Lecture Notes for

Machine Learning in Python

Professor Eric Larson

History and Introduction to Recurrent Neural Networks

Lecture Agenda

- Logistics
  - RNNs due date Reminder
- Recurrent Networks (~multi-lecture agenda)
  - Overview and History
  - Embeddings
  - Types of RNNs
  - Demo A
  - CNNs and RNNs
  - Demo B
  - Ethical Concerns for RNNs
  - Course Retrospective

Class Overview, by topic

Table Data Visualization

Numpy, Pandas, Seaborn
Overviews with some in-depth discussion

Dimension Reduction and Image Processing

Scikit-learn, Scikit Image,
Intuition only, Some mathematics

Linear and Logistic Regression

Numpy, Recreate API for Scikit-learn
Detailed mathematics for simple optimization
intuition for advanced optimization

Neural Networks and Back Prop.

Numpy
Detailed mathematics for NN operations

Wide and Deep Networks

Convolutional Networks

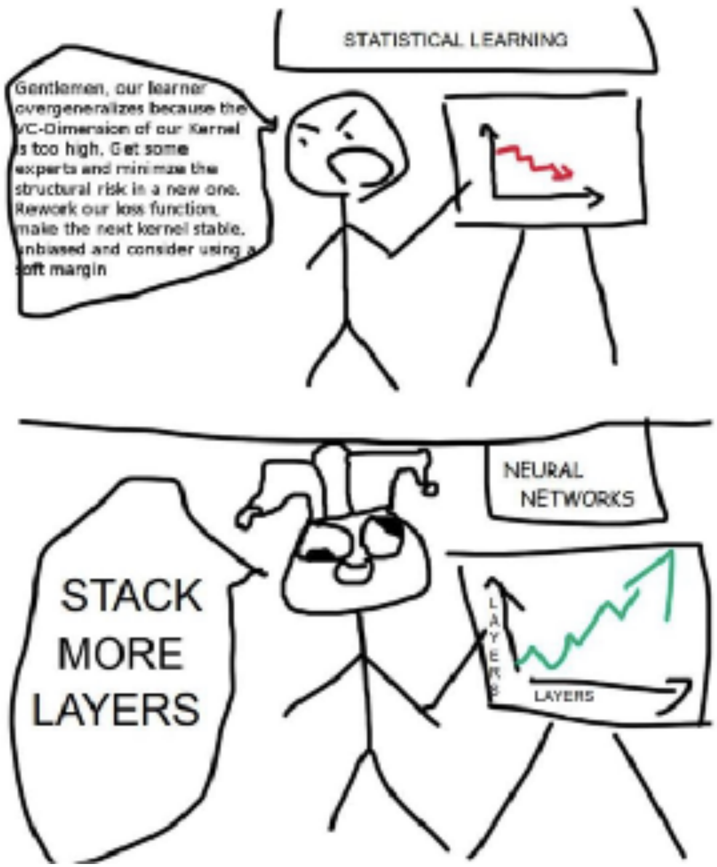
Recurrent Networks

Keras, Tensorflow
Intuition, Detailed implement.

Ethics in Language Models

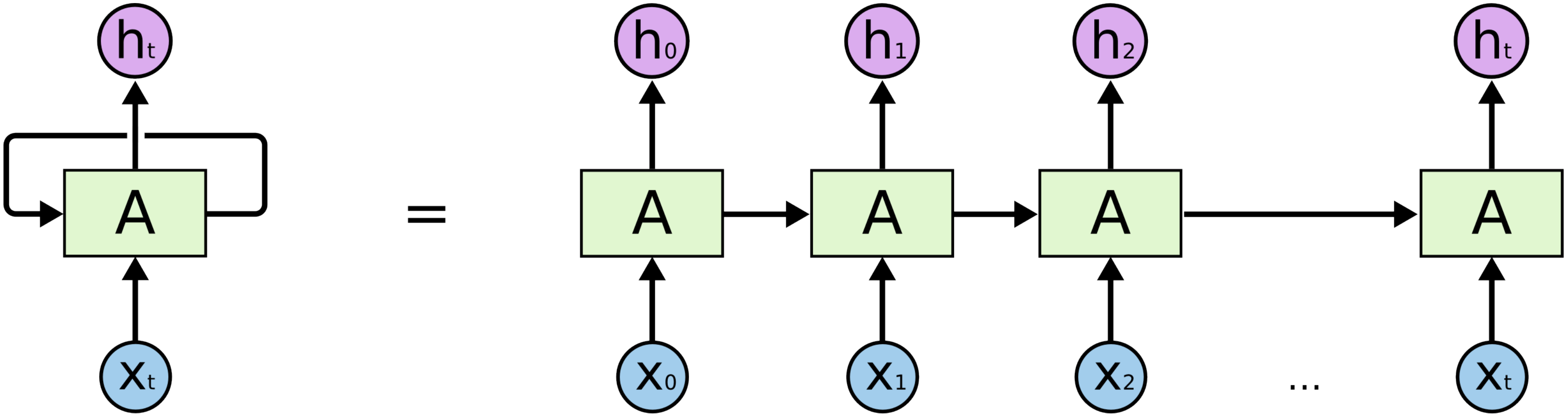
ConceptNet
Case studies

Overview of RNNs



Recurrent Networks: Main Idea

- equations for recurrent networks



compact

$$h_t = f_A(X_t, h_{t-1})$$

unrolled

$$W_A = [U, W]$$

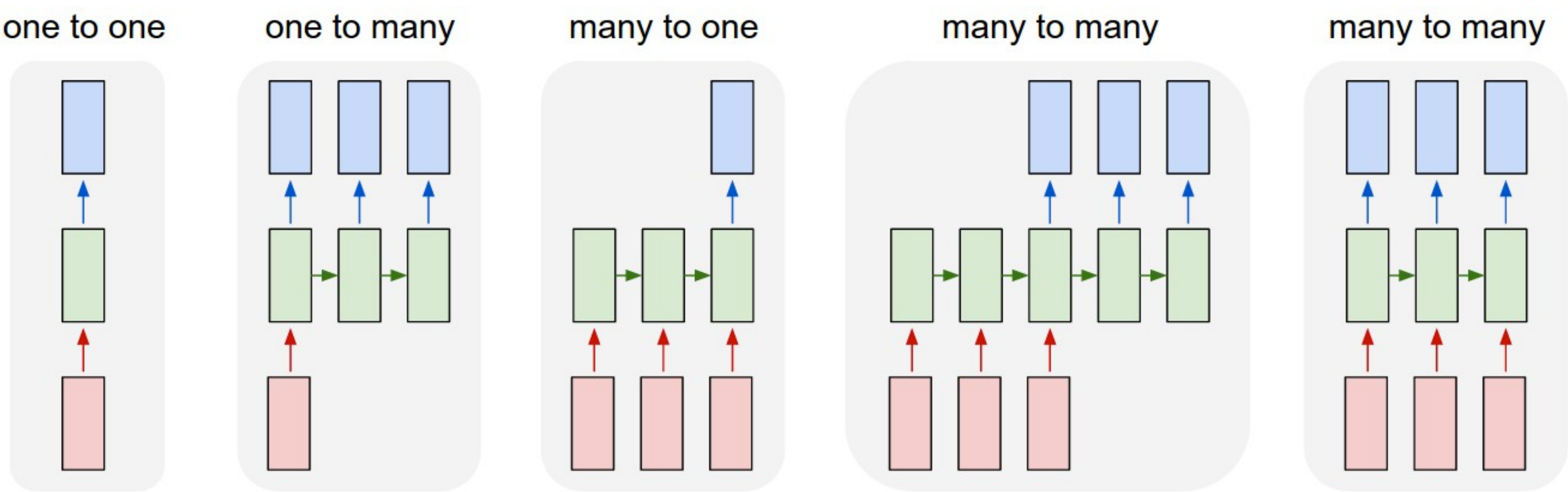
For example:

concatenate

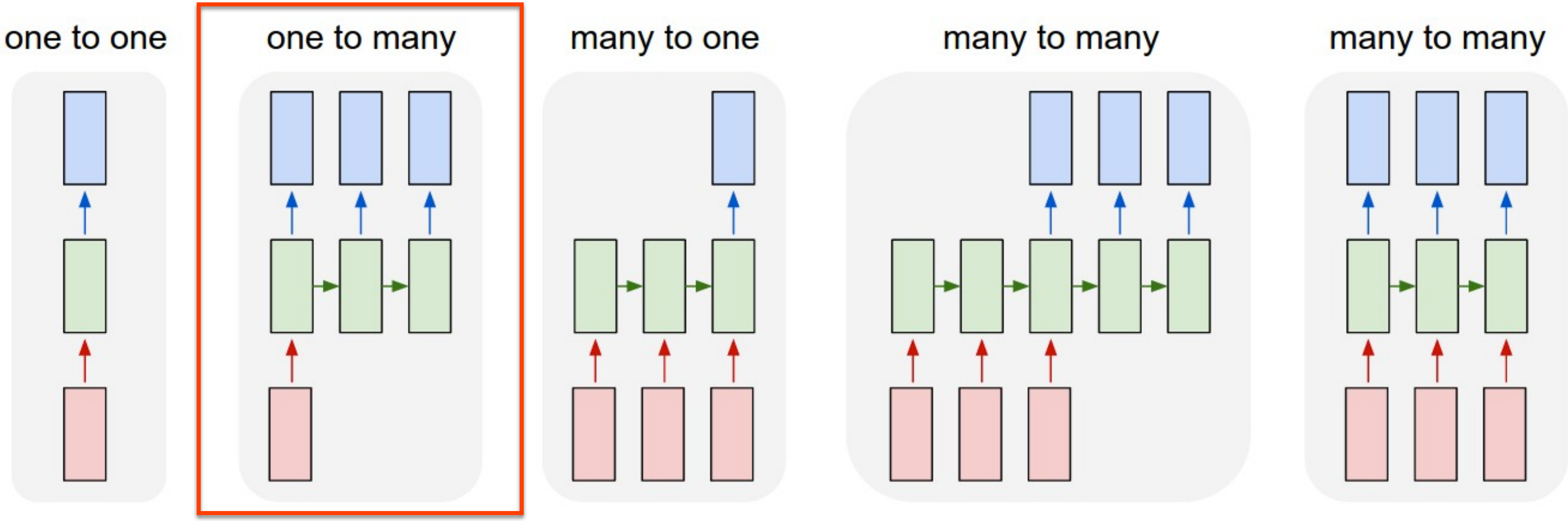
$$h_t = U \cdot X_t + W \cdot h_{t-1} + b \quad = W_A \cdot (X_t \otimes h_{t-1}) + b$$

$$h_t = U \cdot X_t + W \cdot \underbrace{\left(U \cdot X_{t-1} + W \cdot h_{t-2} + b\right)}_{\text{from previous}} + b$$

