Natural Language and Speech Processing

Lecture 10: Recurrent Neural Networks for Modeling Language

Tanel Alumäe

NLP and Sequential Data

- NLP is full of sequential data
  - Words in sentences
  - Characters is words
  - Sentences in document
  - ...

Long-distance dependencies

- Agreement in number, gender, etc
  - *__He__ does not have very much confidence in __himself__.*
  - *__She__ does not have very much confidence in __herself__.*
- Selectional preference
  - *The __reign__ has lasted as long as the life of the __queen__.*
  - *The __rain__ has lasted as long as the life of the __clouds__.*

Long-distance dependencies, II

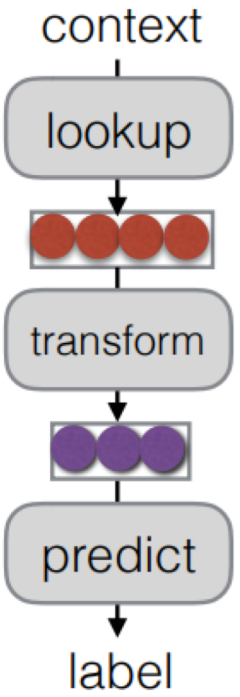
- What does „*it*" refer to?
  - The trophy would not fit in the brown suitcase because it was too **big**.

    Trophy
  - The trophy would not fit in the brown suitcase because it was too **small**.

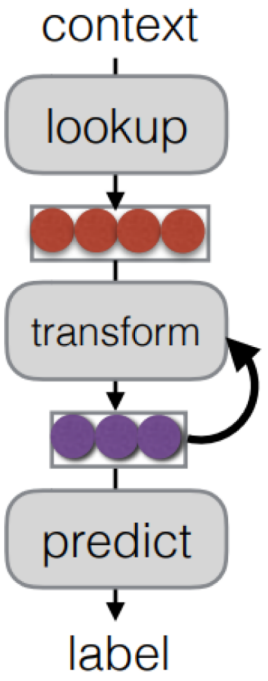
    Suitcase

Recurrent neural networks

- Feed-forward neural network

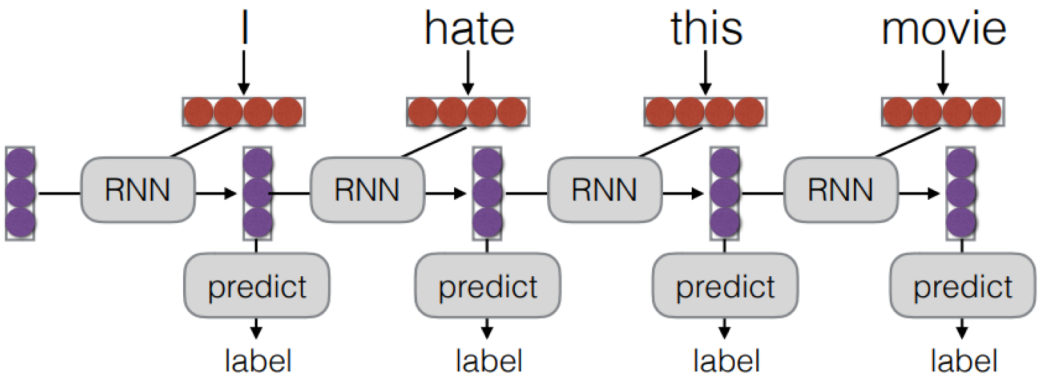


- Recurrent neural network



Unrolling a neural network in time

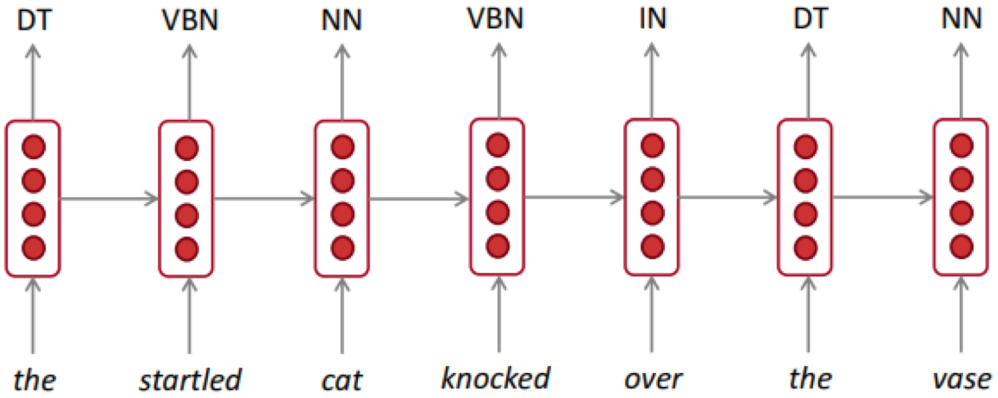
- Processing a sequence (e.g. a sentence) with a neural network
- The current output of RNN depends on the hidden state at the prior state (a **vector**) and current input (another **vector**)
- In NLP t ne current (ne current input word and the hidden layer output of the previous word



- This is the reason RNNs are considered to

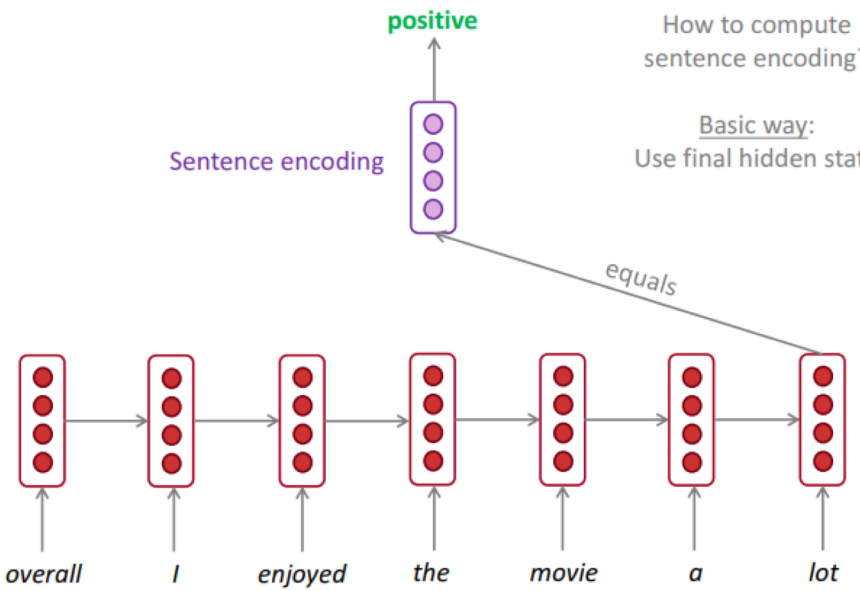
RNNs for word labeling

- RNNs are very flexible and can be customized for different tasks
- Below: RNN for word labeling
  - i.e., classify every element of a sequence



