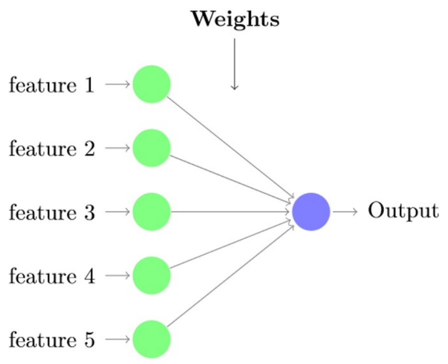
Neural sequence modeling

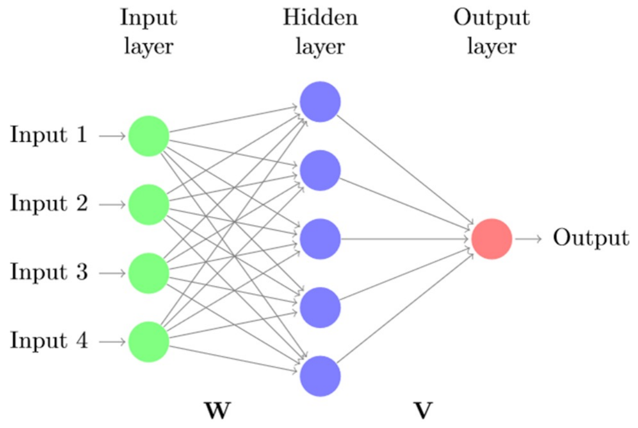
Jakob Prange

(with some updates by Michael Kranzlein)

Graphics and inspirations by Taylor Arnold, Fei-Fei Li, Justin Johnson, Serena Yeung, Dan Jurafsky, James Martin, Austin Blodgett

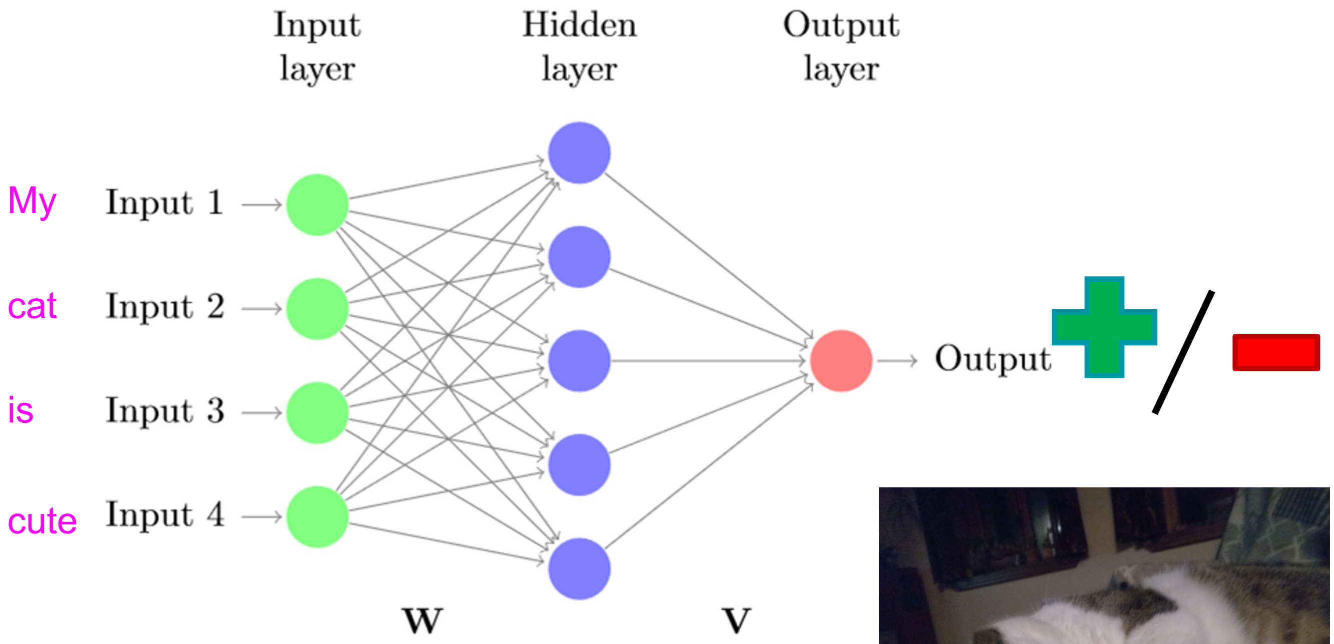
Review: Neural networks





- Simplest architecture: **Feed-forward**
- "Multilayer Perceptron"
- "Depth" = how many hidden layers
- One-to-one

Blodgett


Input layer
Hidden layer
Output layer
My
cat
is
cute
Input 1
Input 2
Input 3
Input 4
Output
W
V

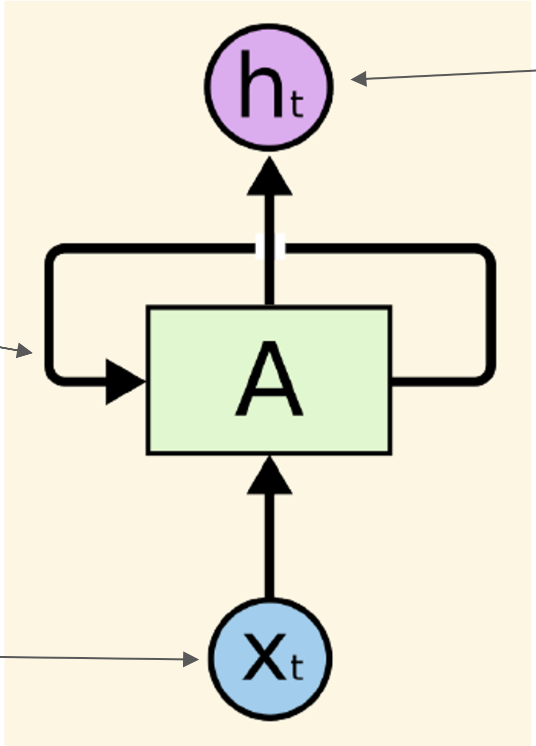
Neural networks and text

- Number of nodes per layer is fixed
  - Number of inputs is fixed
- Length of a sentence, word, sound signal, ...
  - **Not fixed!**
- Idea: Think of language data as streaming in over time.
  - For each new input, we update our prediction.

Today: Recurrent neural networks (RNNs)

self-loop (recurrence) on
hidden layer computation

input layer



hidden layer

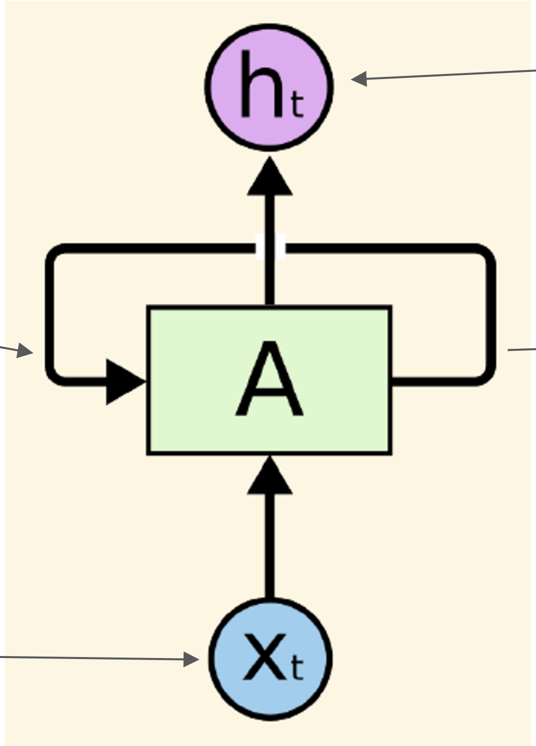
[Arnold](#)

Agenda for today

- What is an RNN?
- What can it be used for?
- How is it trained? (some math, but not too much)
    - The problem of vanishing and exploding gradients
- The LSTM model and some variants
- Interpretability

What is an RNN?

self-loop (recurrence) on
hidden layer computation



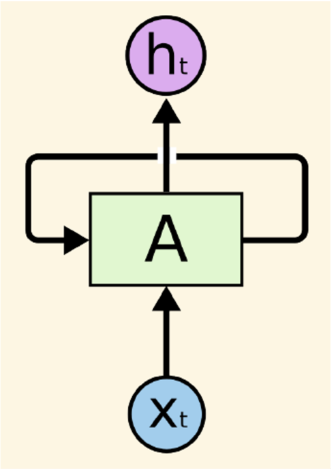
hidden layer

unroll in time...

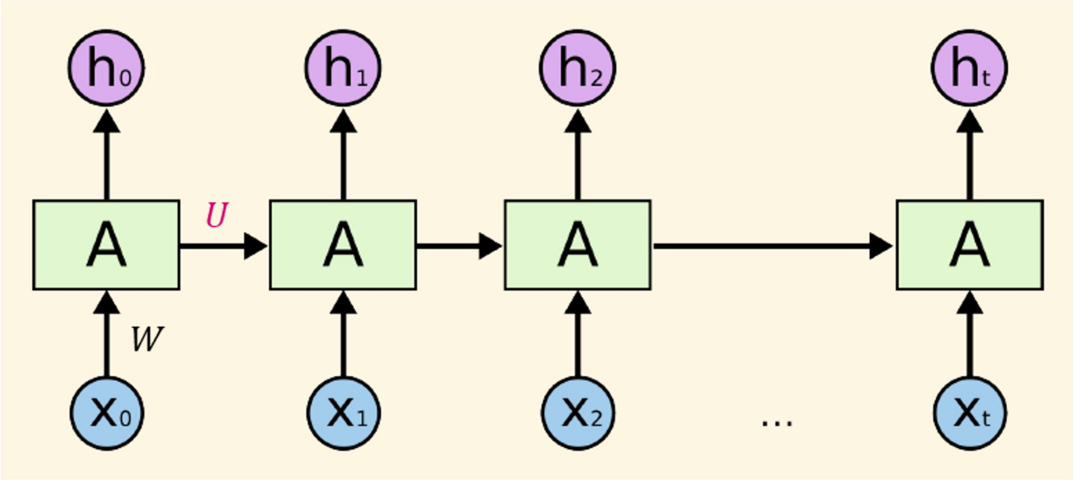
input layer

[Arnold](#)

What is an RNN?