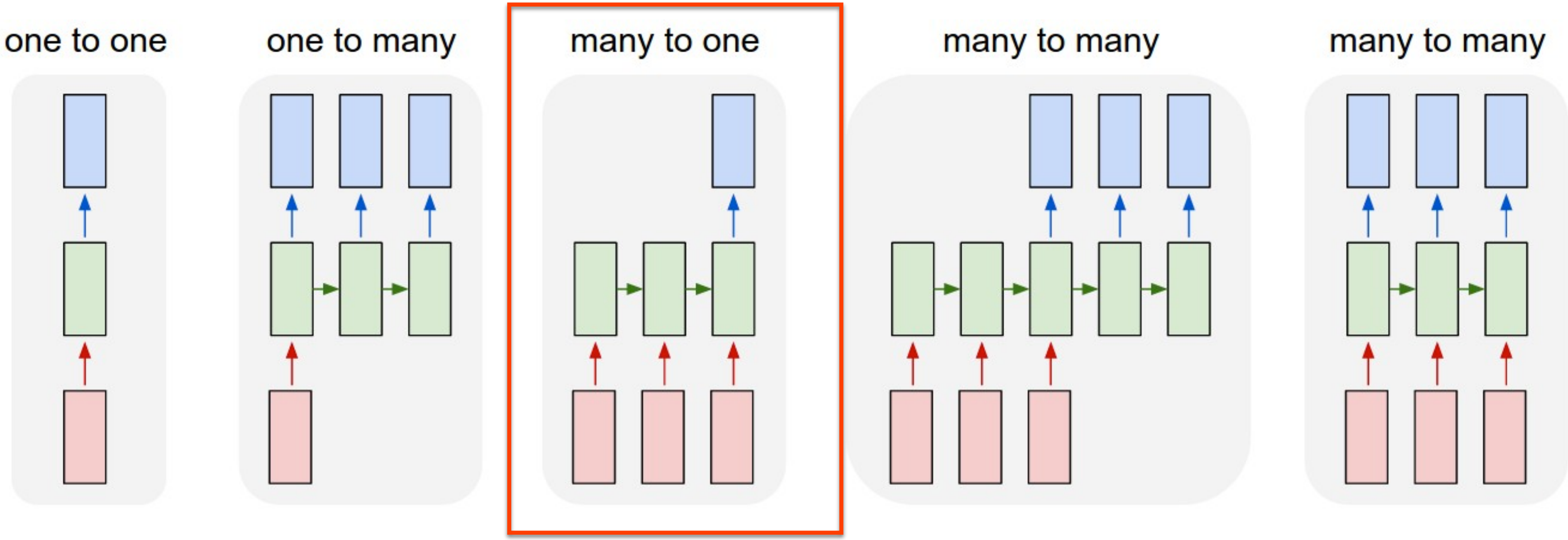
Recurrent Networks: Problem Types

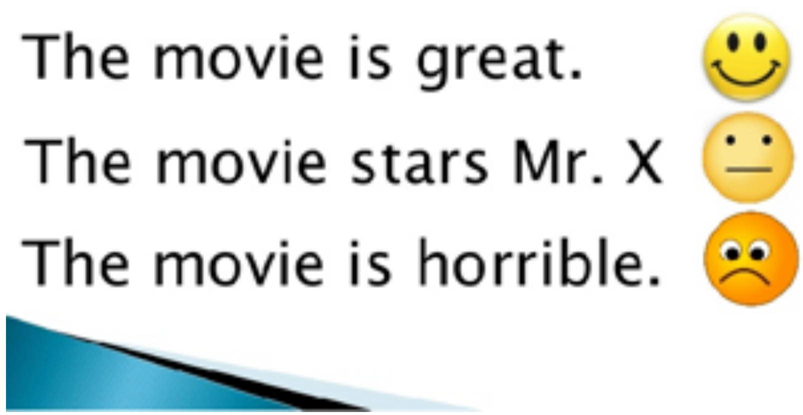


Recurrent Networks: Problem Types



Recurrent Networks: Problem Types





Recurrent Networks: Ontology Classification

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

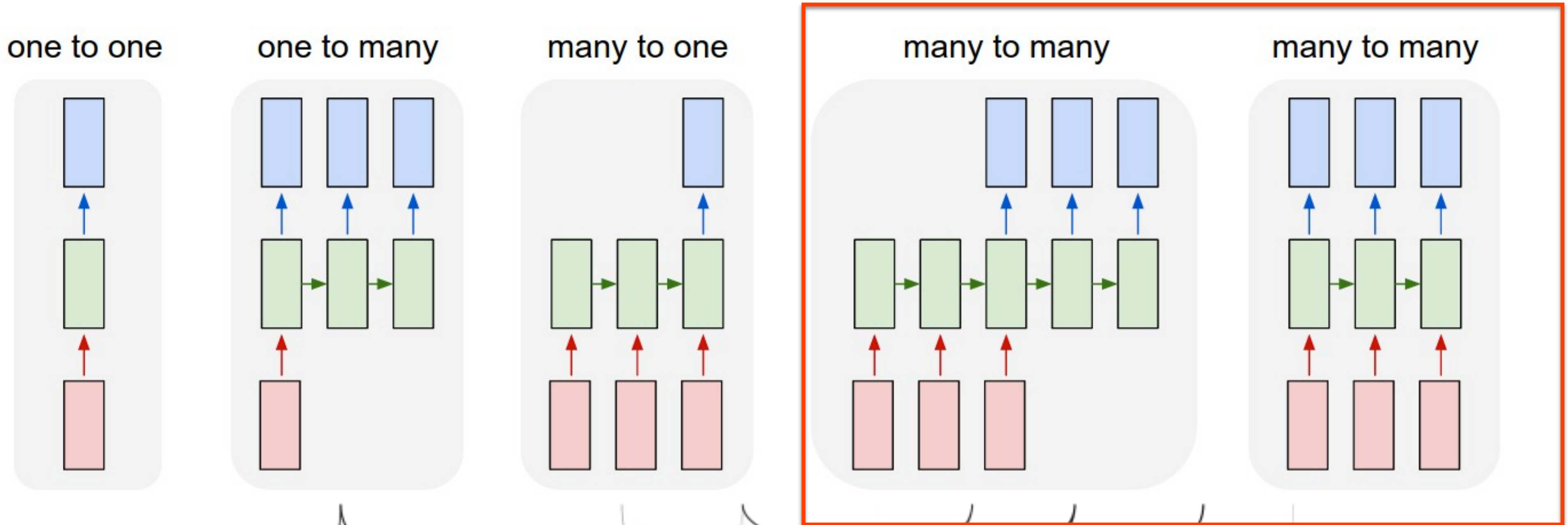
Shaun Norris (born 14 May 1982) is a South African professional golfer.Norris plays on the Sunshine Tour where he has won twice. He won the inaugural Africa Open in 2008 and the Nashua Masters in 2011. He also began playing on the European Tour in 2011 after graduating from qualifying school.

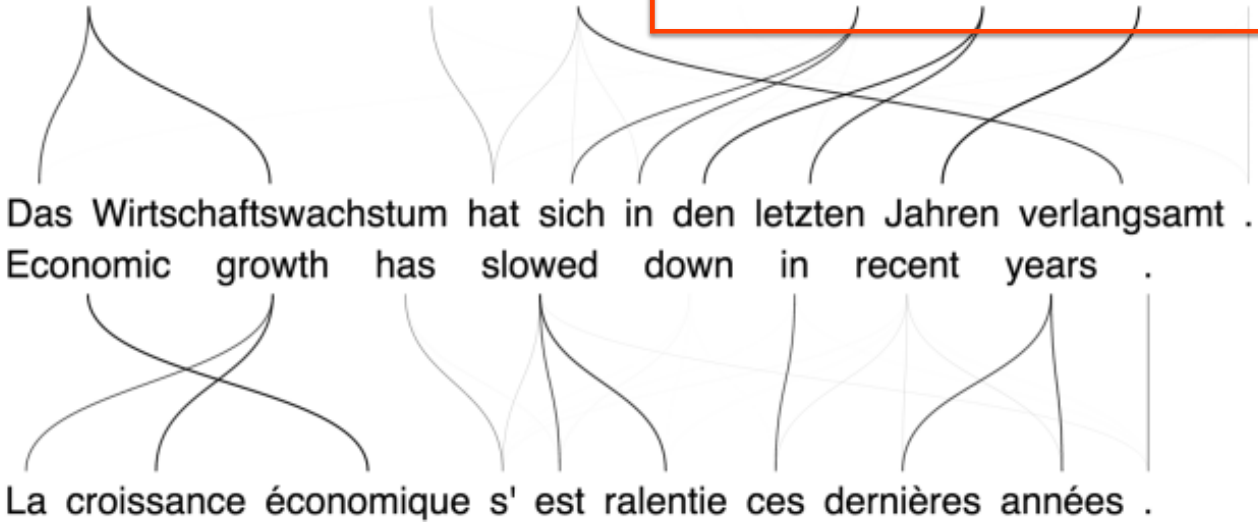
Athlete

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.