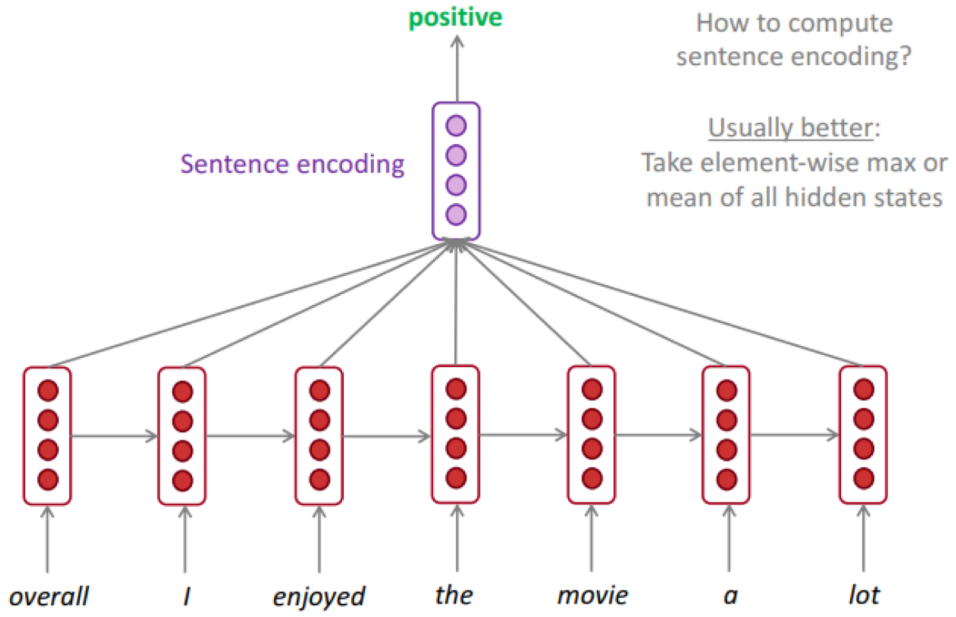
RNN for sequence labeling

- RNNs can be used for labeling the whole sequence
  - E.g., for classifying sentences/documents
- Simplest: just use the final hidden state for classifi
  - The RNN ate the most impc ugh the sentence



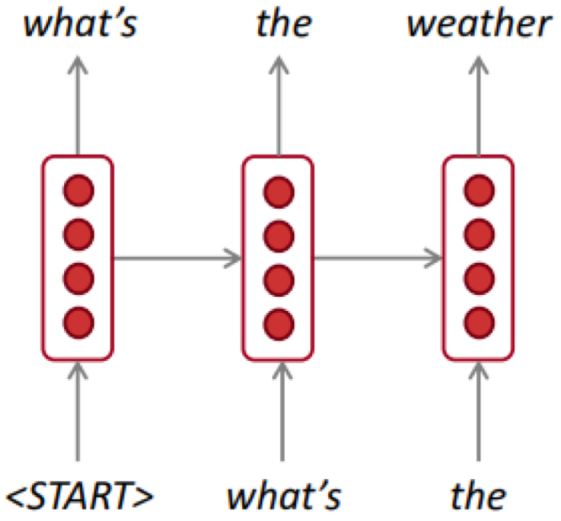
RNN for sequence labeling

- Alternative: average or max the hidden states over the whole sequence
- Even better: attention mechanism – learn a function that weights the hidden states appropriately (e.g., downweight the functi



RNN for generating text

- RNN can be used for generaing text
  - Start with some random initial state, use a pseudo-word <START> as the initial input
  - Generate a word (or character)
- **Sample** it f t distribution (**sample** = word, with the given probal
  - Feed it back RNN, and generate an



RNN text generation samples I

.Shakespeare:

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.

Wikipedia:

Naturalism and decision for the majority of Arab countries' capitalide was grounded by the Irish language by [[John Clair]]. [[An Imperial Japanese Revolt]], associated with Guangzham's sovereignty. His generals were the powerful ruler of the Portugal in the [[Protestant Immineners]], which could be

RNN generated text, II

- PDF compiled from generated LaTeX source (with some minor syntactic r

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the compariccoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, **??** and the fact that any U affine, see Morphisms, Lemma **??**. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma **??** we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example **??**. It may replace S by $X_{spaces,étale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma **??**. Namely, by Lemma **??** we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ *(by the formal open covering* X *and a single map* $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ *over* U *compatible with the complex*

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ *is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition* **??** *(without element is when the closed subschemes are catenary. If* T *is surjective we may assume that* T *is connected with residue fields of* S*. Moreover there exists a closed subspace* $Z \subset X$ *of* X *where* U *in* X' *is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem*

(1) f *is locally of finite type. Since* $S = \mathrm{Spec}(R)$ *and* $Y = \mathrm{Spec}(R)$.

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{\mathcal{X},\ldots,0}$.

Lemma 0.2. *Let* X *be a locally Noetherian scheme over* S, $E = \mathcal{F}_{X/S}$. *Set* $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. *Since* $\mathcal{I}^n \subset \mathcal{I}^n$ *are nonzero over* $i_0 \leq \mathfrak{p}$ *is a subset of* $\mathcal{J}_{n,0} \circ \overline{A}_2$ *works.*

Lemma 0.3. *In Situation* **??**. *Hence we may assume* $\mathfrak{q}' = 0$.

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (**??**). On the other hand, by Lemma **??** we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

RNN generated text, III

- Generated baby names, based on 8000 actual names (only generated names not in the training data are listed): *Rudi Levette Berice Lussa Hany Mareanne Chrestina Carissy Marylen Hammine Janye Marlise Jacacrie Hendred Romand Charienna Nenotto Ette Dorane Wallen Marly Darine Salina Elvyn Ersia Maralena Minoria Ellia Charmin Antley Nerille Chelon Walmor Evena Jeryly Stachon Charisa Allisa Anatha Cathanie Geetra Alexie Jerin Cassen Herbett Cossie Velen Daurenge Robester Shermond Terisa Licia Roselen Ferine Jayn Lusine Charyanne Sales Sanny Resa*

RNN for conditional text generation

- As before, but the initial state encodes some input (e.g. an image)
  - E.g., use a convolutional neural network to reduce the image to a fixed length feature vector
- Genera
- Can be tioning
- Applica ple to underst