=



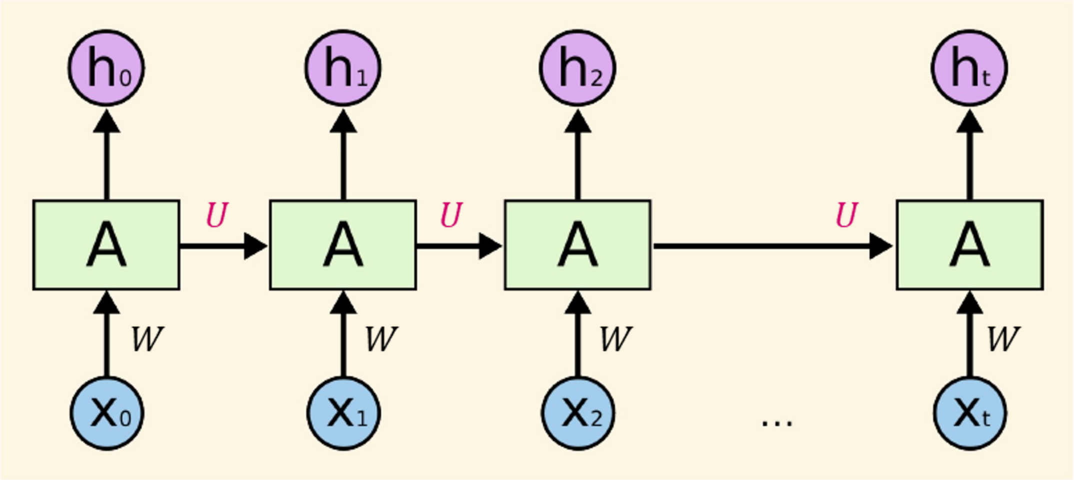
$$h_t = f(Wx_t + b + Uh_{t-1})$$

Arnold

What is an RNN?

Some notes

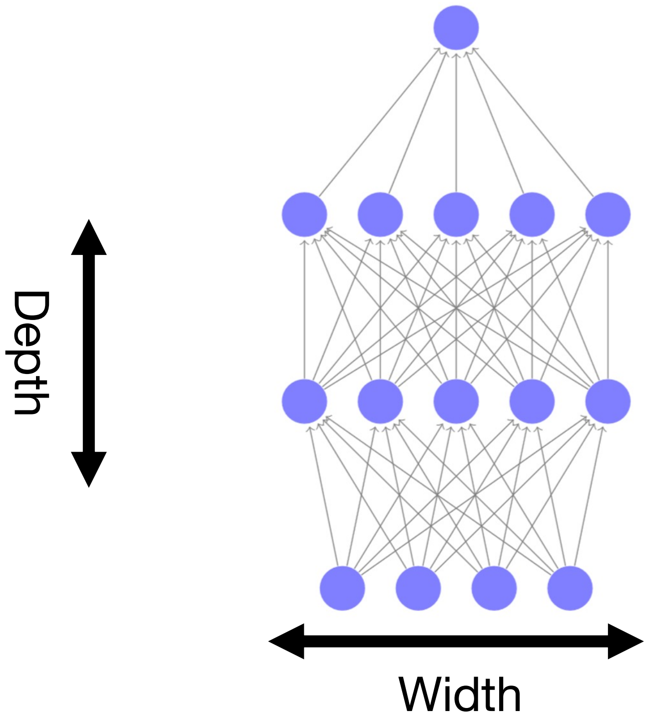
- Weights are shared between each time step
- We initialize a new RNN for each sequence!
- "Deep" in the length of the sequence

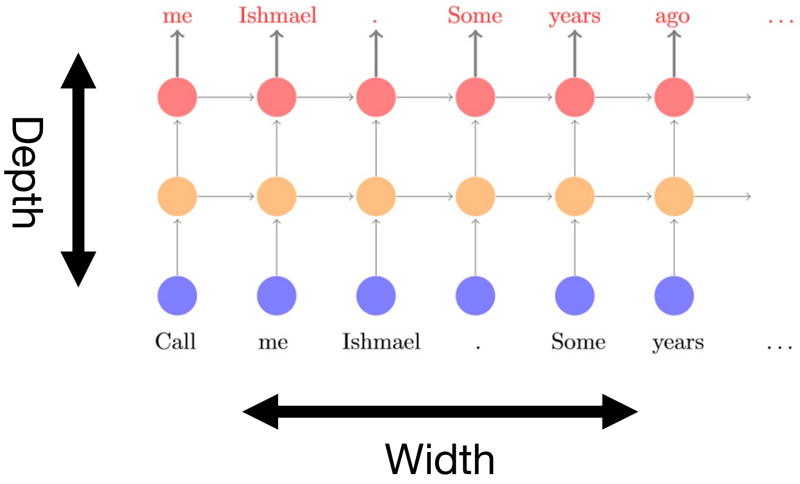


$$h_t = f(Wx_t + b + Uh_{t-1})$$

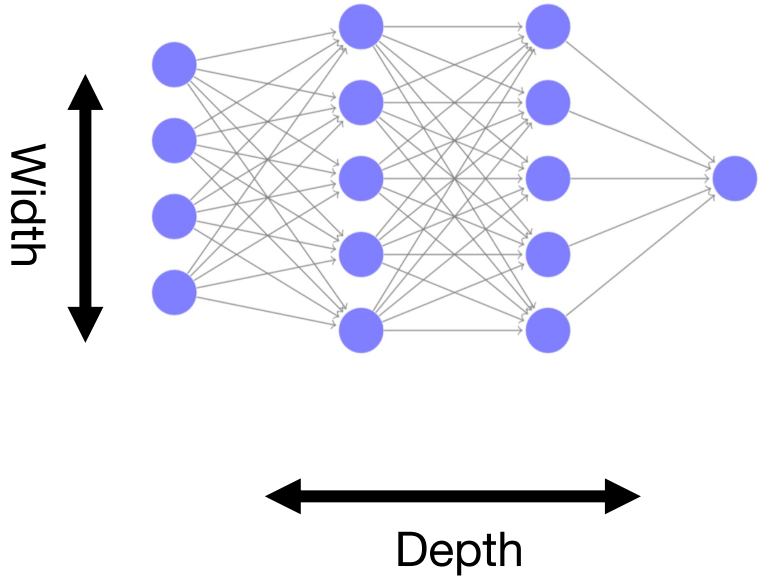
Arnold

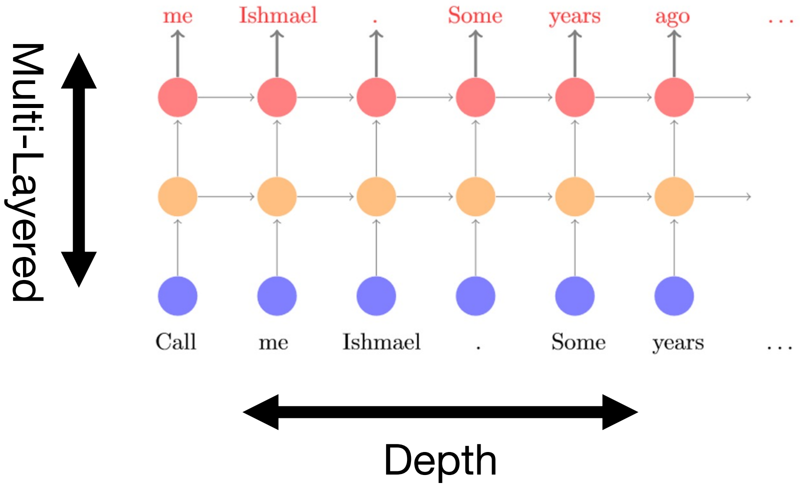
RNNs as Deep Networks





RNNs as Deep Networks



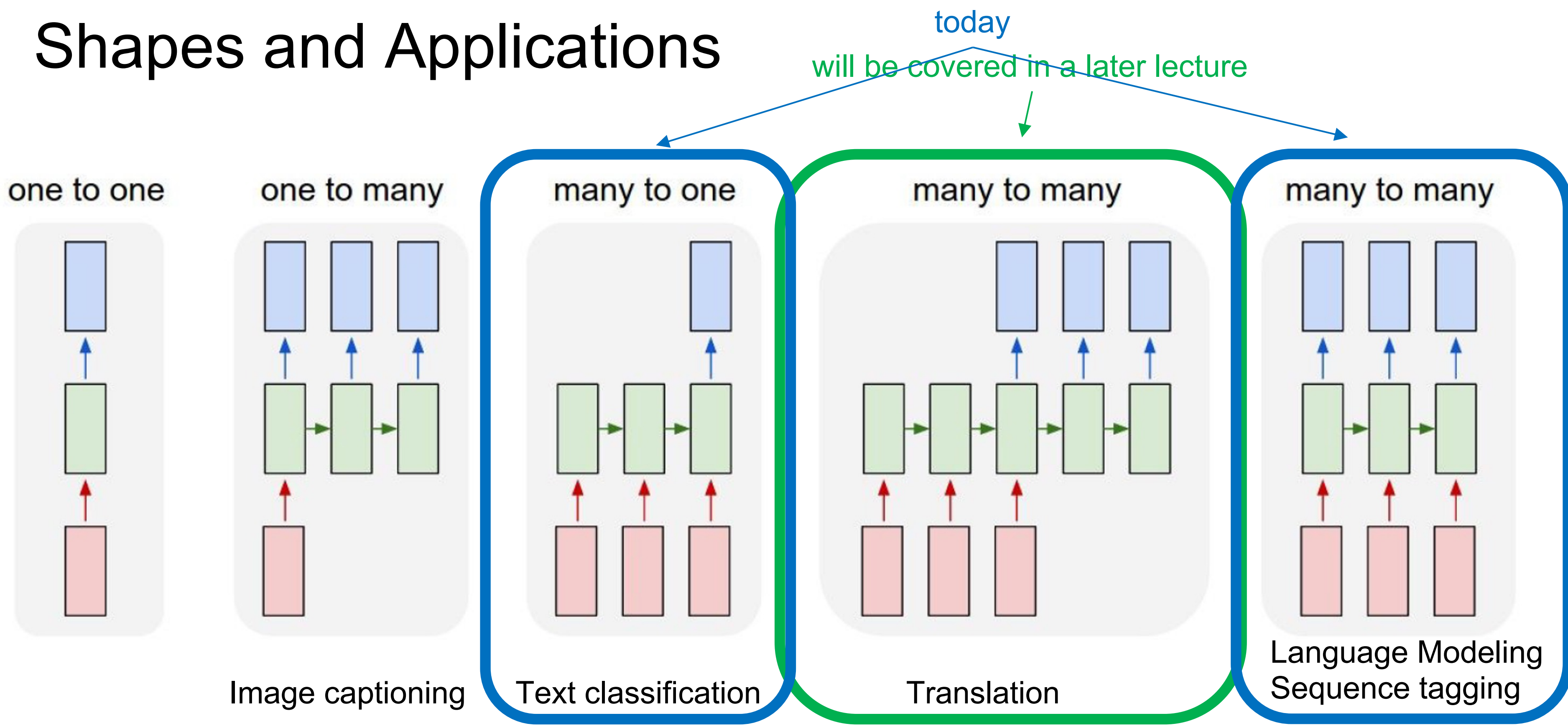


RNNs as input to other NNs

- Usually, we feed the hidden representation produced by an RNN into another layer or multi-layered network to produce a prediction, which can be…
    - per-token (tagging) or for the whole sequence (classification)
    - non-probabilistic or probabilistic (using softmax)