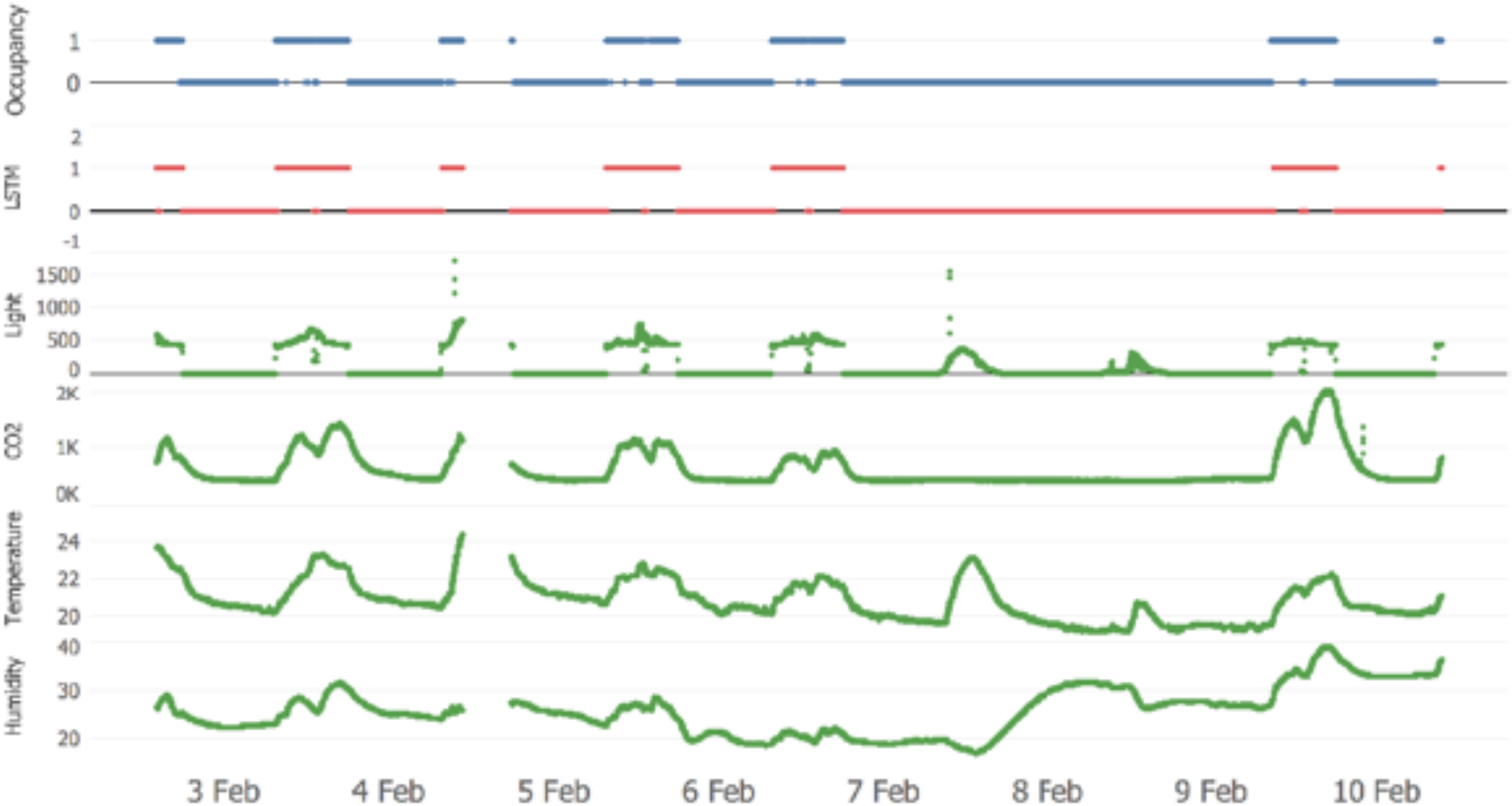
Company

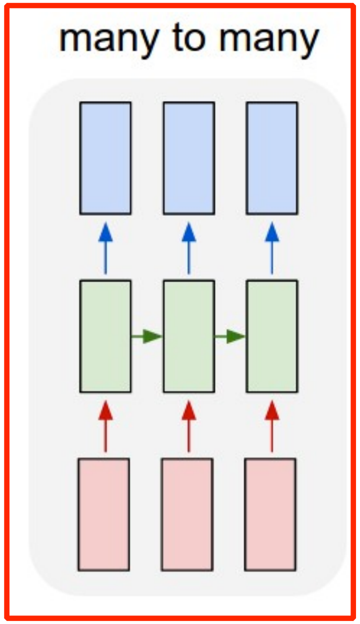
Recurrent Networks: Problem Types





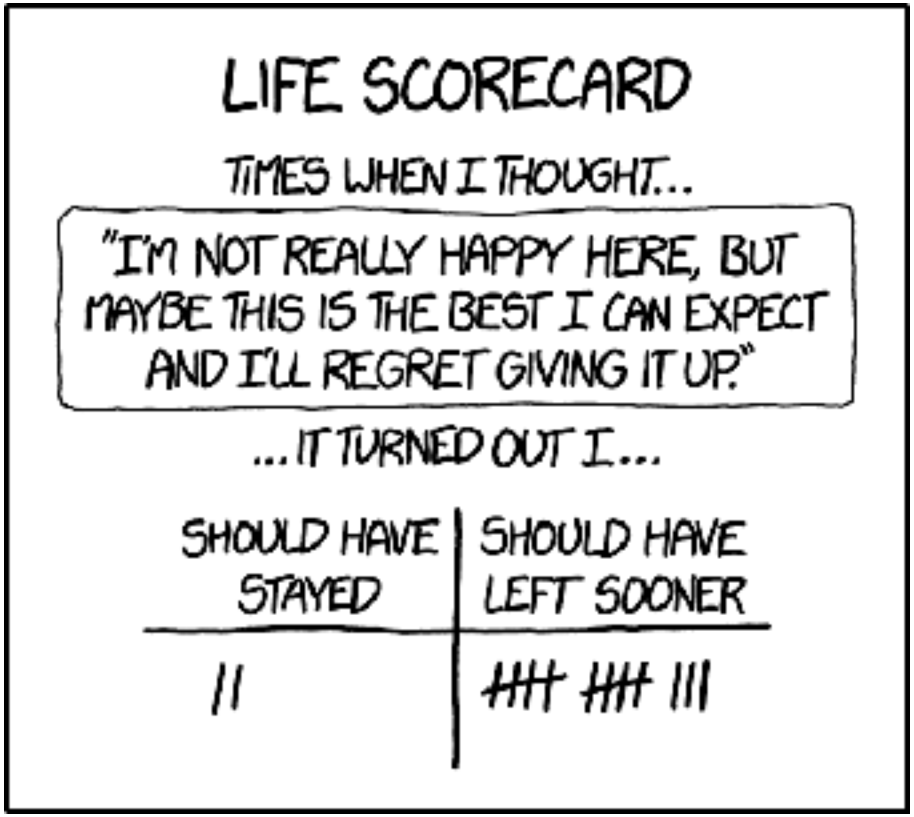
Recurrent Networks: Problem Types





sequence to sequence

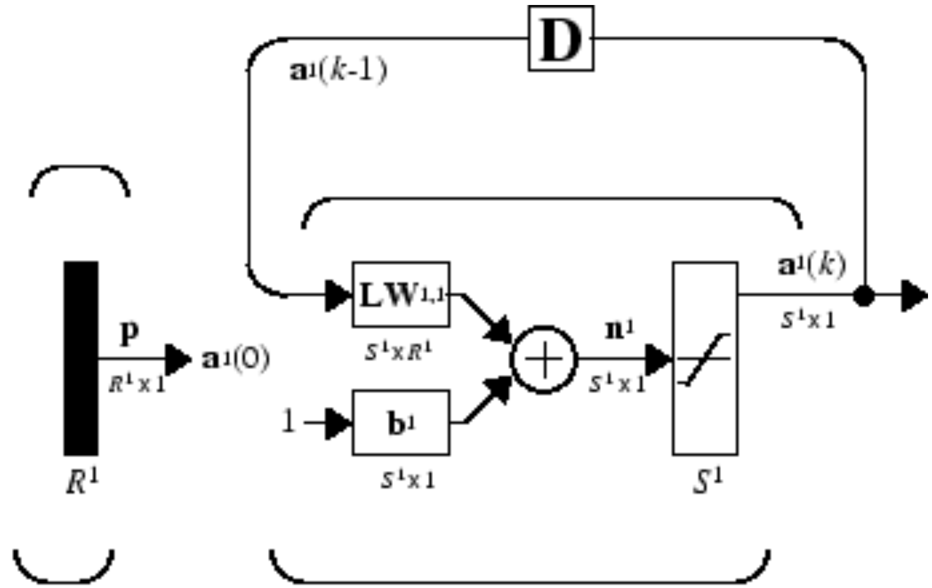
History of Recurrent Neural Networks



History of Recurrent Networks

- Hopfield Network, 1982

John Hopfield, Princeton



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

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

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

Contribution:
Training with Feedback

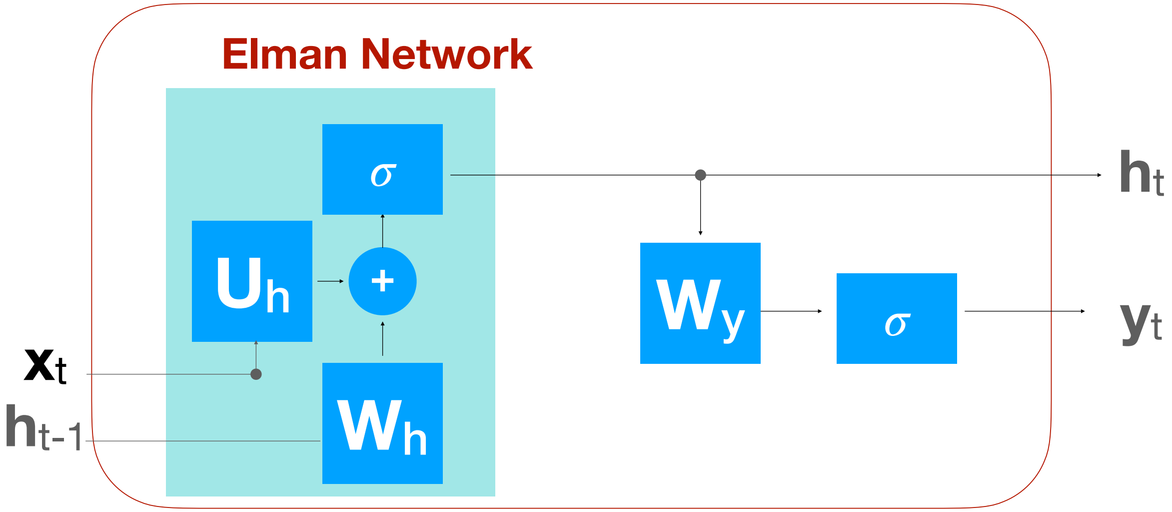
History of Recurrent Networks

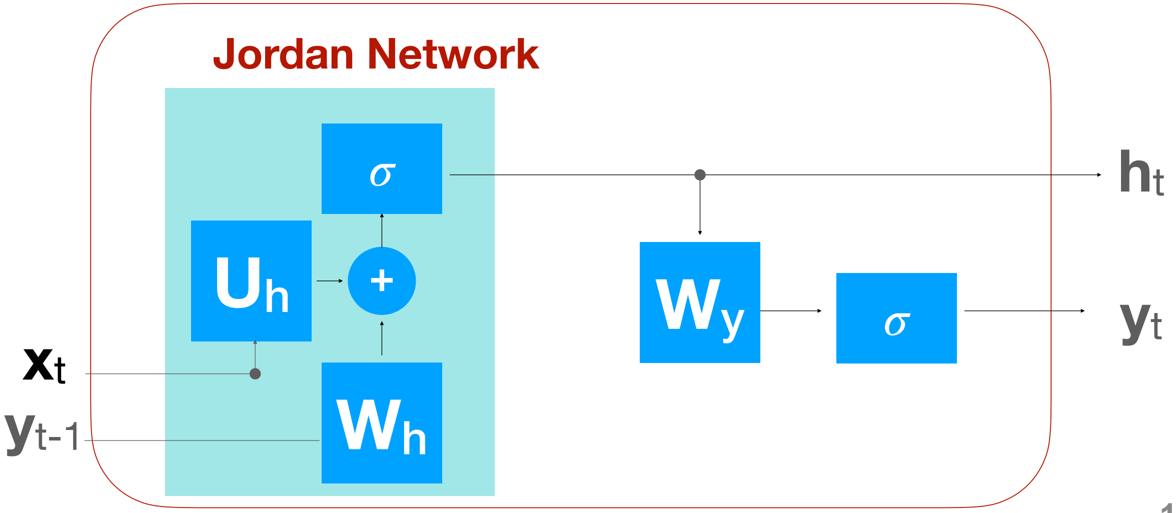