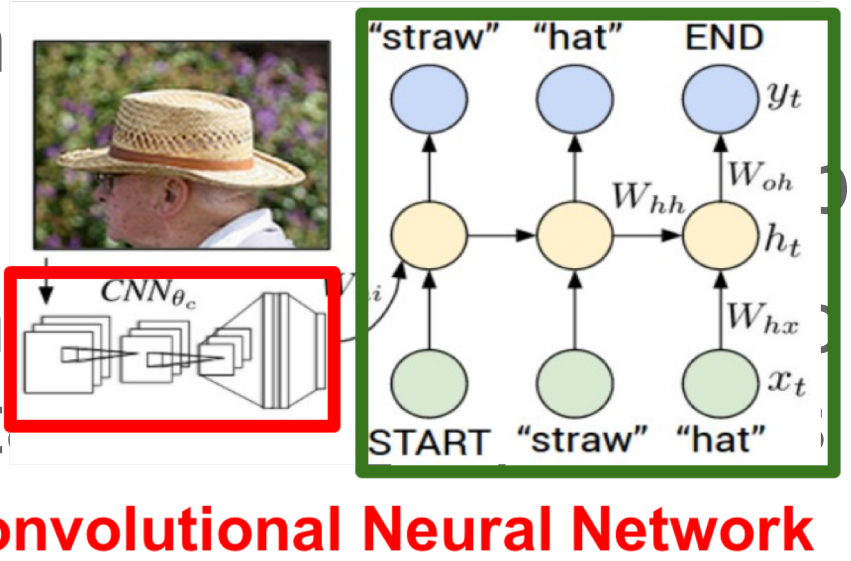


Image captioning examples

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."

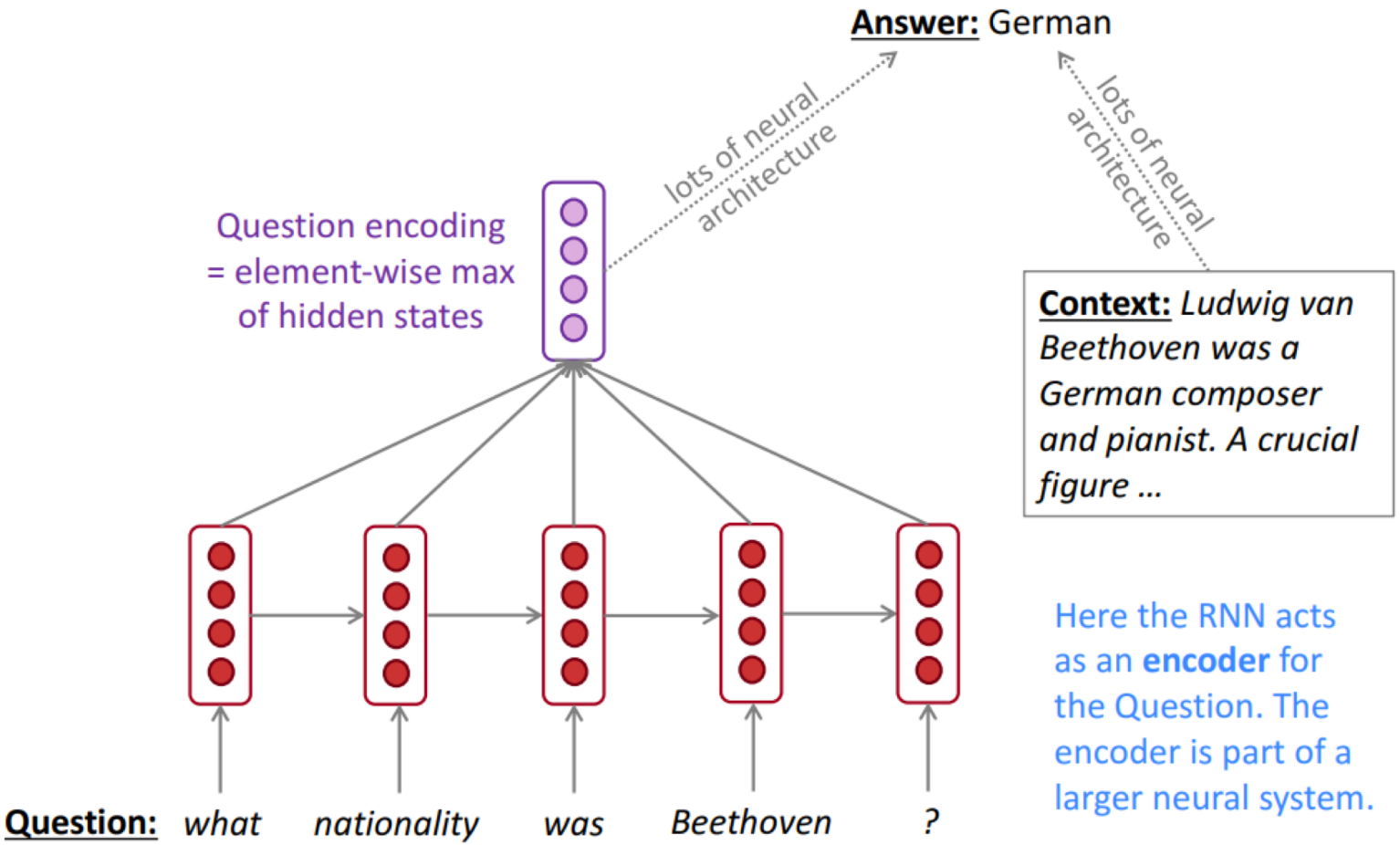
"a young boy is holding a baseball bat."

"a cat is sitting on a couch with a remote control."

"a woman holding a teddy bear in front of a mirror."

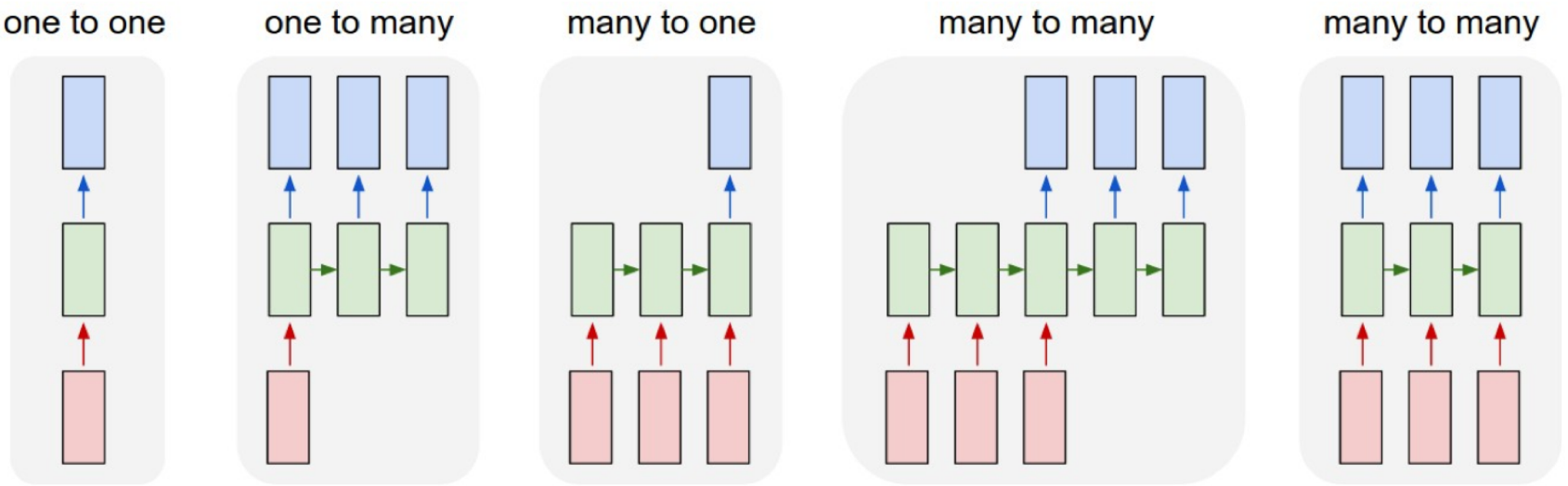
"a horse is standing in the middle of a road."