- Or we are interested in learning embeddings of the input itself

Shapes and Applications

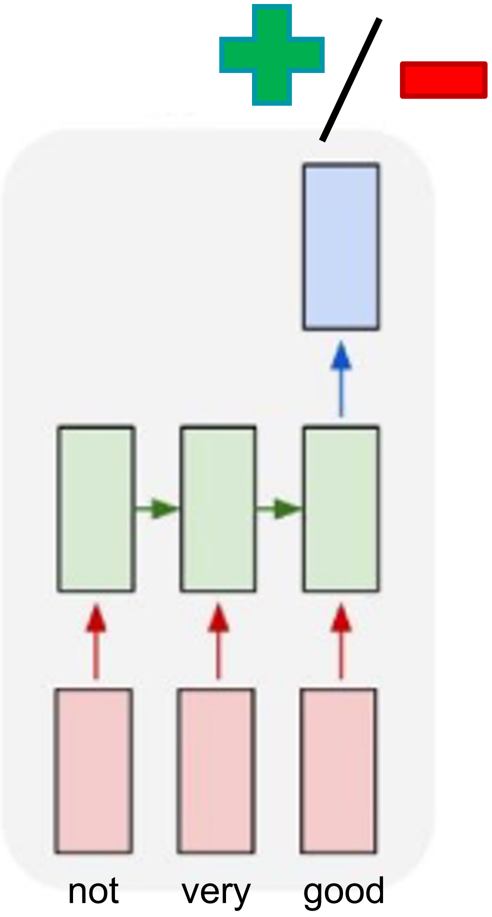


Li, Johnson, Yeung

Text classification

- Let RNN read and process input text
- Use hidden representation of last input token to make prediction for the whole sequence

- Example: sentiment analysis

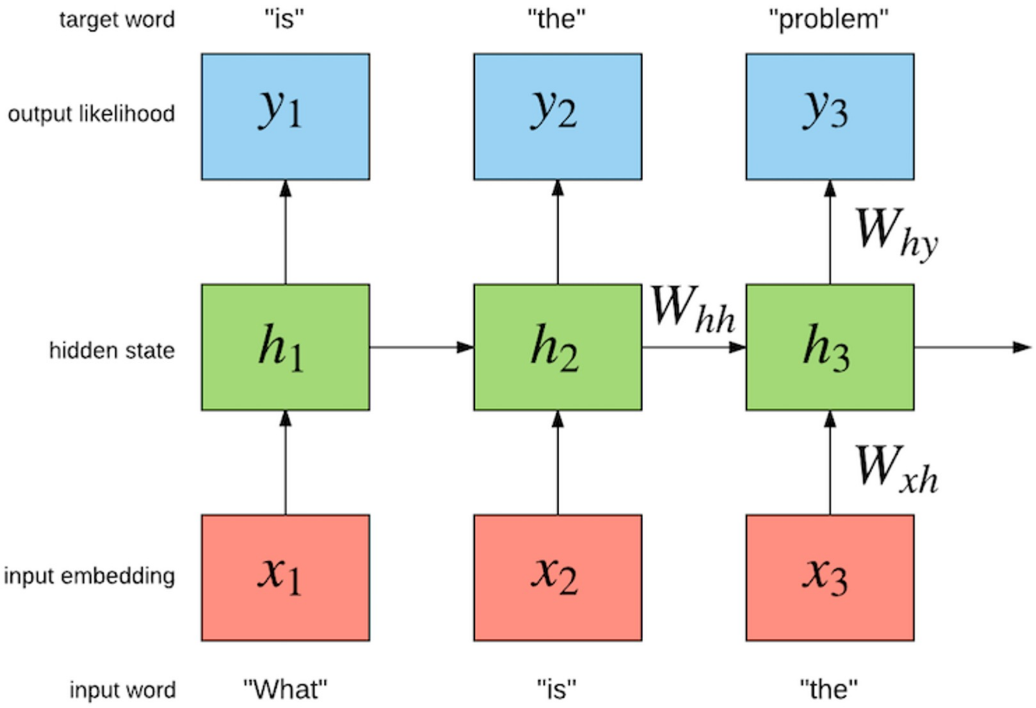


Language modeling

- Recall generative (**n-gram**) language models
  - Given the previous context, predict next word

- How can we implement this as an RNN?

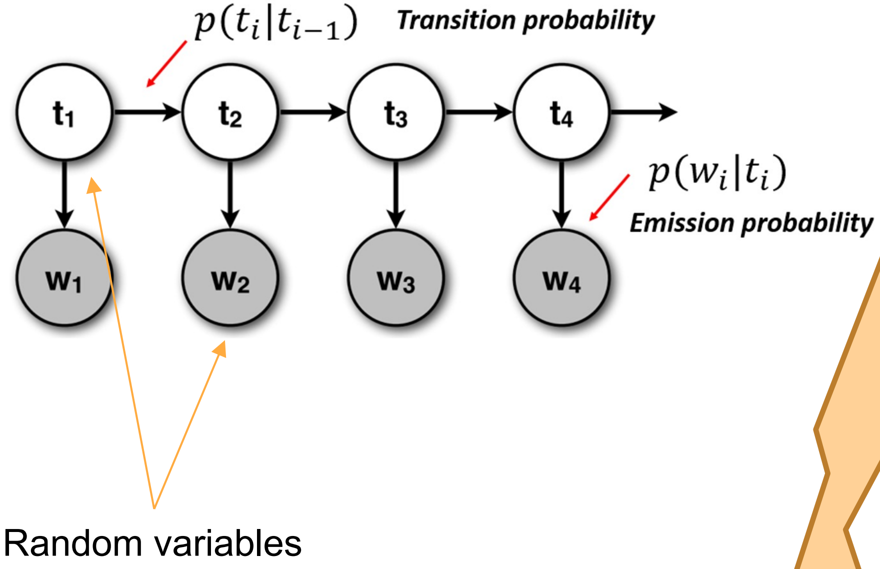
Recurrent Neural language modeling



http://torch.ch/blog/2016/07/25/nce.html

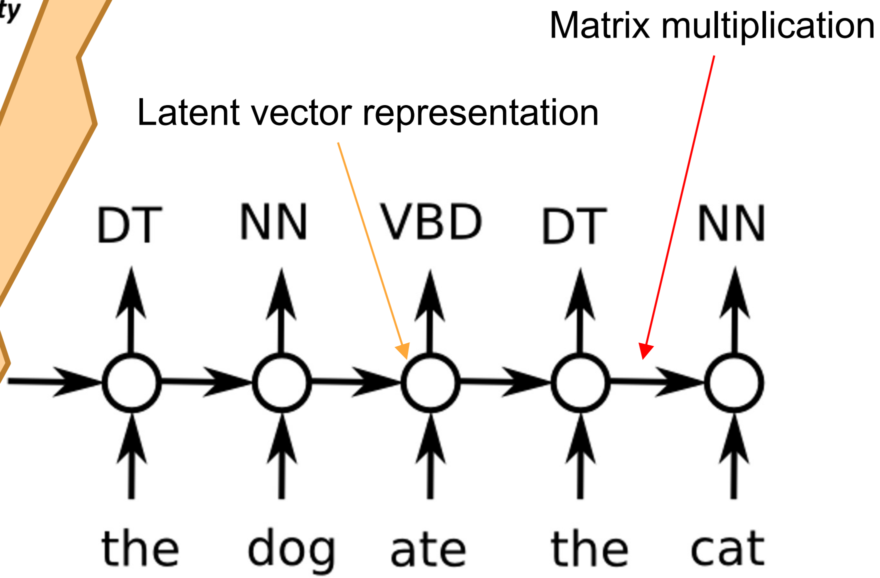
POS tagging

- Recall generative (**HMM**) POS tagging
  - Given previous POS tag, predict tag that is most likely to generate current word
  - Find optimal sequence (Viterbi)
- By default, RNNs (as neural networks in general) are discriminative, not generative!
  - Can model output directly at each timestep



HMM POS tagging

Neural POS tagging

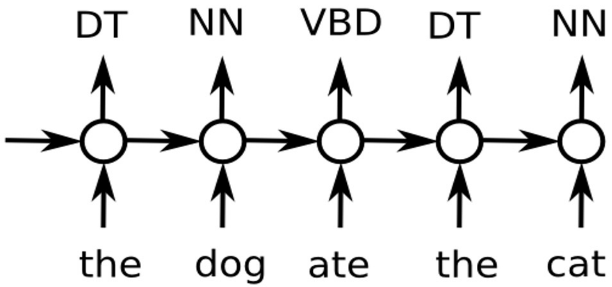


Issues with the vanilla RNN

Despite having no explicit independence assumptions, distant cells are unlikely to influence each other.

Why?