Contribution:
Time Steps for Unrolling
Separated output / state

- Elman/Jordan Networks, ~1988

Jeffrey Elman, UCSD

Michael Jordan, Berkeley

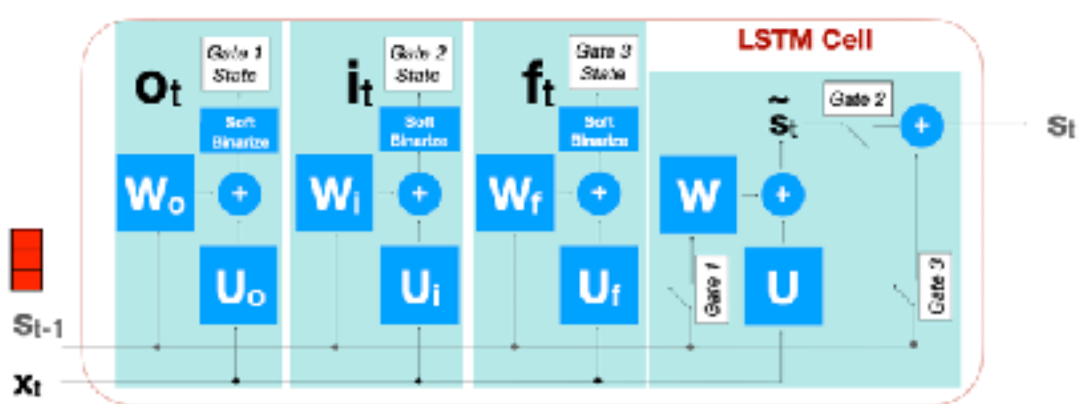




- Long Short Term Memory, ~1997 - 2010

Sepp Hochreiter, Many Universities

Jürgen Schmidhuber, Switzerland



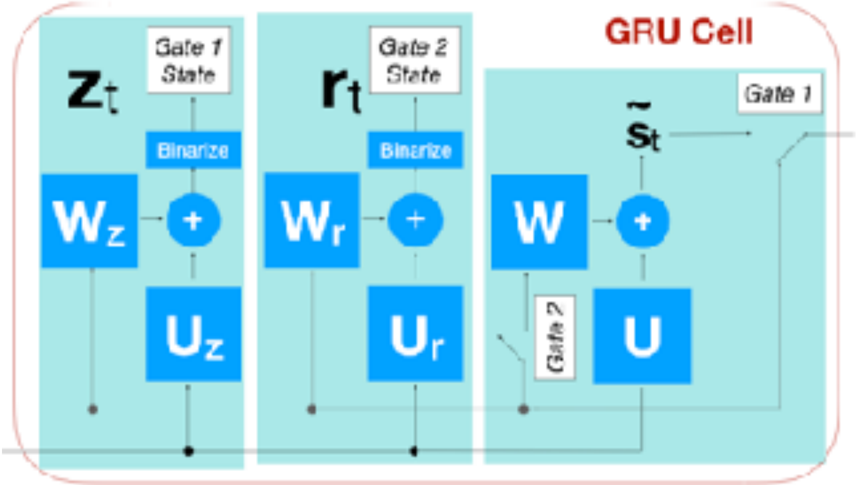
More on these later

Contribution:
Long Duration Memory
State Vector Separate from Output

- Gated Recurrent Units, ~2014

Yoshua Bengio

Kyunghyun Cho, Professor at NYU



More on these later

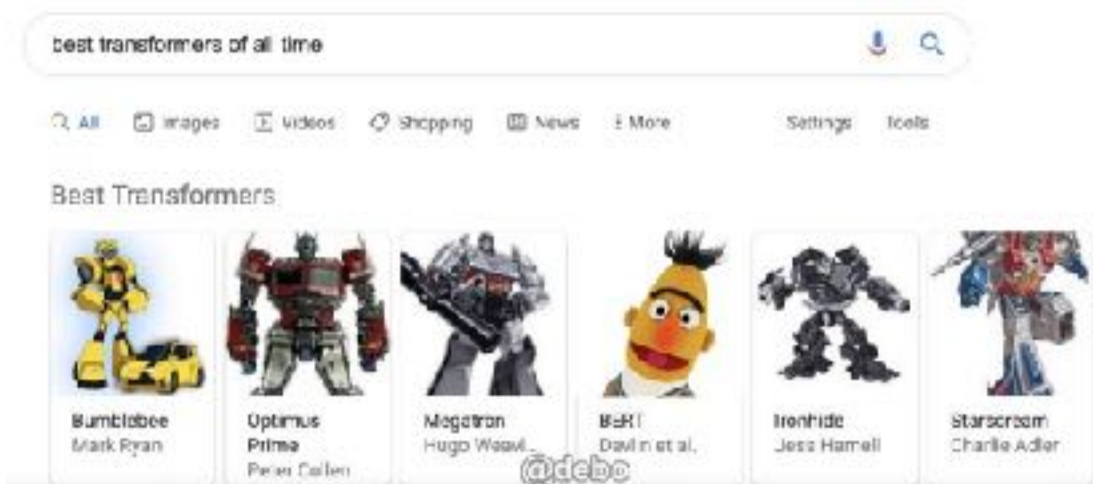
Contribution:
Forced Decision on State Vector

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-present)