RNN as encoder module



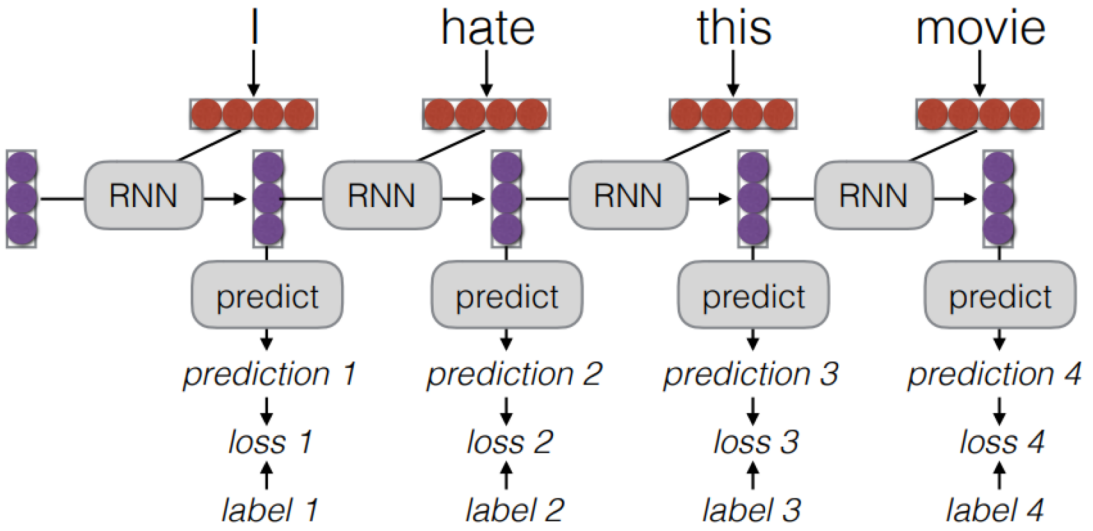
Summary of neural network architectures

- One-to-one: feed-forward DNN we have studied previously (e.g. document classification with CNN, named entity recognition)
- One-to-many: image captioning
- M
  us
- M
- M
  using RNN



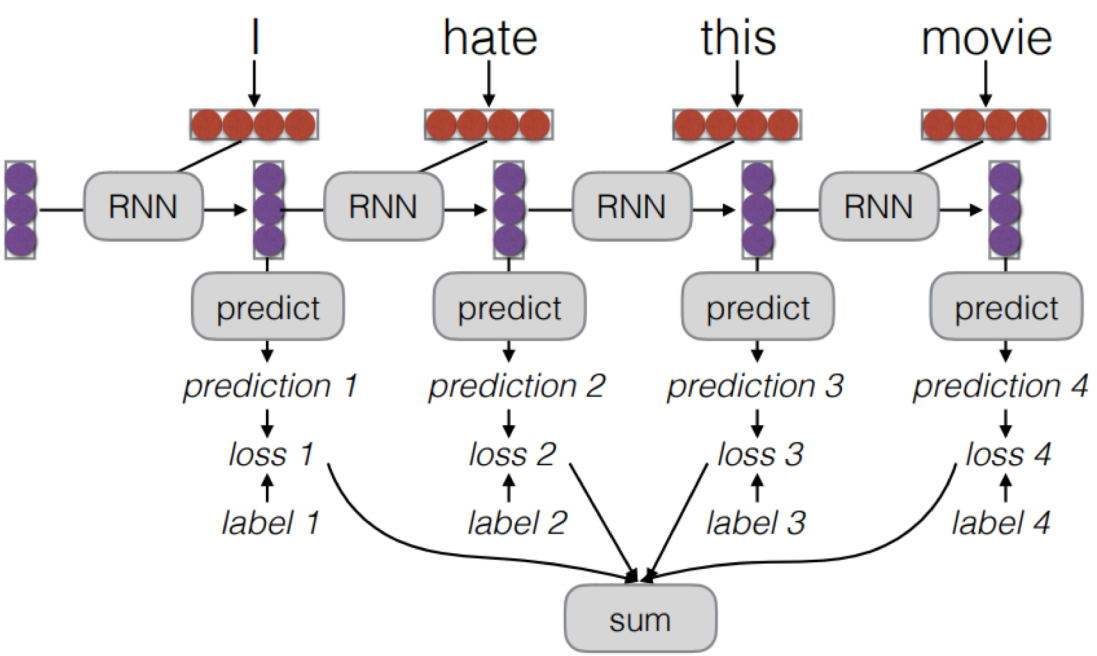
Training RNN

- Training RNN can be done using backpropagation, as with feed-forward NNs
- As before, we (or our toolkit) need to have a c[illegible]nting the RNN
- Training [illegible]sing an **unro**