1. At **prediction** time because new inputs “overwrite” old memory
2. At **training** time because of how backpropagation works

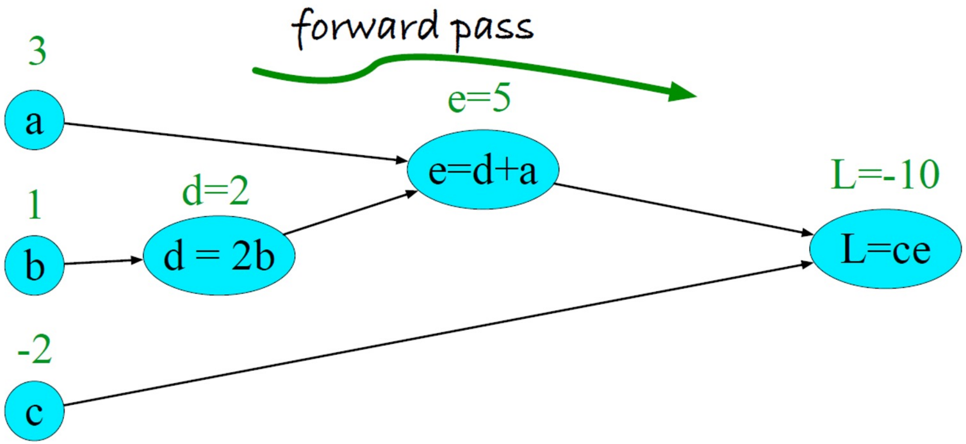


Training a Deep Neural Network

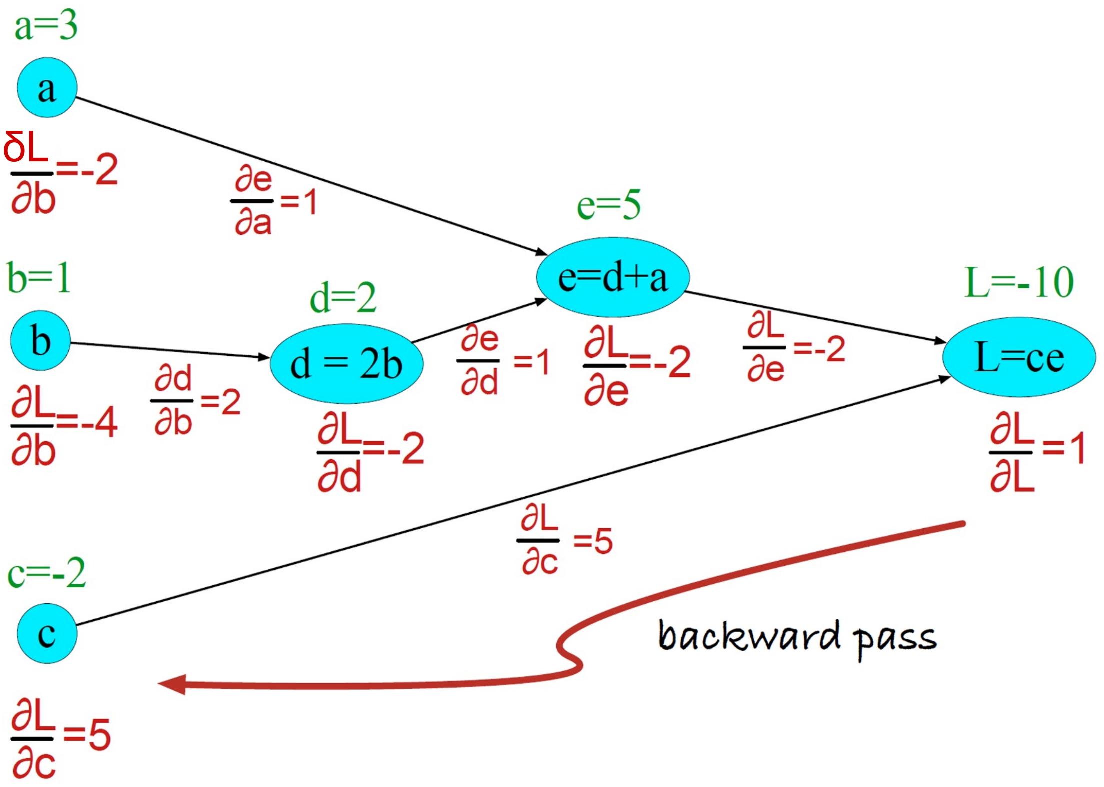
- High-level: Tuning of hyperparameters and architecture
  - Dimensionality of hidden layers
  - Dropout rate
  - Learning rate
  - Batch size
  - “Resolution” of input:
    sentences, words, characters, …

- Low-level: **Backpropagation**
  - Error-driven (minimizing loss function)
  - Lots of matrix multiplications
  - Made possible through modern computing power, especially GPUs
    (and more recently, TPUs)

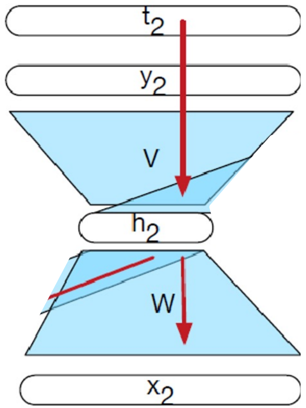
Training a Deep Neural Network



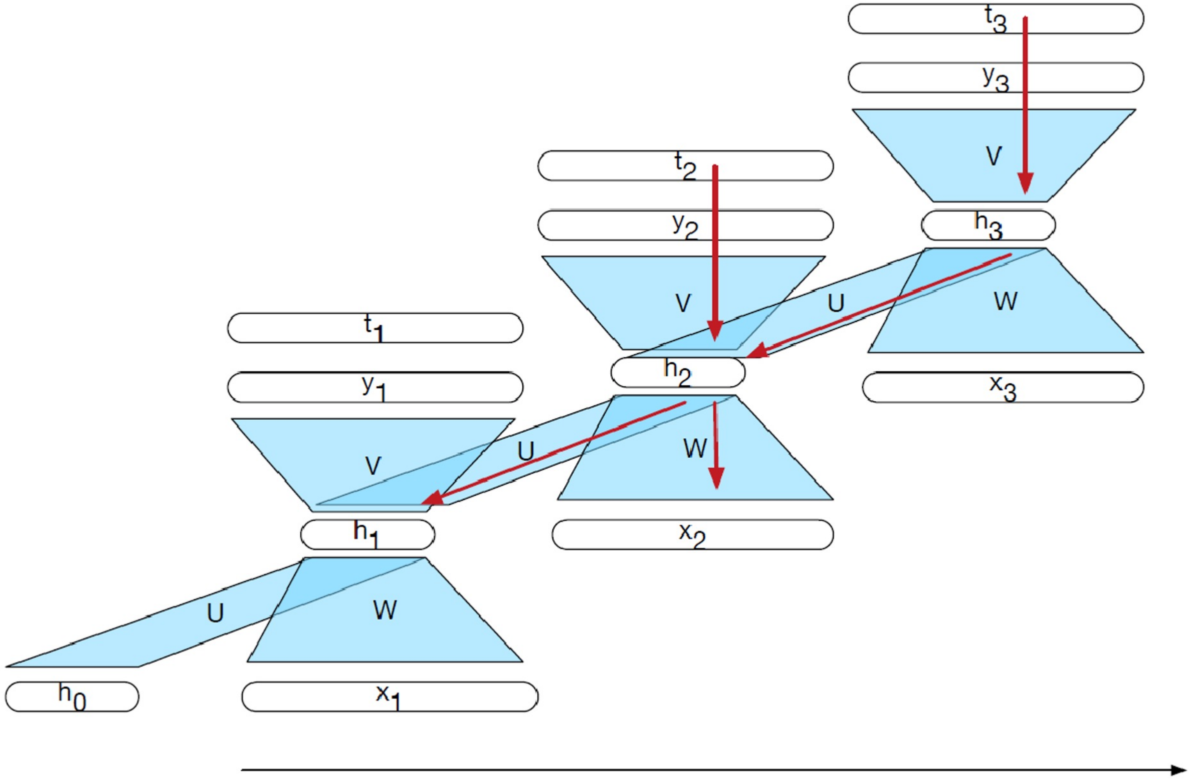
Training a Deep Neural Network



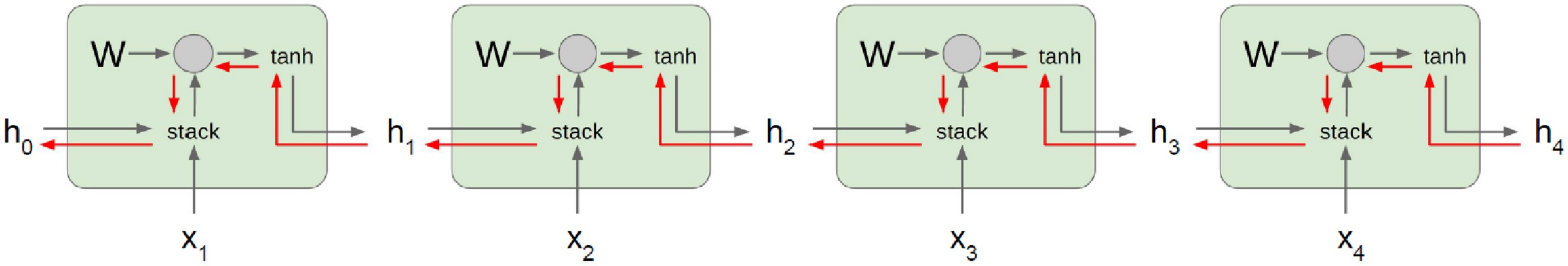
Gradient flow in a FFNN



Gradient flow in an RNN



Gradient flow in an RNN



Computing gradient of h0 involves many factors of W.

If gradients in deeper layers are **> 1**, they will get exponentially **bigger**.

“Exploding gradients”

If gradients in deeper layers are **< 1**, they will get exponentially **smaller**.

“Vanishing gradients”

Interim summary

- RNNs can be used to ________________________________ sequences of arbitrary length, thanks to a self-loop on the hidden layer

Interim summary

- RNNs can be used to ______________<u>classify</u>________________ sequences of arbitrary length, thanks to a self-loop on the hidden layer

Interim summary

- RNNs can be used to ___________classify, generate_____________ sequences of arbitrary length, thanks to a self-loop on the hidden layer

2 for the price of 1

Interim summary

- RNNs can be used to classify, generate, learn representations of sequences of arbitrary length, thanks to a self-loop on the hidden layer

3 for the price of 1

Interim summary

- RNNs can be used to classify, generate, learn representations of sequences of arbitrary length, thanks to a self-loop on the hidden layer
- Shared weights between time steps
- New initialization per sequence
- Can be “unrolled” and viewed as deep FFNN

- **Problem:** Vanishing and exploding gradients

3 for the price of 1

Quick Break! – Something to think about:

1. How are representations we get from RNNs different from EMBEDDINGS?

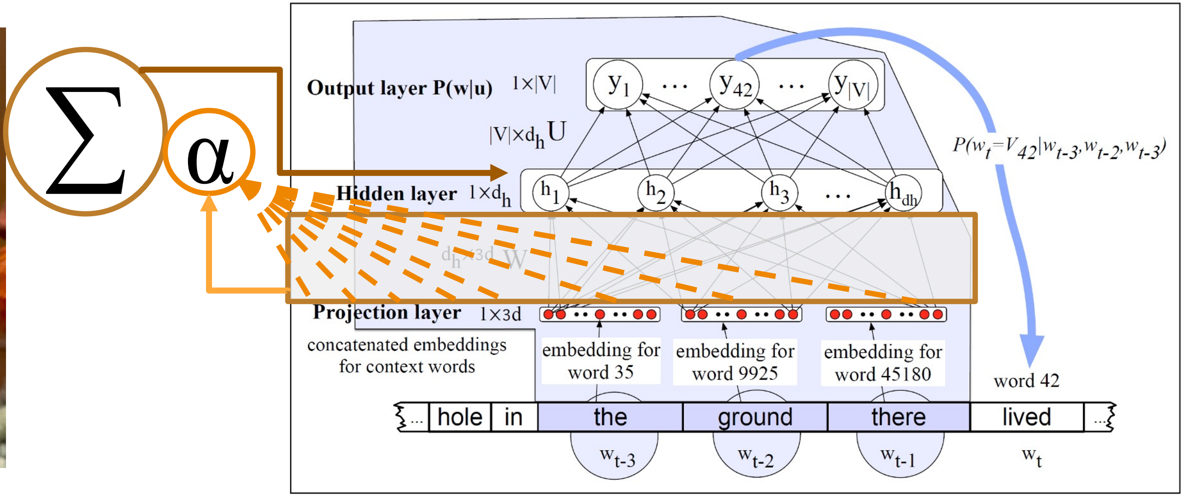
Contextualized

Global

2. Could we get a similar effect WITHOUT the recurrence?

Non-Recurrent Neural LM

- Non-neural approach: n-grams $\mathbf{p(w_t \mid w_{t-1}, w_{t-2}, w_{t-3})}$
- Neural approaches:
  - Convolutional neural net (CNN) with kernel size = n
  - Transformers: „Attention is all you need“ (BERT and friends)