This course

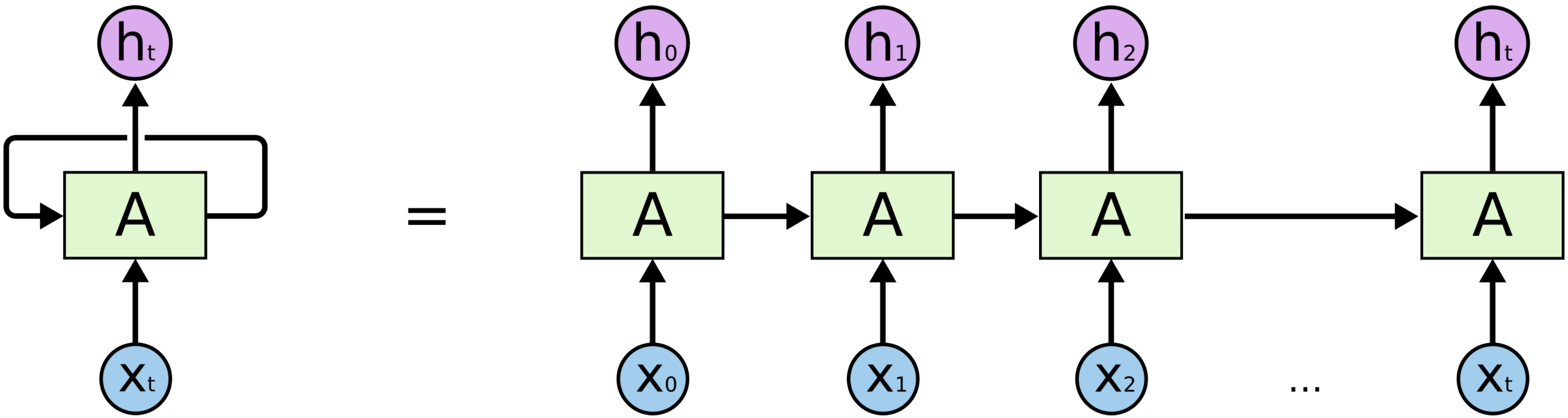
NLP Course



Basics of Recurrent Neural Networks

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

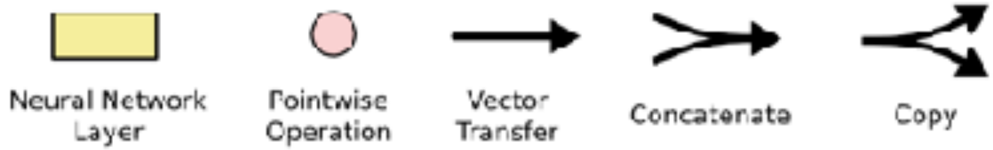
Recurrent Networks: Main Idea



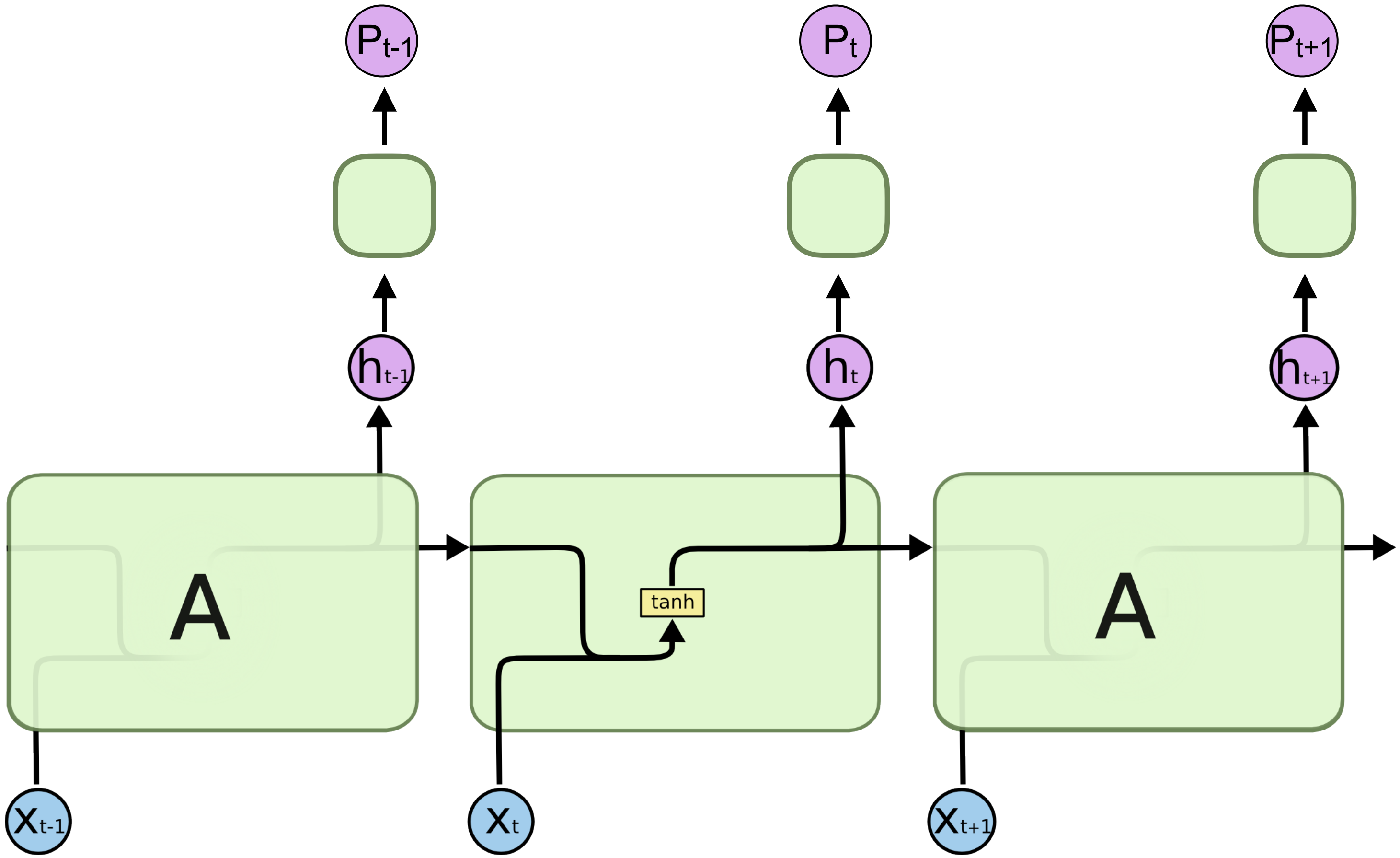
compact

unrolled

Starting Basic



- basic RNN



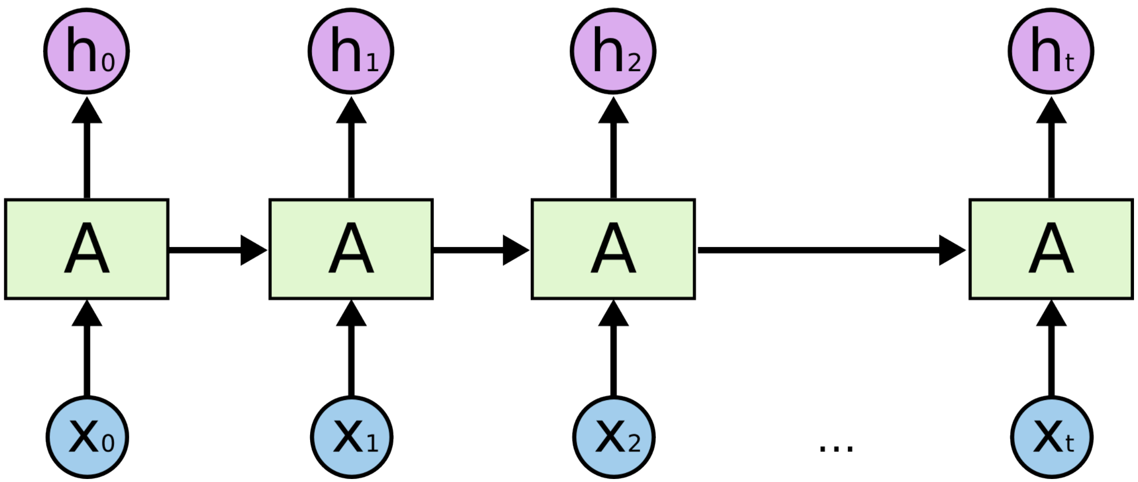
$$h_t = \tanh(W_A (X_t \oplus h_{t-1}) + b_A)$$

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

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

Recurrent Networks: Representation

- python:

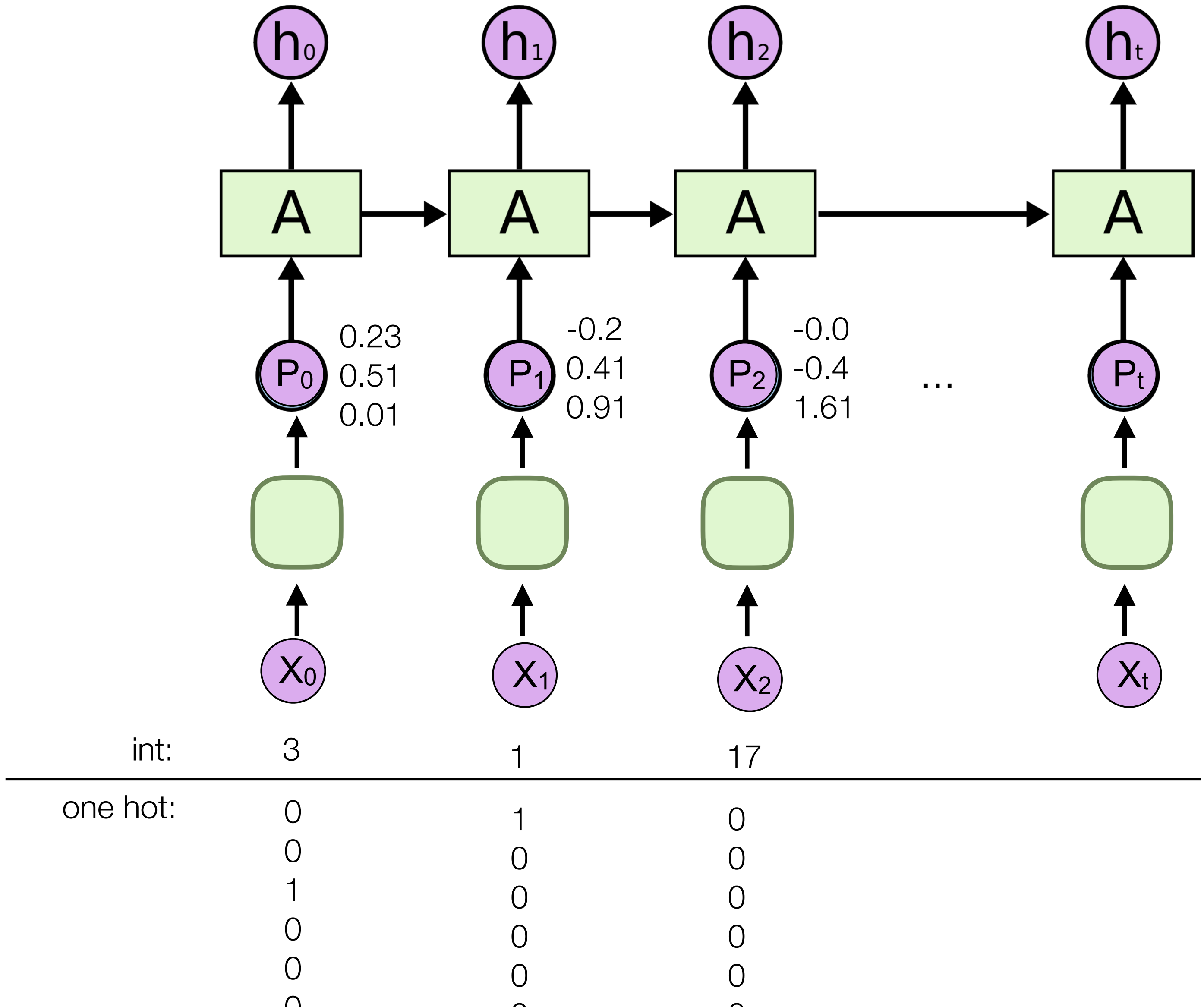


character:	C	A	T
word:	The	quick	brown
int:	3	1	17
one hot:	0 0 1 0 0 0 0 ... 0 0	1 0 0 0 0 0 0 ... 0 0	0 0 0 0 0 0 0 ... 1 0