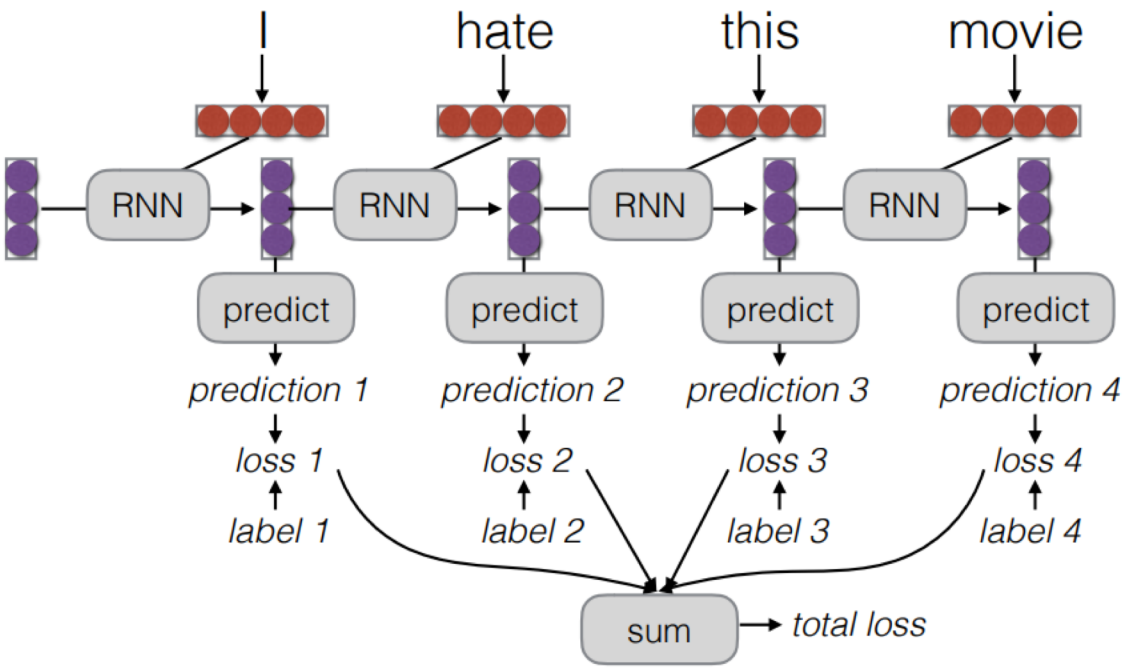
Training RNN

- Training RNN can be done using backpropagation, as with feed-forward NNs
- Ad before, we (or our toolkit) need to have a c senting the RNN
- Training using an **unro**



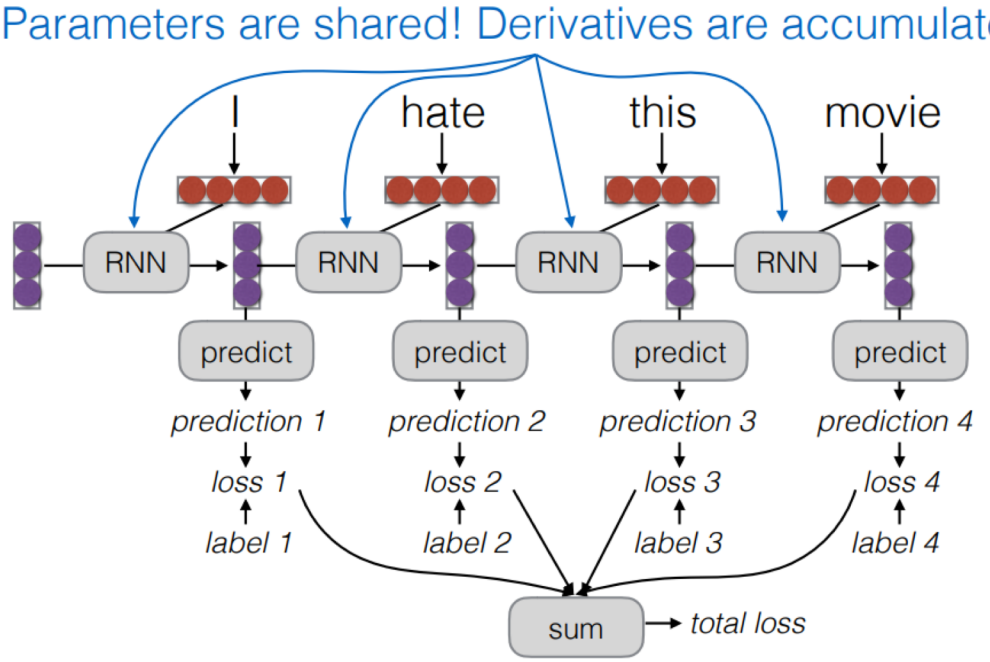
Training RNN

- Training RNN can be done using backpropagation, as with feed-forward NNs
- Ad before, we (or our toolkit) need to have a c enting the RNN
- Training using an **unro**



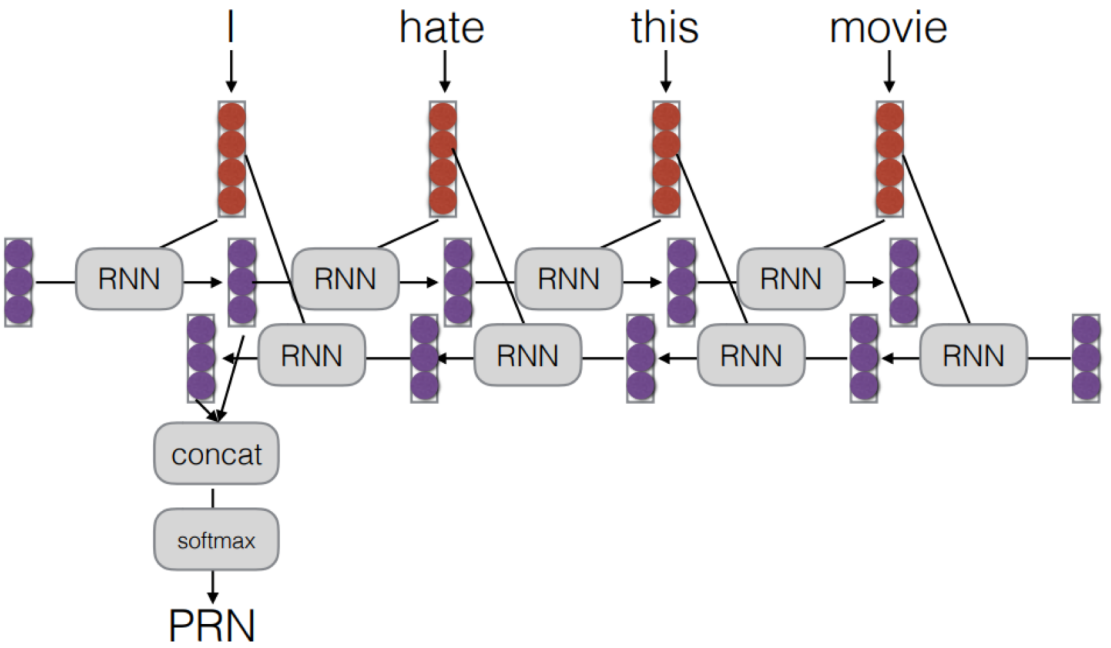
Training RNN

- Training RNN can be done using backpropagation, as with feed-forward NNs
- Ad before, we (or our toolkit) need to have a c esenting the RNN s
- Training d using an **unrc**