Exploding and vanishing gradients

- Occur in FFNNs as well
  - Worse the more layers you have

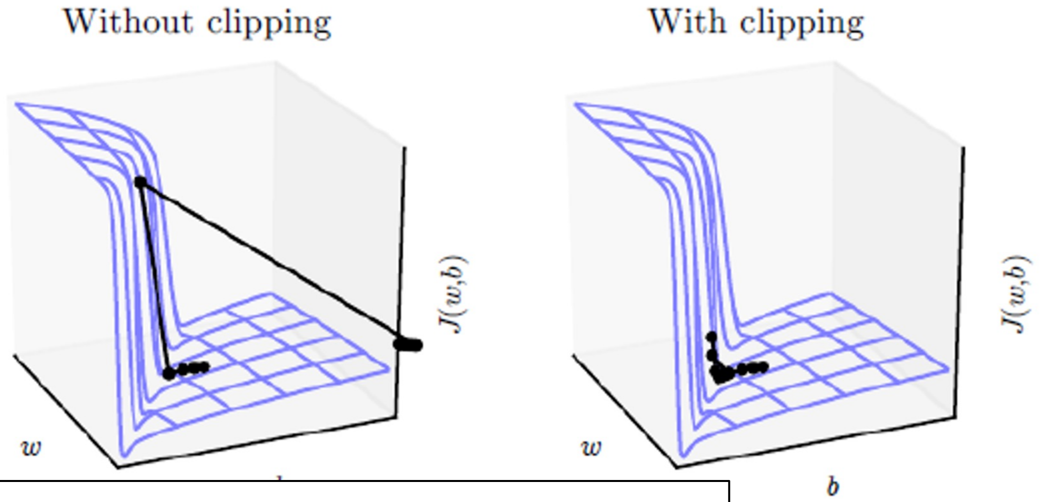
- Bigger problem with RNNs because network is **deep in the length of the sequence** and **deep in the number of layers**

Exploding and vanishing gradients

<u>Gradient clipping:</u>

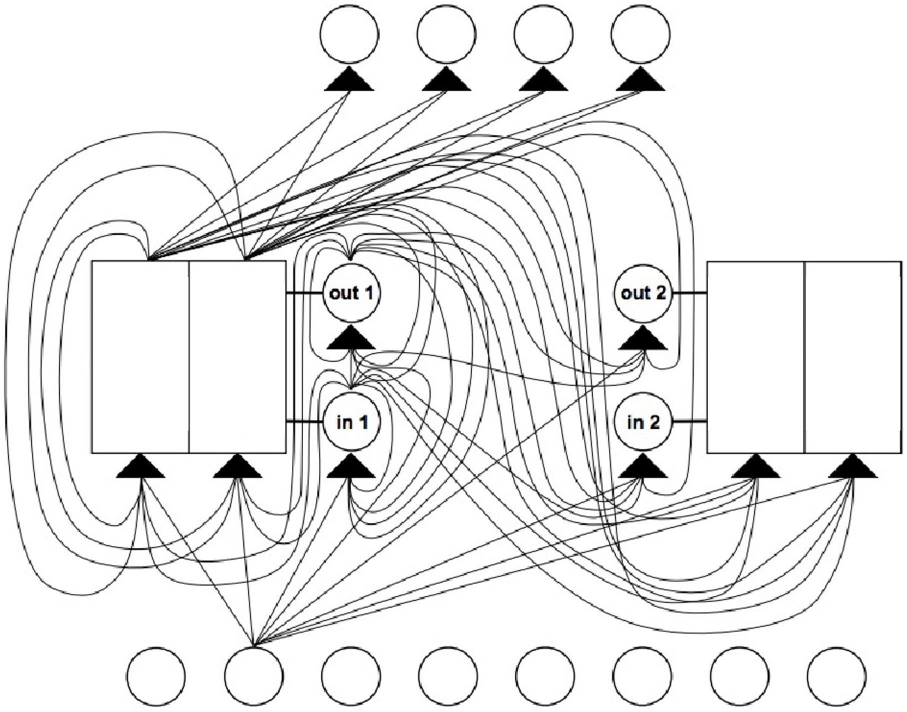
Scale gradient if its norm is too big

```
grad_norm = np.sum(grad * grad)
if grad_norm > threshold:
  grad *= (threshold / grad_norm)
```

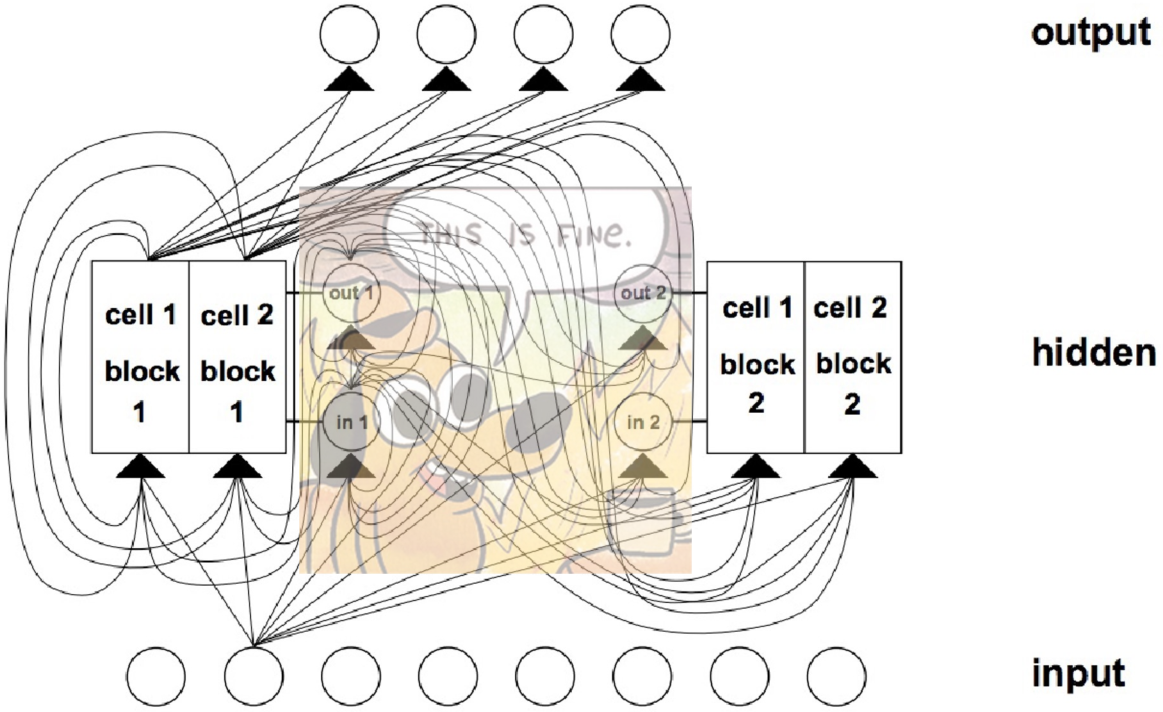


Dealing with vanishing gradients is more complicated.

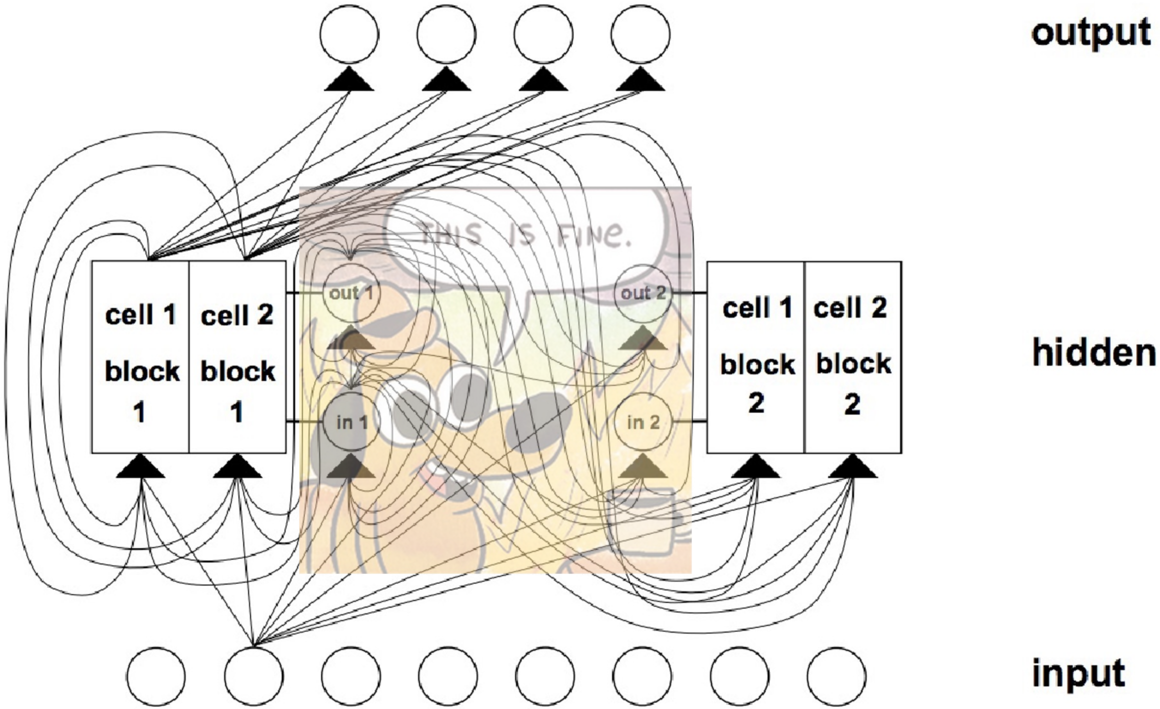
Ancient Treasure Map



Long Short-Term Memory (LSTM)