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

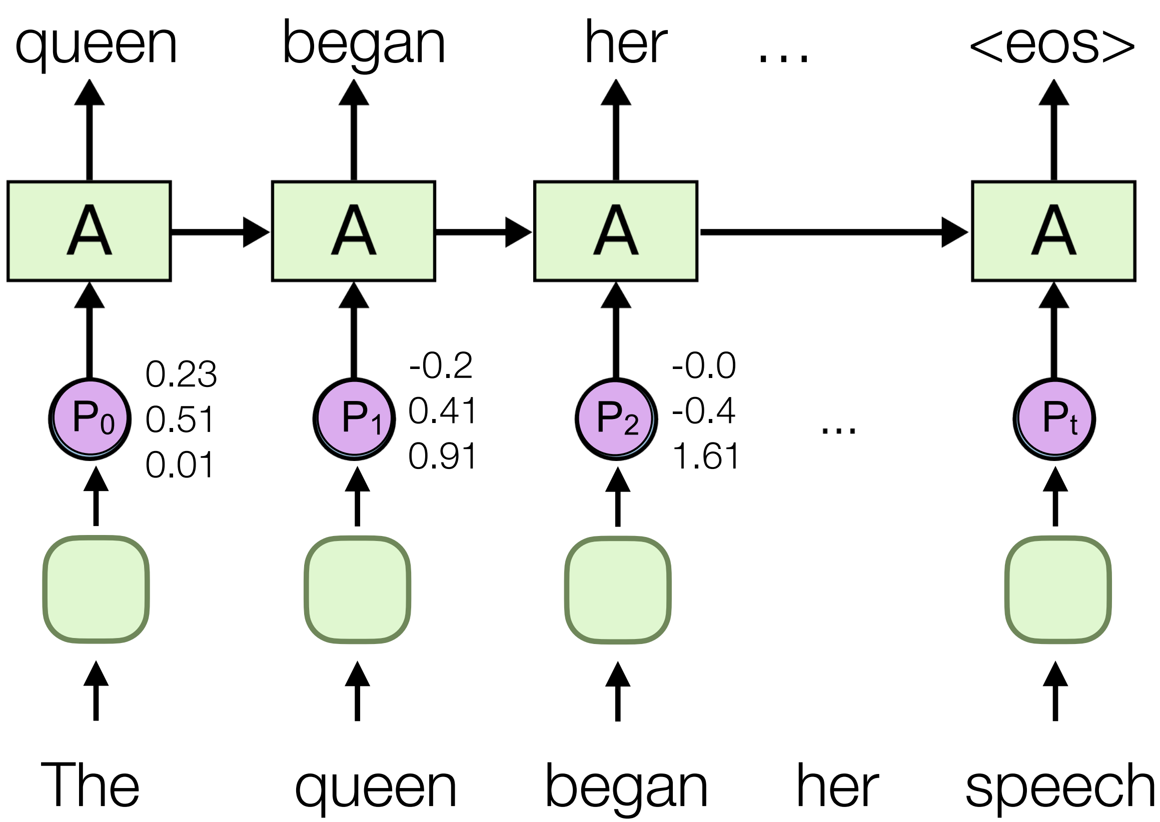
http://r2rt.com/recurrent-neural-networks-in-tensorflow-i.html

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

Global Vectors for Word Representation

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

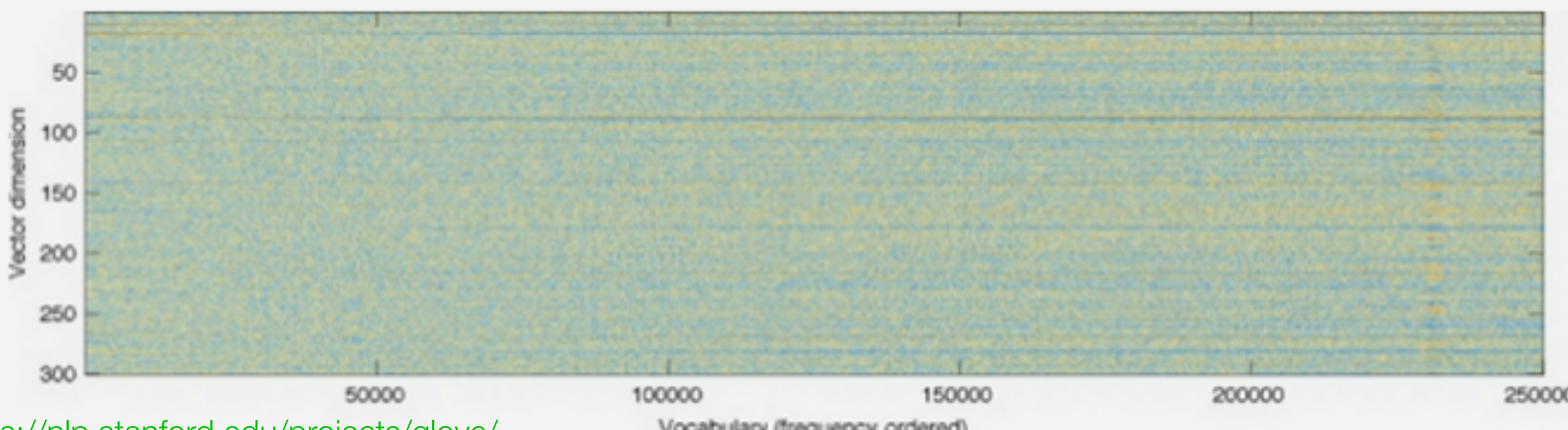

3. litoria


4. leptodactylidae


5. rana


7. eleutherodactylus

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

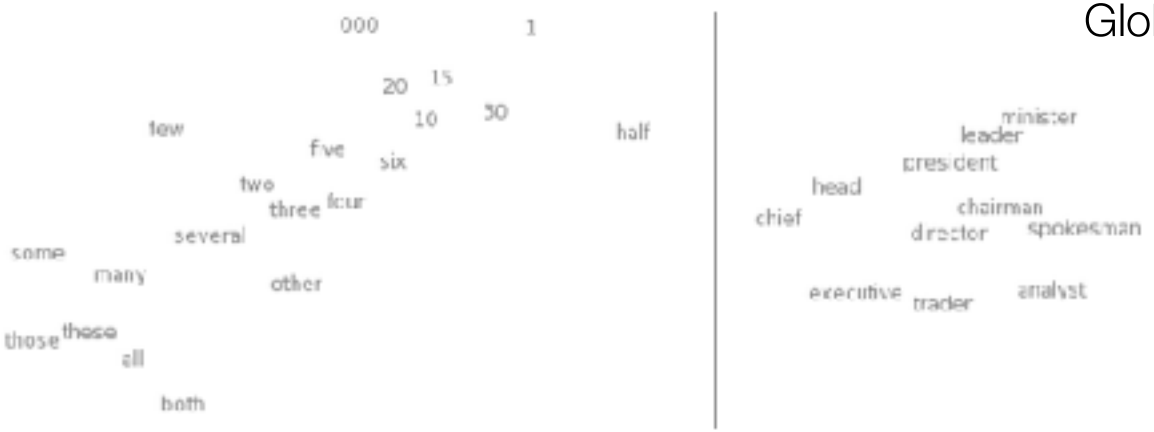


https://nlp.stanford.edu/projects/glove/

Word Embeddings: proximity

GloVe

Global Vectors for Word Representation



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

FRANCE	JESUS	XBOX	REDDISH	SCRATCHED	MEGABITS
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
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)

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

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

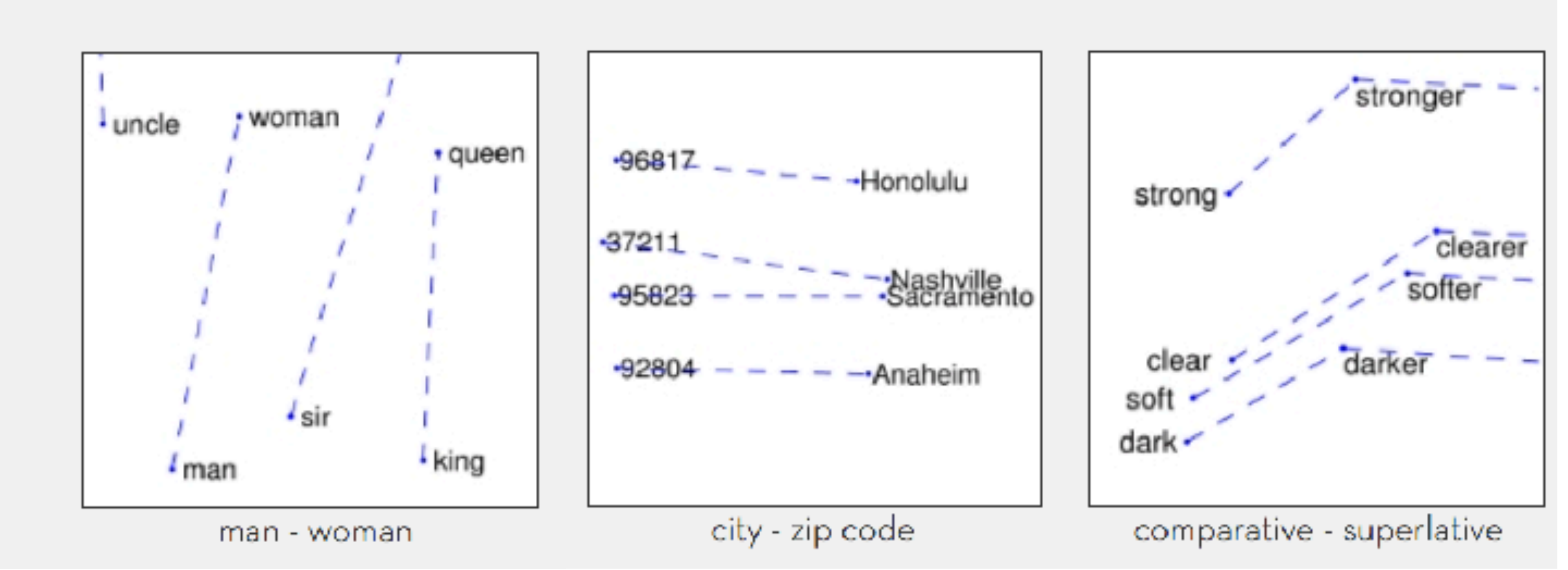
The **chief** called the **council** to order.