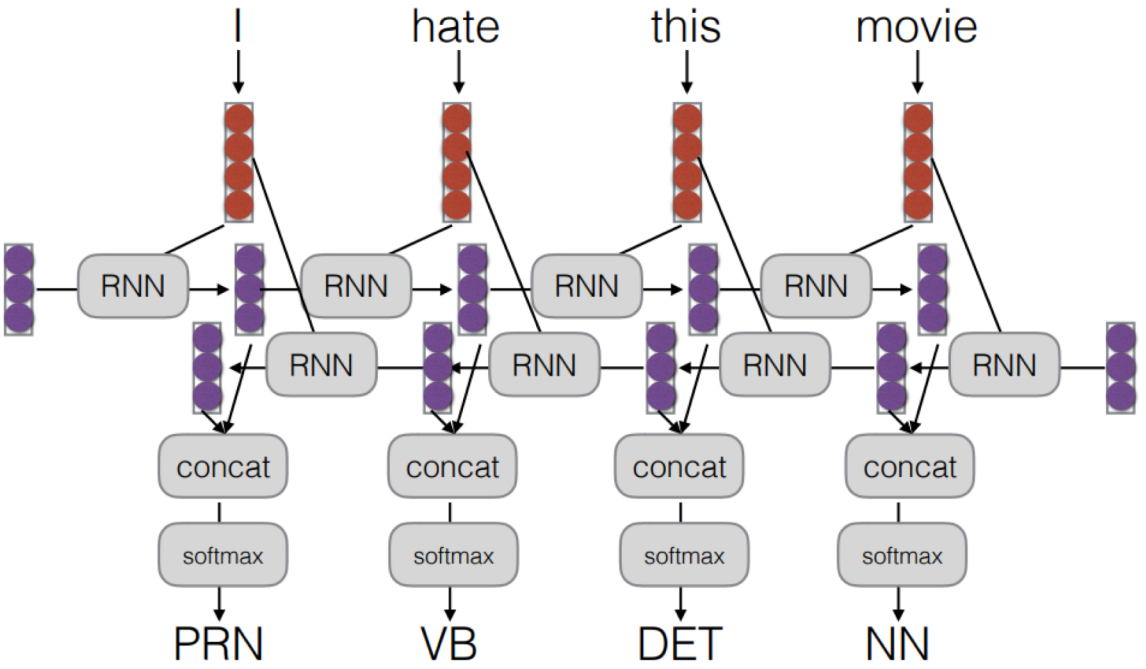
Bidirectional RNNs

- Simple extension: have two RNNs, running in both directions
- The hidden states from the two RNNs are simply concatenated



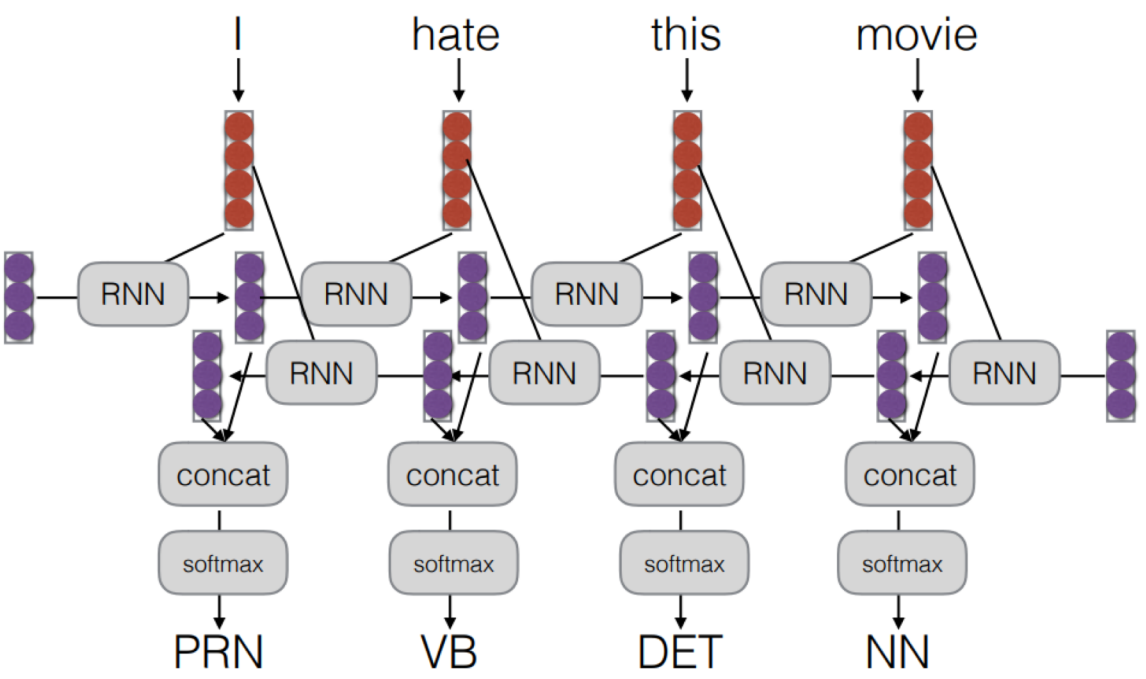
Bidirectional RNNs

- Simple extension: have two RNNs, running in both directions
- The hidden states from the two RNNs are simply concatenated



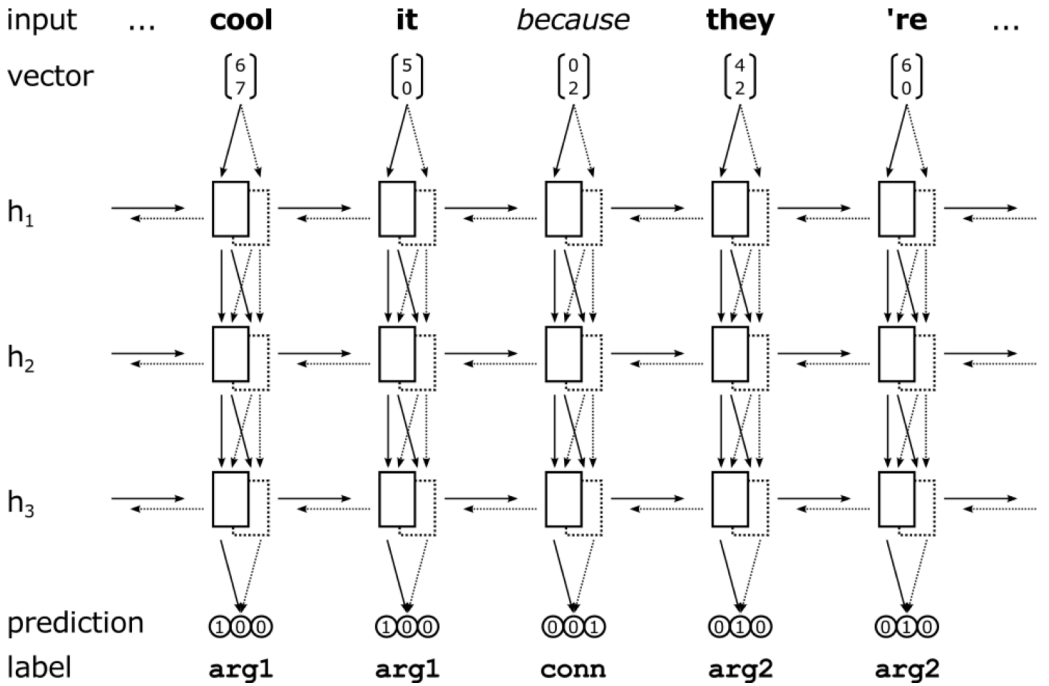
Bidirectional RNNs

- Why?
  - Word classification: enables to use information from both past and future words
  - Alternat dict the label consumi



