax'):
    W = tf.get_variable('W', [state_size, num_classes])
    b = tf.get_variable('b', [num_classes], initializer=tf.constant_initializer(0.0))
logits = [tf.matmul(rnn_output, W) + b for rnn_output in rnn_outputs]
predictions = [tf.nn.softmax(logit) for logit in logits]

# Turn our y placeholder into a list labels
y_as_list = [tf.squeeze(i, squeeze_dims=[1]) for i in tf.split(1, num_steps, y)]

#losses and train_step
losses = [tf.nn.sparse_softmax_cross_entropy_with_logits(logit, label) for \
          logit, label in zip(logits, y_as_list)]
total_loss = tf.reduce_mean(losses)
train_step = tf.train.AdagradOptimizer(learning_rate).minimize(total_loss)
```

# recurrent networks

<http://r2rt.com/recurrent-neural-networks-in-tensorflow-i.html>

```
def train_network(num_epochs, num_steps, state_size=4, verbose=True):
    with tf.Session() as sess:
        sess.run(tf.initialize_all_variables())
        training_losses = []
        for idx, epoch in enumerate(gen_epochs(num_epochs, num_steps)):
            training_loss = 0
            training_state = np.zeros((batch_size, state_size))
            if verbose:
                print("\nEPOCH", idx)
            for step, (X, Y) in enumerate(epoch):
                tr_losses, training_loss_, training_state, _ = \
                    sess.run([losses,
                             total_loss,
                             final_state,
                             train_step],
                             feed_dict={x:X, y:Y, init_state:training_state})
                training_loss += training_loss_
                if step % 100 == 0 and step > 0:
                    if verbose:
                        print("Average loss at step", step,
                              "for last 250 steps:", training_loss/100)
                training_losses.append(training_loss/100)
                training_loss = 0
    return training_losses
```

# TensorFlow

<http://r2rt.com/recurrent-neural-networks-in-tensorflow-i.html>

```
def train_network(num_epochs, num_steps, state_size=4, verbose=True):
    with tf.Session() as sess:
        sess.run(tf.initialize_all_variables())
        for idx, epoch in enumerate(gen_epochs(num_epochs, num_steps)):
            training_state = np.zeros((batch_size, state_size))
            for X, Y in epoch:
                tr_losses, training_loss_, training_state, _ = \
                    sess.run([losses,
                             total_loss,
                             final_state,
                             train_step],
                             feed_dict={x:X, y:Y, init_state:training_state})
```

# TensorFlow (simplified)

<http://r2rt.com/recurrent-neural-networks-in-tensorflow-1.html>

```
cell = tf.nn.rnn_cell.BasicRNNCell(state_size)
rnn_outputs, final_state = tf.nn.rnn(cell, rnn_inputs, initial_state=init_state)

loss_weights = [tf.ones([batch_size]) for i in range(num_steps)]
losses = tf.nn.seq2seq.sequence_loss_by_example(logits, y_as_list, loss_weights)

x = tf.placeholder(tf.int32, [batch_size, num_steps], name='input_placeholder')
y = tf.placeholder(tf.int32, [batch_size, num_steps], name='labels_placeholder')
init_state = tf.zeros([batch_size, state_size])

x_one_hot = tf.one_hot(x, num_classes)
rnn_inputs = tf.unpack(x_one_hot, axis=1)

cell = tf.nn.rnn_cell.BasicRNNCell(state_size)
rnn_outputs, final_state = tf.nn.rnn(cell, rnn_inputs, initial_state=init_state)

with tf.variable_scope('softmax'):
    W = tf.get_variable('W', [state_size, num_classes])
    b = tf.get_variable('b', [num_classes], initializer=tf.constant_initializer(0.0))
logits = [tf.matmul(rnn_output, W) + b for rnn_output in rnn_outputs]
predictions = [tf.nn.softmax(logit) for logit in logits]

y_as_list = [tf.squeeze(i, squeeze_dims=[1]) for i in tf.split(1, num_steps, y)]

loss_weights = [tf.ones([batch_size]) for i in range(num_steps)]
losses = tf.nn.seq2seq.sequence_loss_by_example(logits, y_as_list, loss_weights)
total_loss = tf.reduce_mean(losses)
train_step = tf.train.AdagradOptimizer(learning_rate).minimize(total_loss)
```