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

Word Embeddings: Analogy

GloVe

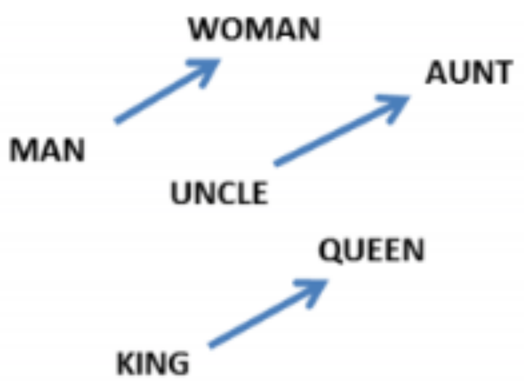
Global Vectors for Word Representation



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

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

Word Embeddings: Analogy



From Mikolov *et al.* (2013a)

GloVe

Global Vectors for Word Representation

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

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

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

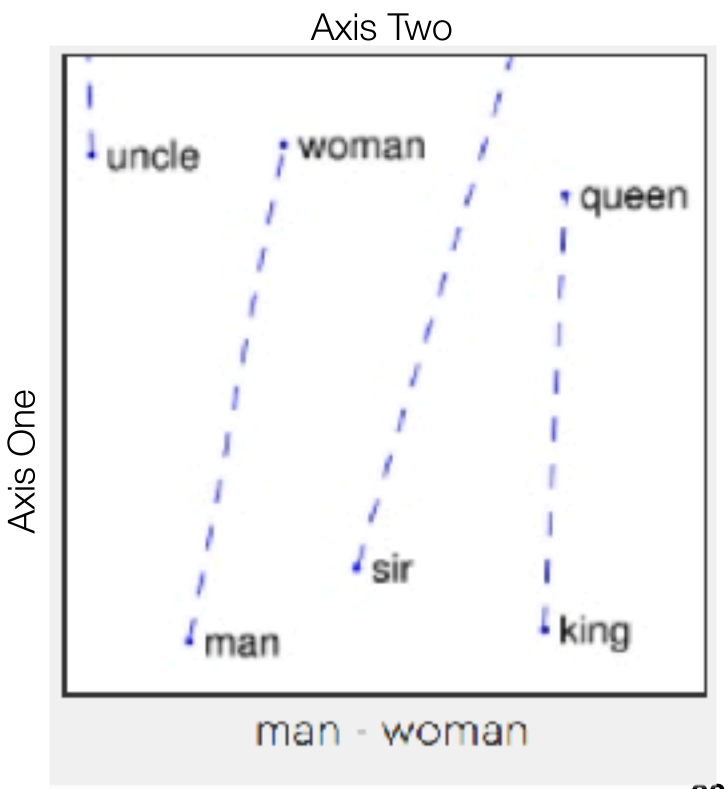
Relationship pairs in a word embedding. From Mikolov *et al.* (2013b).

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

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

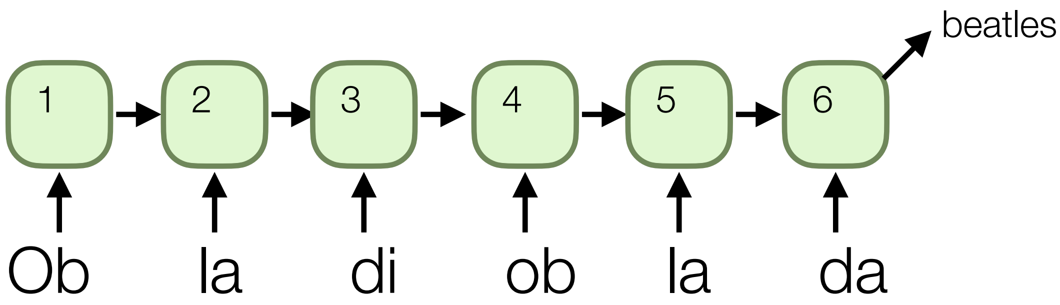
Self Test: Analogy

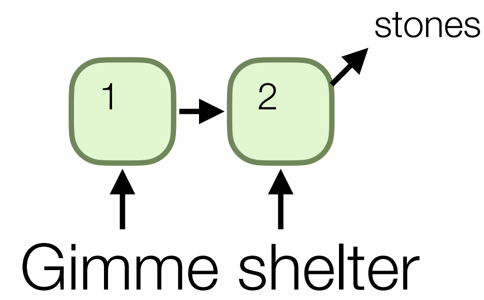
- Each axis on the embedding plot below corresponds to:
- A. a weight inside the embedding layer
- B. an average of weights inside the embedding layer
- C. the average of the one hot encoding for a word
- D. an output of the embedding layer



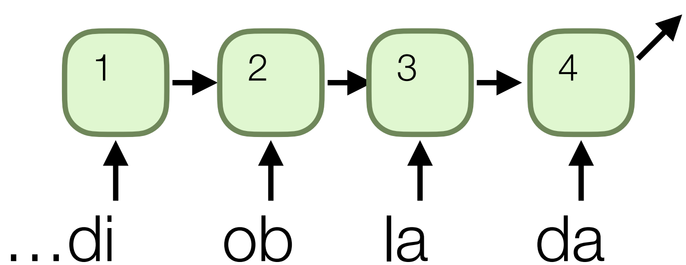
Practical Logistics: Sequence Length

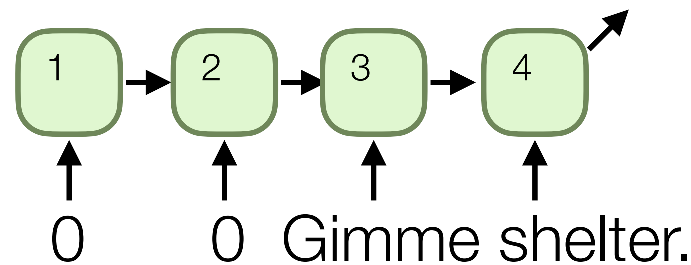
- option A: dynamic length sequences





- option B: padding/clipping





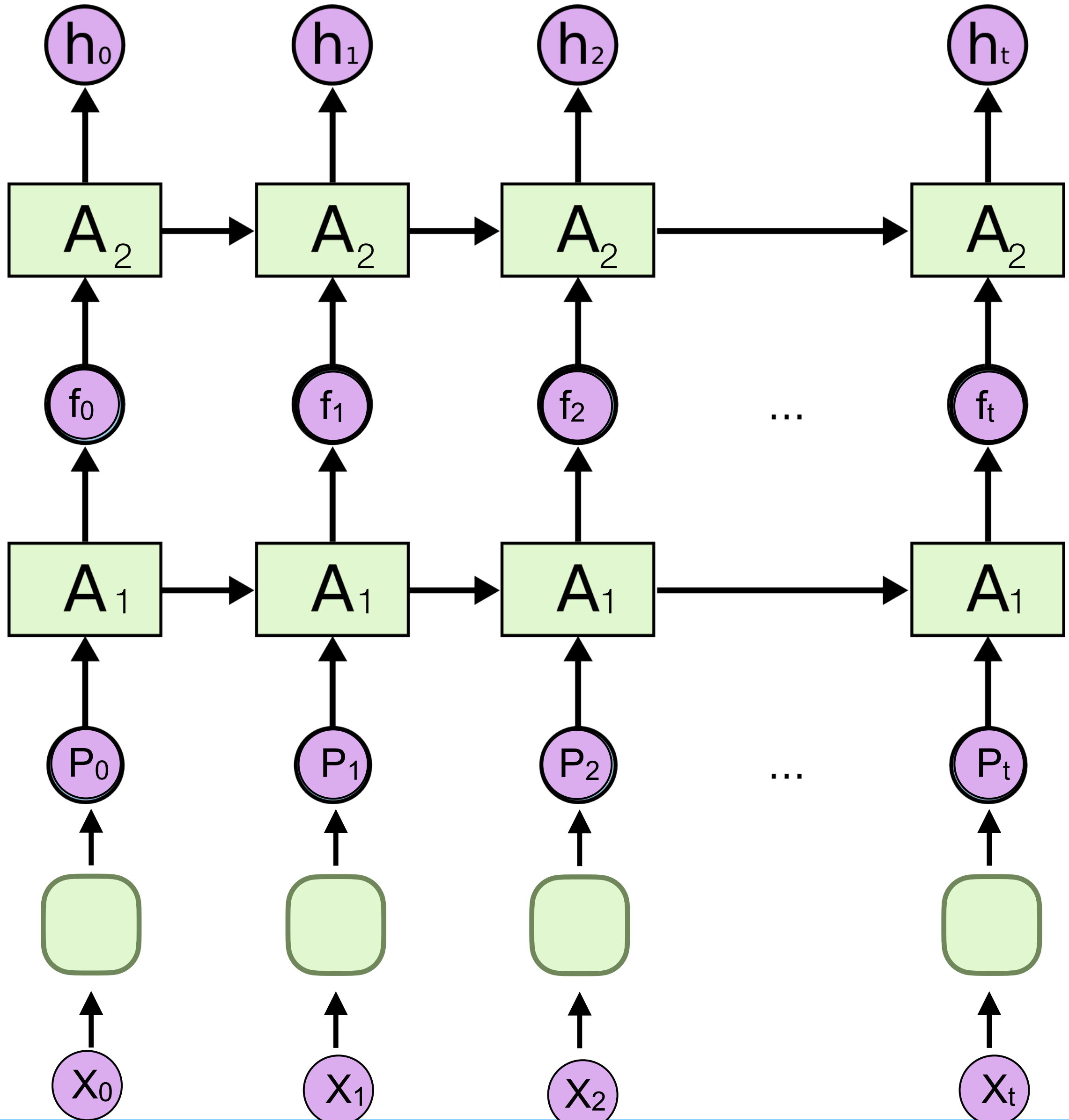
- main difference:
  speed based on computation graph design

Self Test

- The main reason dynamic length is slow is because:
  - A. the computation graph must be updated
  - B. weights must be tied together for each recurrent node
  - C. the weights must be multiplied until the output converges
  - D. the unrolling operation takes some time