Ancient (!) Long Short-Term Memory (LSTM)



Hochreiter and Schmidhuber, **1997**, accessed through lecture slides by [Arnold]

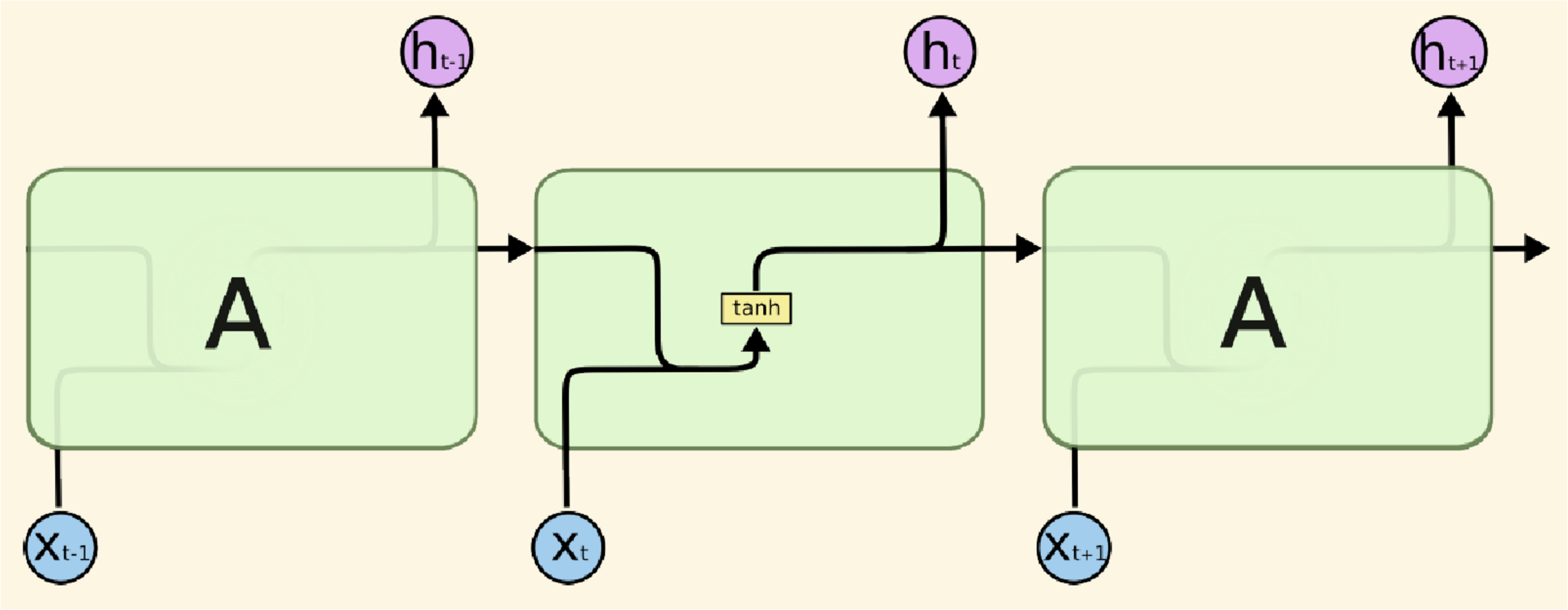
Long Short-Term Memory (LSTM)

Fundamentally, an LSTM is just an RNN with some additional machinery within each node of the network.

Long Short-Term Memory (LSTM)

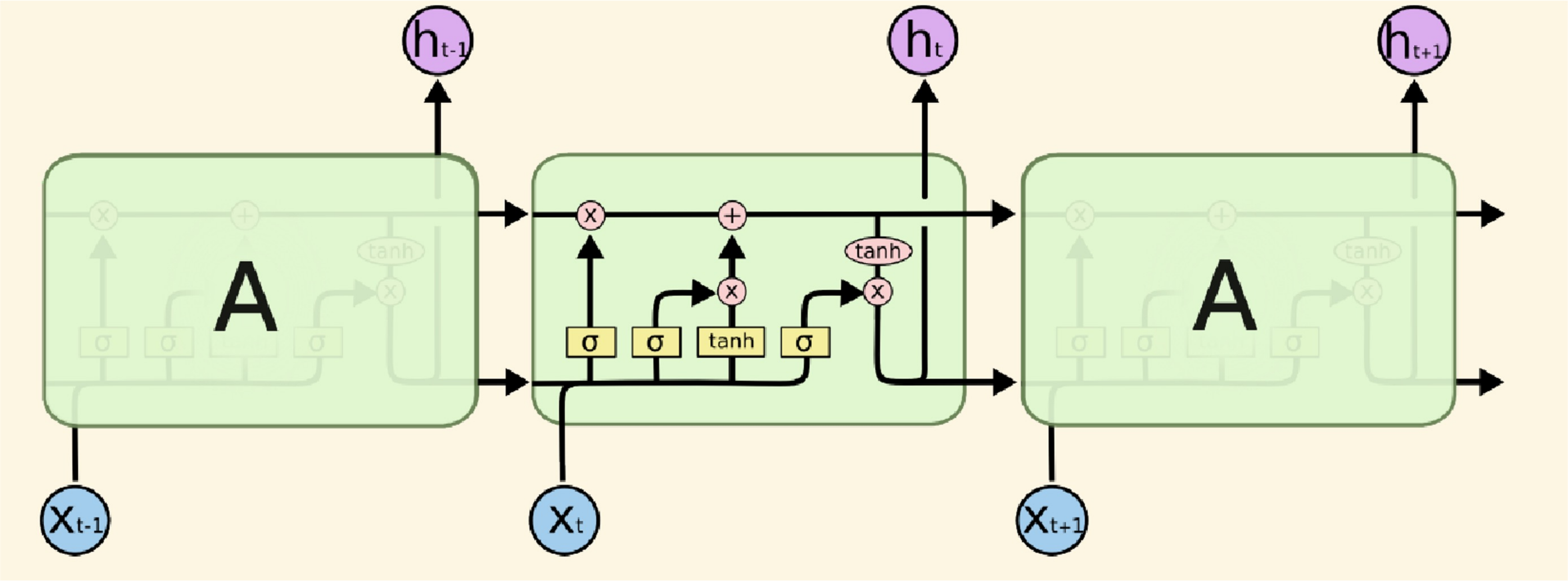
- New concept of **cell state**
- 3 **gates** that are just functions of the cell state and hidden state:
  - Forget
  - Input
  - Output

Long Short-Term Memory (LSTM)



Arnold

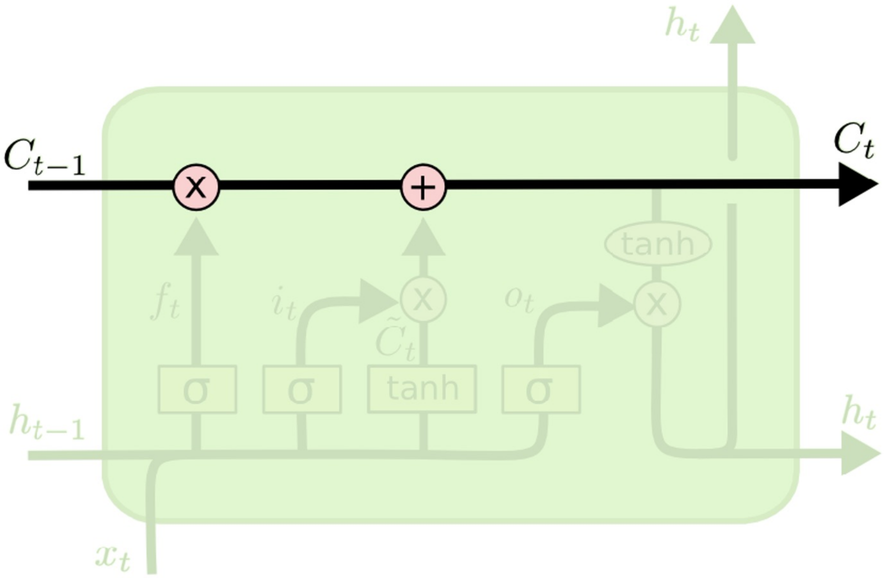
Long Short-Term Memory (LSTM)



Arnold

Long Short-Term Memory (LSTM)

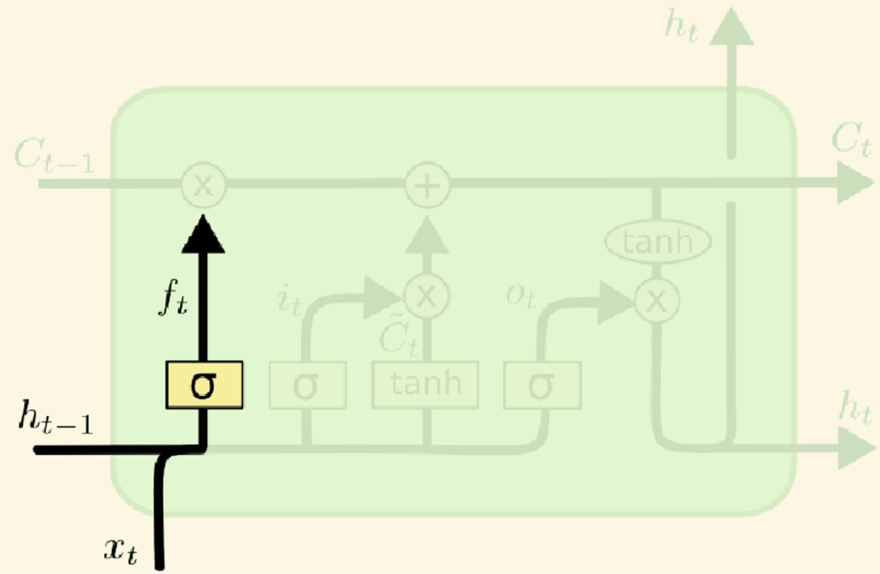
Arnold



Cell state / context memory is separated from cell output, and is only changed by two linear functions at each time step.