Deep RNNs

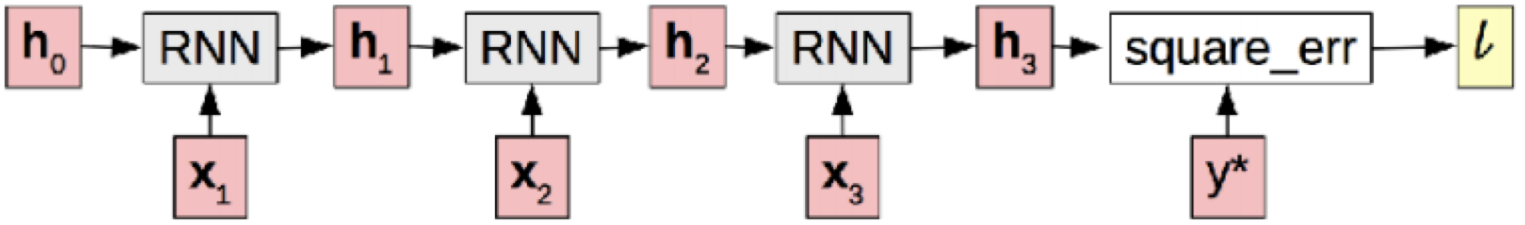
- We can stack many recurrent layers
- Each subsequent layer uses the output of the previous recurrent layer as input



Vanishing Gradients

- Result: the RNN doesn't learn to use information that is **many time steps in the past**
- Or, (in the exploding case), the parameters of the RNN get quickly very la
n

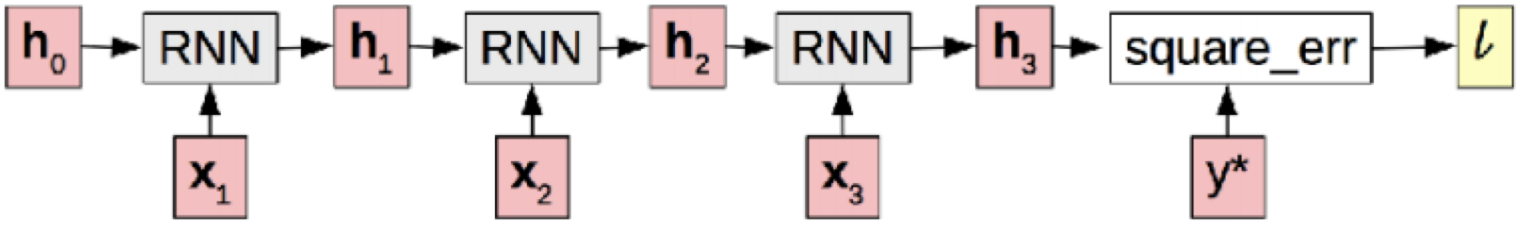
$$\frac{dl}{d_{h_0}} = \text{tiny} \quad \frac{dl}{d_{h_1}} = \text{small} \quad \frac{dl}{d_{h_2}} = \text{med.} \quad \frac{dl}{d_{h_3}} = \text{large}$$



Vanishing Gradients

- Gradients decrease (vanish) or increase (explode) when they get pushed back through the RNN
- They get „squashed” through non-linearities and small (or large) weight m

$$\frac{dl}{d_{h_0}} = \text{tiny} \qquad \frac{dl}{d_{h_1}} = \text{small} \qquad \frac{dl}{d_{h_2}} = \text{med.} \qquad \frac{dl}{d_{h_3}} = \text{large}$$