Sequence Stacking

keras:
return sequences

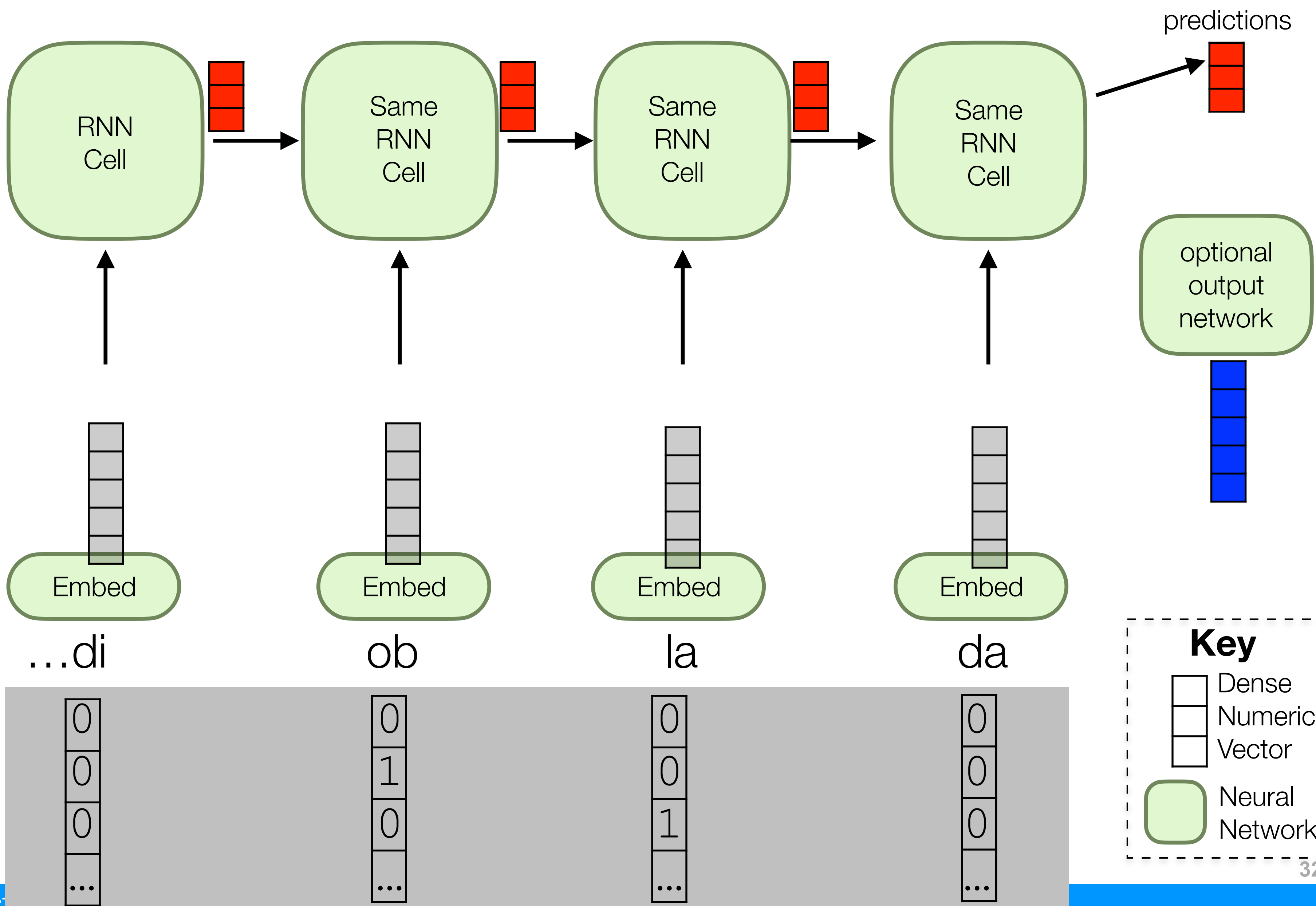


keras:
add final dense layer for prediction from last element only

keras RNN default:
return last element only
go to dense layer for prediction

keras embed:
length of embedding is meant for sequences

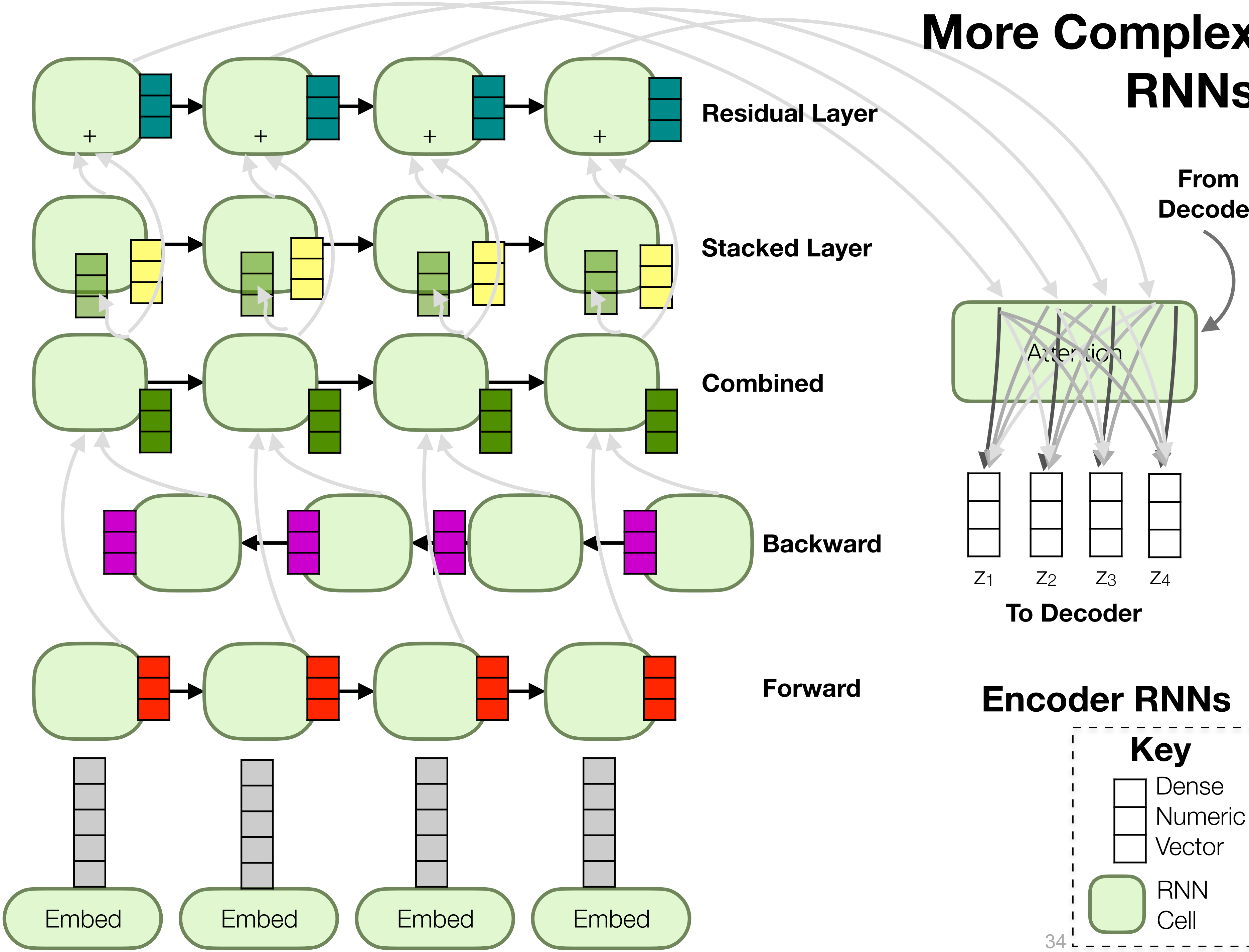
General recurrent flow (many to one)



Next time

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

More Complex RNNs

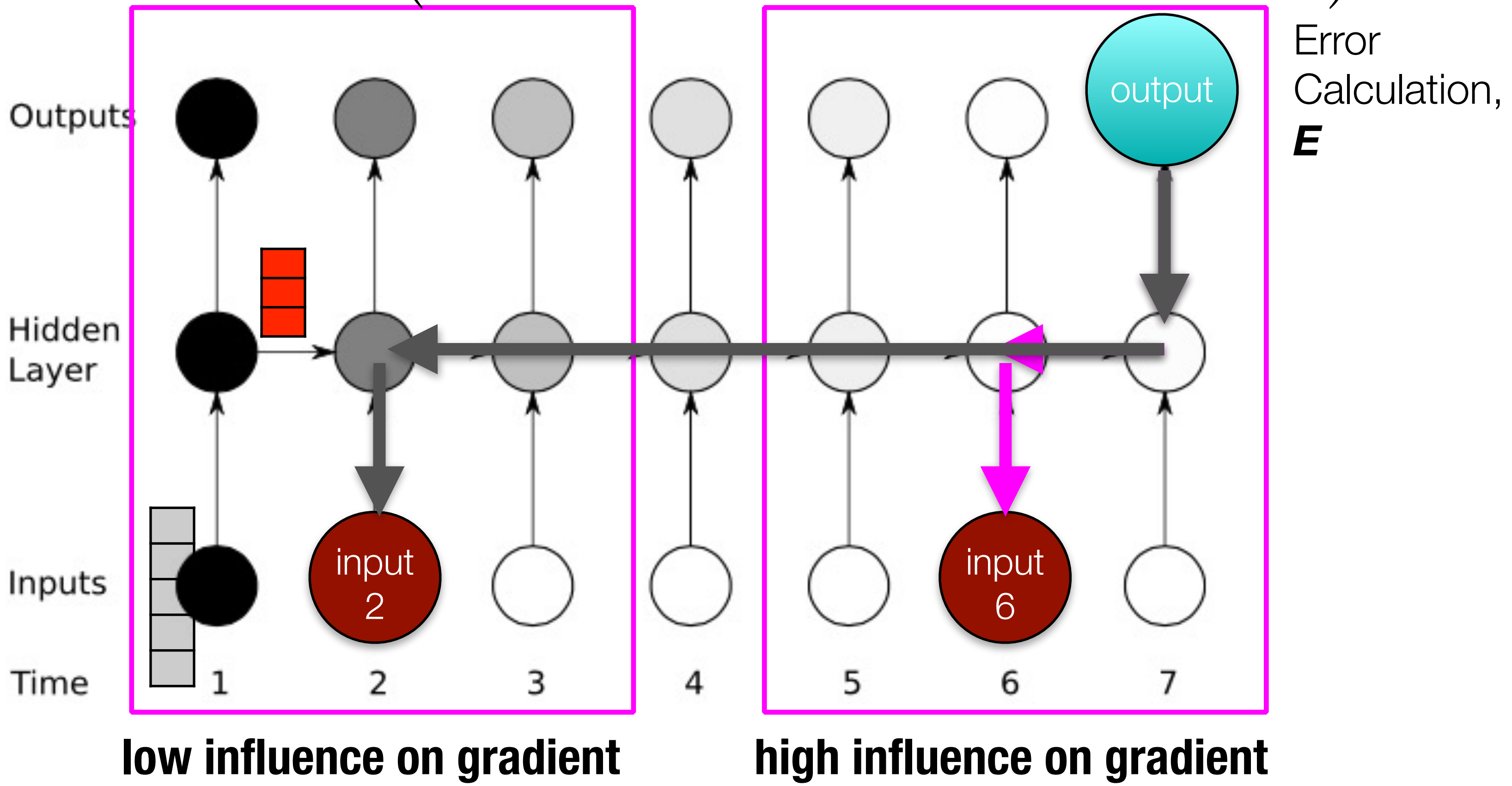


Encoder RNNs

Recurrent Networks, the Age Old Problem

- vanishing gradients: why are these a problem?

$$h_t = U \cdot X_t + W \cdot \left(U \cdot X_{t-1} + W \cdot \left(U \cdot X_{t-2} + W \cdot h_{t-3} \right) \right)$$



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

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