Long Short-Term Memory (LSTM)

Arnold

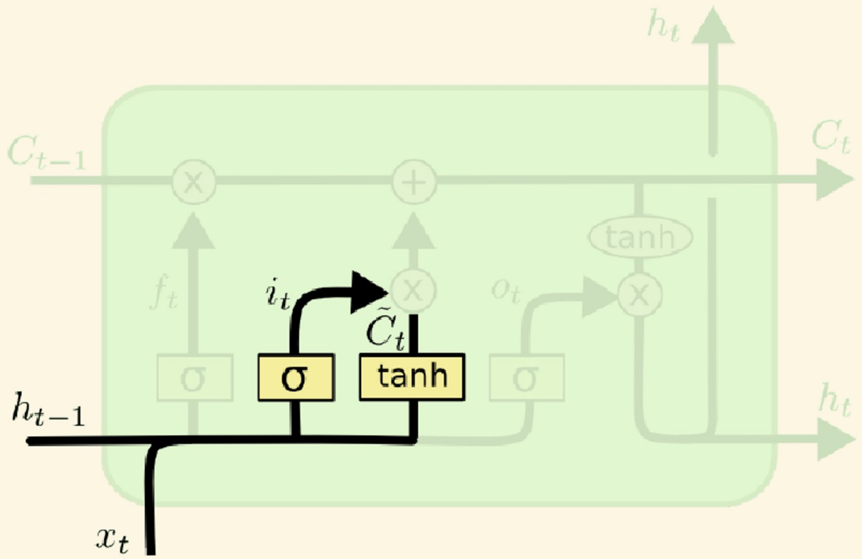


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Forget gate determines if previous context should be taken into account.

Long Short-Term Memory (LSTM)

Arnold



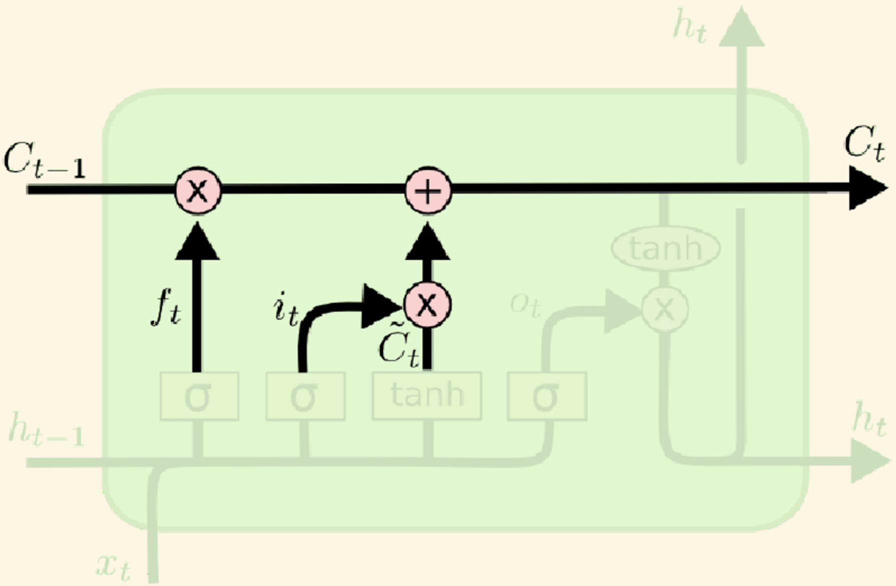
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Input gate determines if and how much of the current input should be taken into account.

Long Short-Term Memory (LSTM)

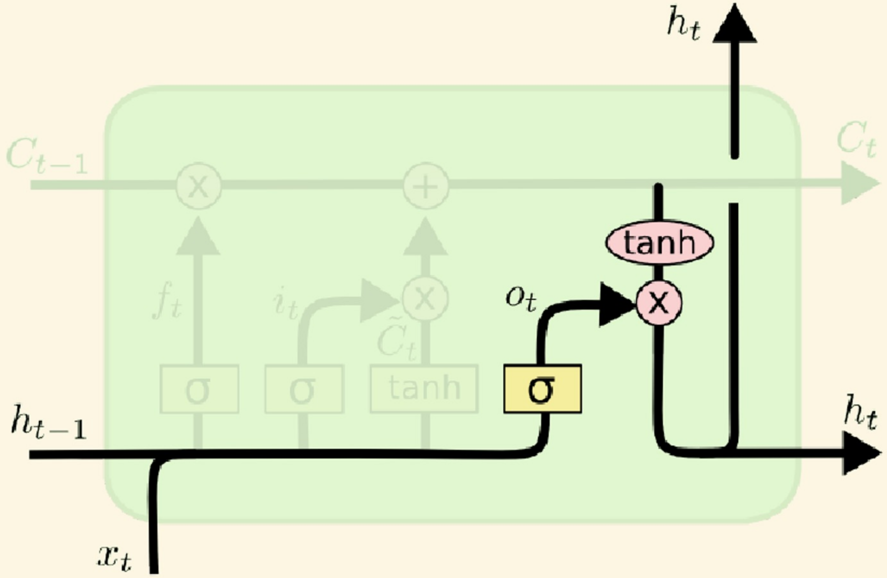
Arnold



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Cell state / context memory is now completely determined and can be calculated directly.

Long Short-Term Memory (LSTM)



Arnold

$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

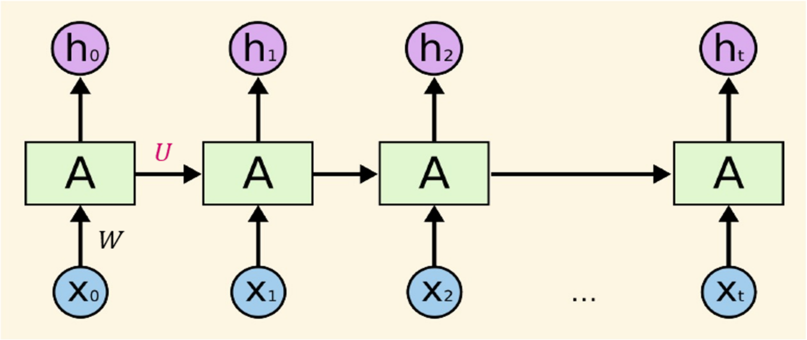
$$h_t = o_t * \tanh\left(C_t\right)$$

Output gate determines if and how much of the cell state should be yielded as output.

Long Short-Term Memory (LSTM)

Instead of weights **W** and **U** in an RNN,
we have:

- W_f
- W_i
- W_o



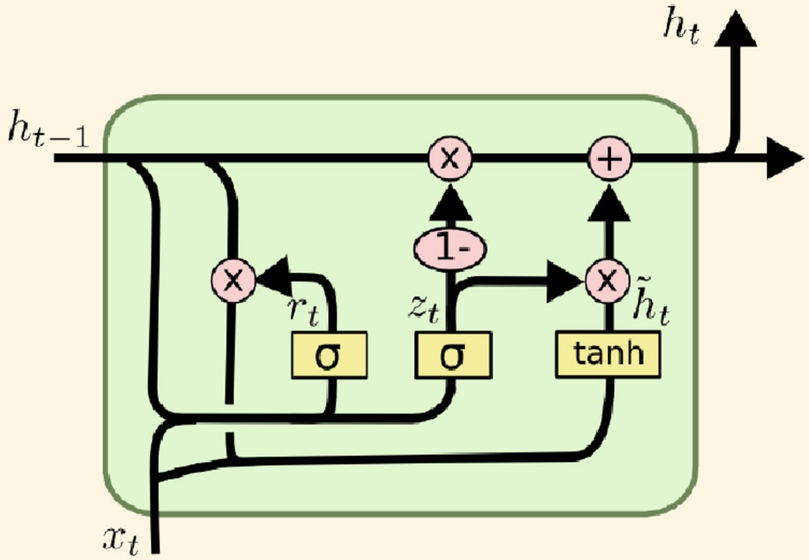
Long Short-Term Memory (LSTM)

- Separates cell state and output
- Vanishing gradient only occurs if all partial derivatives tend toward 0
  - Gradient of cell state is conveniently *additive* instead of *multiplicative*, reducing likelihood of all sub gradients being 0
- 3 **gates**:
  - Forget
  - Input
  - Output
- Infinite memory is regulated via these gates to better capture long-range dependencies

Arbel: "How LSTM networks solve the problem of vanishing gradients"

Gated Recurrent Unit (GRU)

- Simpler: combines forget and input gates
- Equally powerful: gates still take care of vanishing/exploding gradients



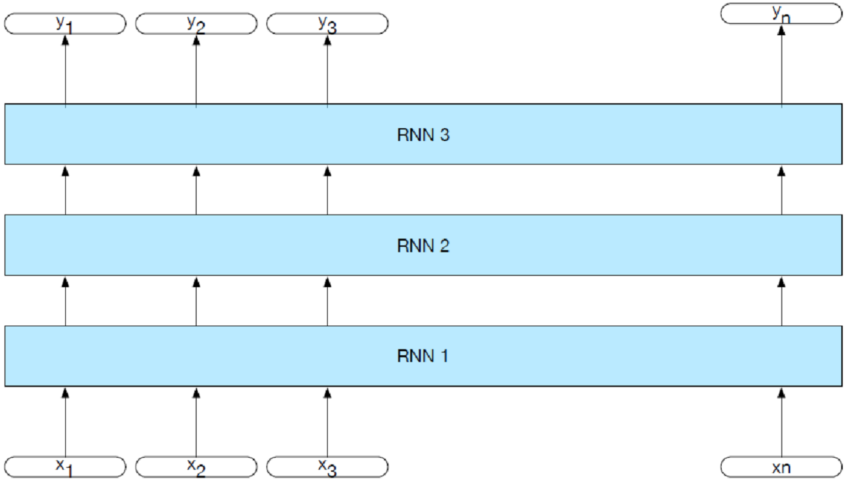
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

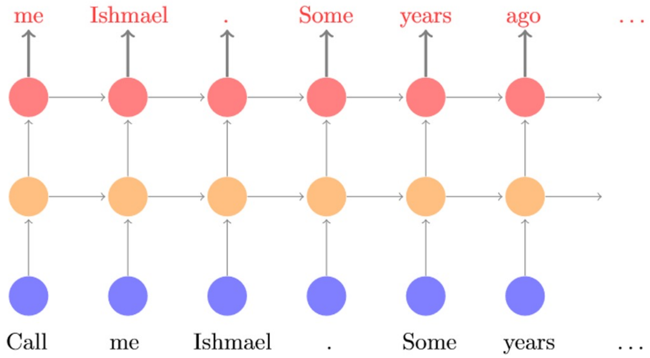
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Stacked RNN

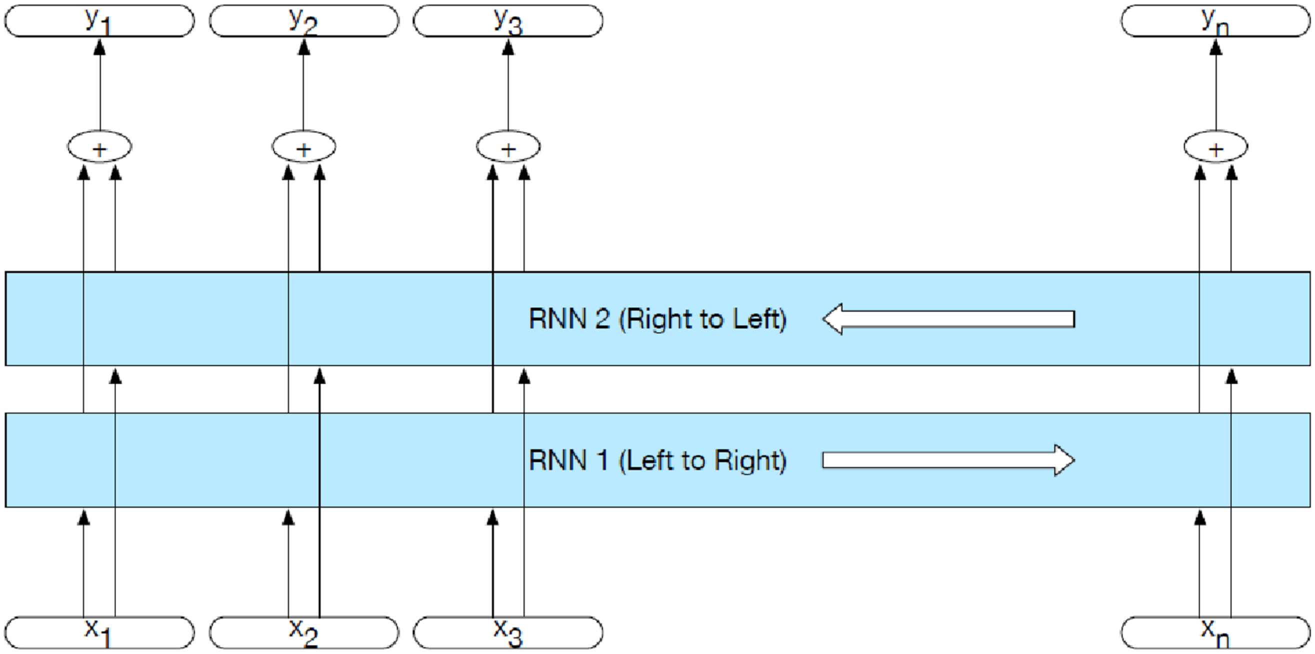


y_1
y_2
y_3
y_n
RNN 3
RNN 2
RNN 1
x_1
x_2
x_3
xn



me Ishmael . Some years ago . . .
Call me Ishmael . Some years . . .

Bidirectional RNN



Jurafsky & Martin

Li, Johnson, Yeung

RNNs are powerful!

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

train more

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

train more

```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

train more

```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

RNNs are powerful!

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```

```
VIOLA:
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.
```

[Li, Johnson, Yeung]

RNNs are powerful!

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

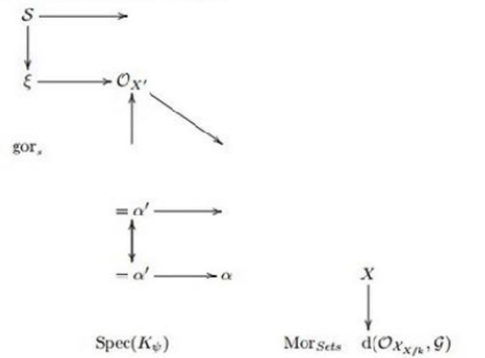
be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

1. $\mathcal{F}$ is an algebraic space over S.
2. If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram



is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.

A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \;\; \text{-}1(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_\ell}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{v}})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.

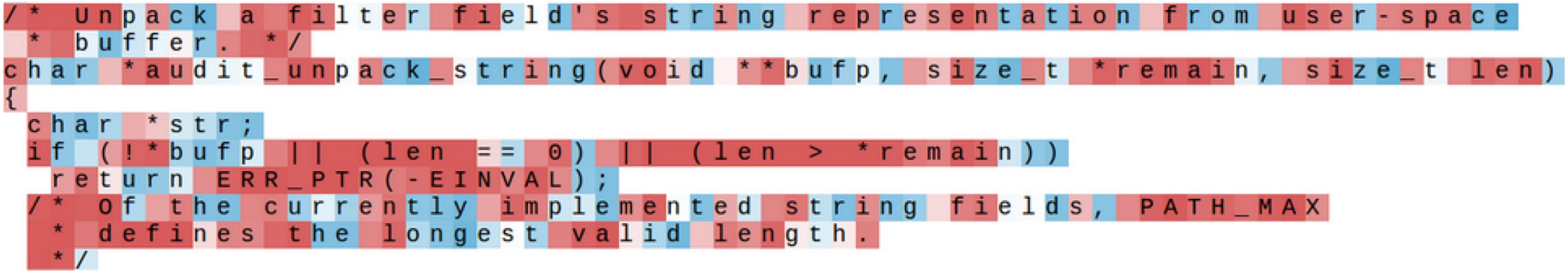
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.

If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

[Li, Johnson, Yeung]

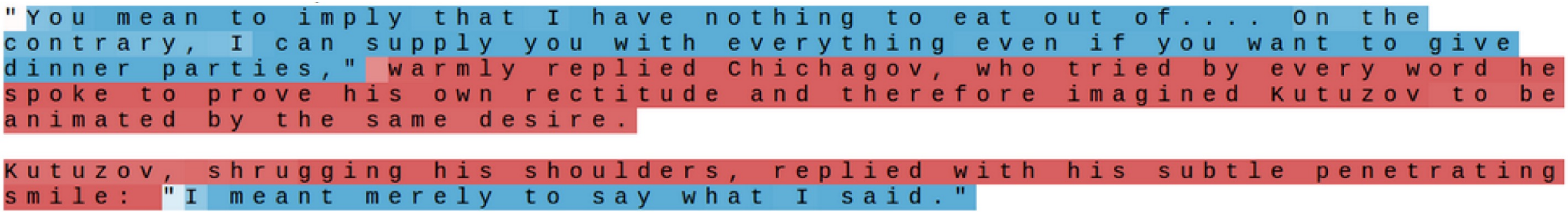
RNNs are hard to interpret!



Pick a neuron in the hidden representation and trace when it “fires”.

Li, Johnson, Yeung

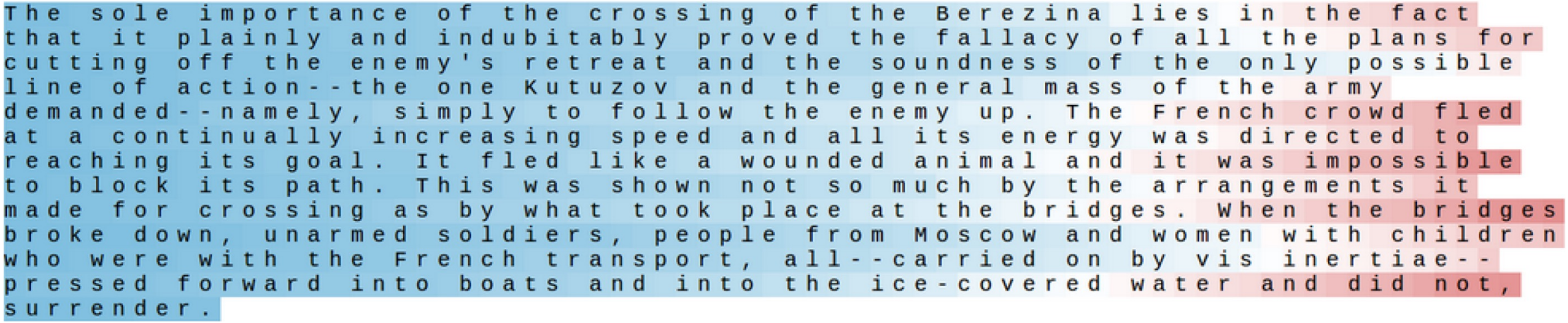
RNNs are hard to interpret!



Quote detection cell

[Li, Johnson, Yeung]()

RNNs are hard to interpret!



Line length tracking cell

RNNs are hard to interpret!

- **Probing** tasks to discern whether these models implicitly learn different aspects of syntax, semantics
- E.g., Linzen et al. (2018), Linzen et al. (2016) , Ettinger et al. (2016), Ettinger and Linzen (2016)

- Enjoy with caution!

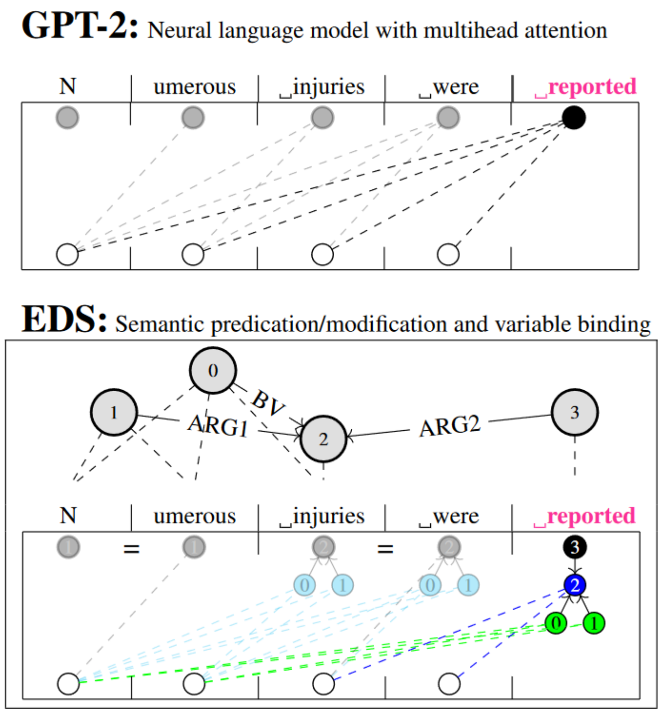
Summary

- Feed-forward NNs are very powerful, but not well-suited for language data
- Recurrent NNs model variable-width data as **streaming in over time**
- Many different applications in NLP (and elsewhere, e.g., Bioinformatics)

- LSTMs and GRUs:
    - Address the problem of exploding and vanishing gradients
    - Better at capturing long-range dependencies
- These days more and more replaced by Transformers

- Drawbacks (of **all** sequential NNs):
    - More parameters to train
    - Usually require large amounts of training data
    - Tricky to interpret what exactly is learned

Linguistic Structure beyond the Sequence



[Prange, Schneider, Kong: Ongoing Work.]

- Scaffolding sequential NNs with linguistically-motivated hierarchical structure
- → **Improve LM quality!**

- Better explainability (?)
- Less training data required (?)
- Fewer parameters (?)

?