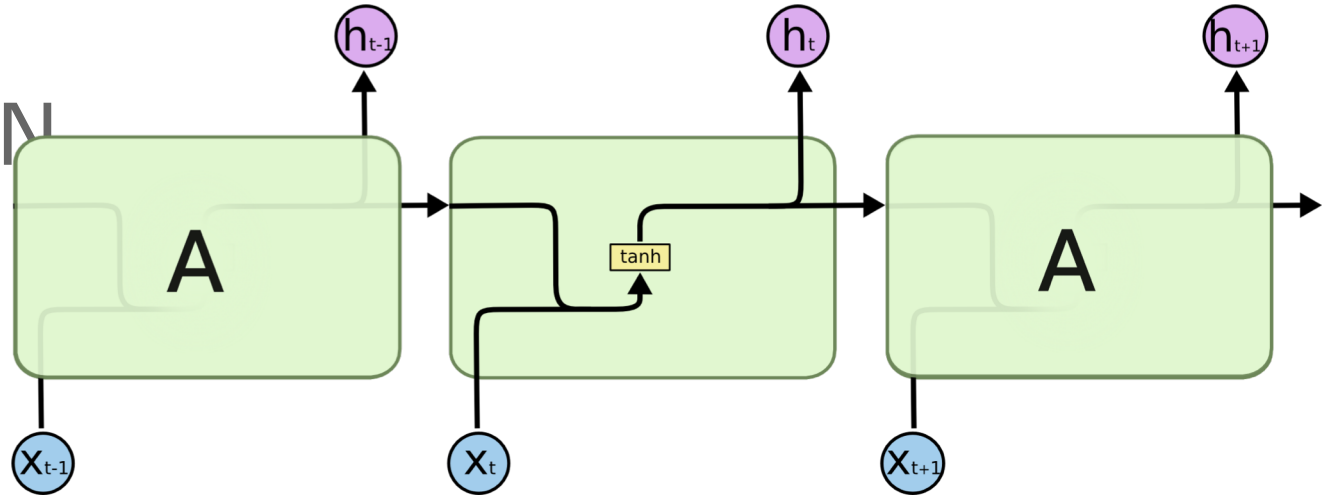


Solution to vanishing gradients: gated units

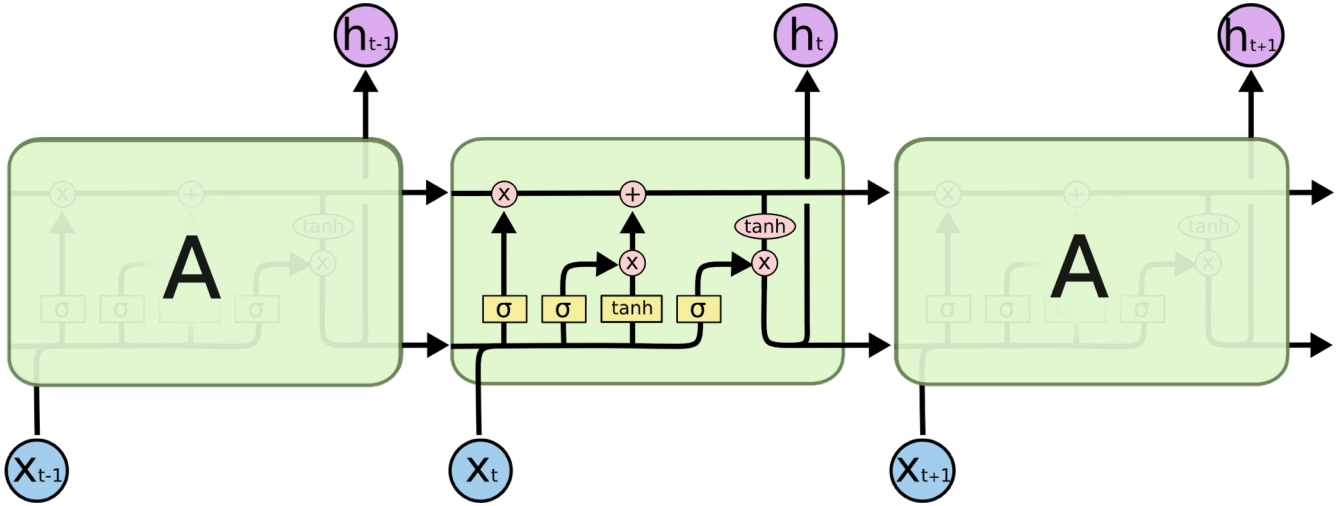
- The main solution to the vanishing gradient problem is to use a more complex hidden unit in the recurrent cell
- Gated Recurrent Unit (**GRU**) and Long Short-Term Memory (**LSTM**)
- Main ideas:
  - Make additive connections between time steps (not multiplicative)
- Addition does not modify the gradient -- no vanishing

LSTM

- Standard RNN

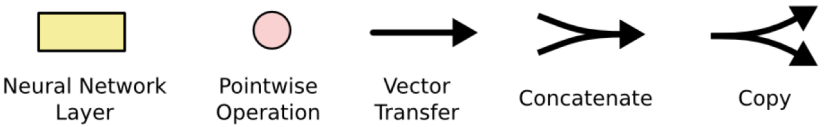


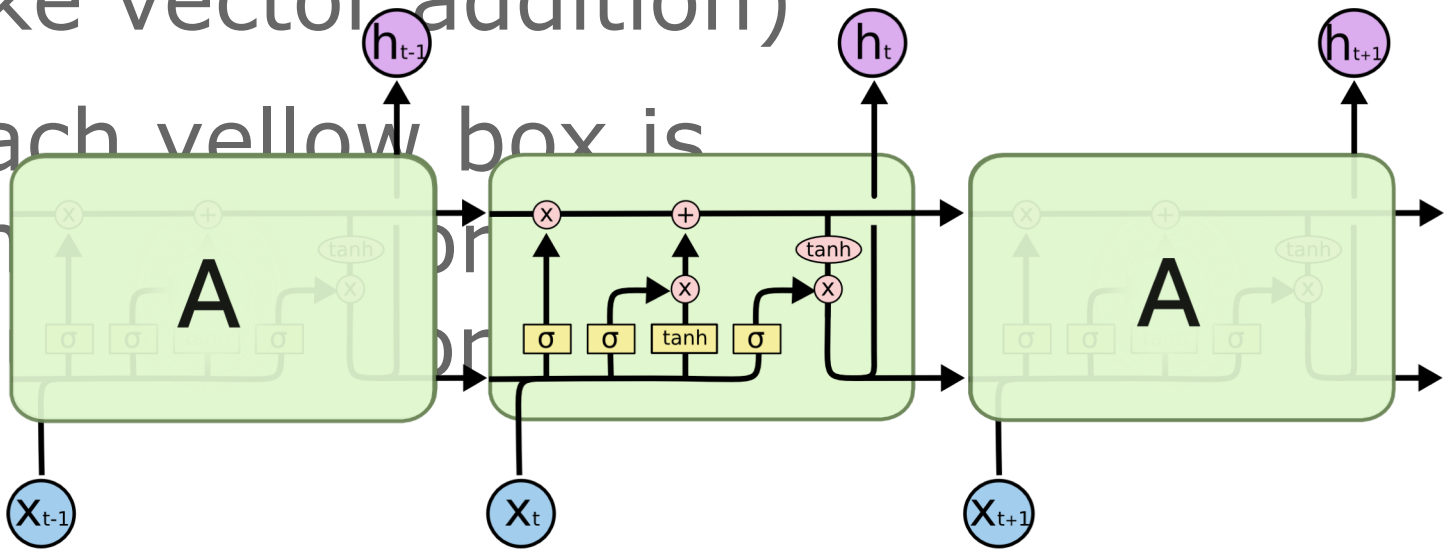
- LSTM



LSTM

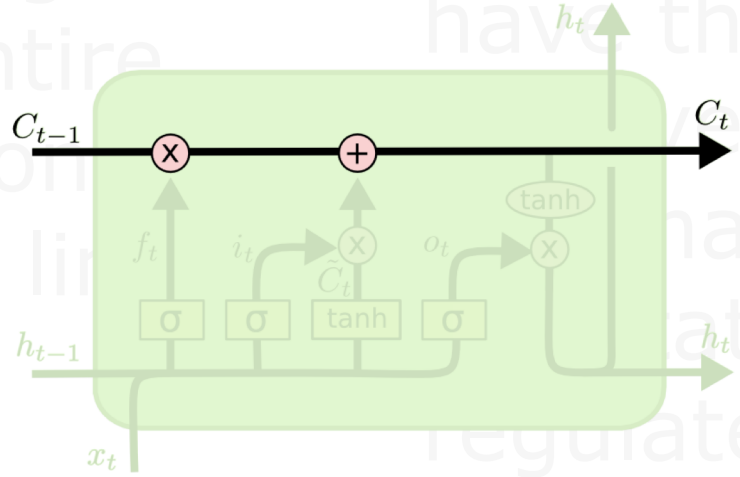
- Each line carries a vector
- Pink circle is a pointwise operation (like vector addition)
- Each yellow box is sm
ne



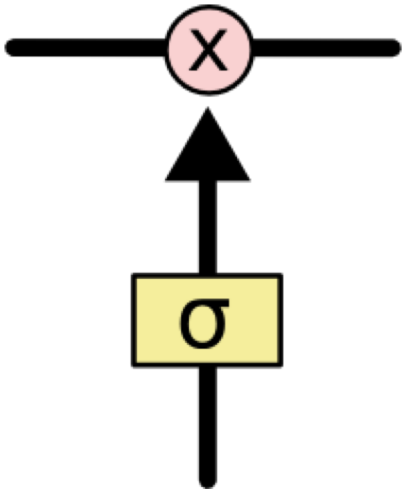


LSTM walkthrough

- Each LSTM cell has a **state**
- Cell state is kind of like a conveyor belt
- It runs straight down the entire chain, with only some minor interactions
- It's very easy for information to just flow along it unchanged
- The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called



LSTM walkthrough: gates

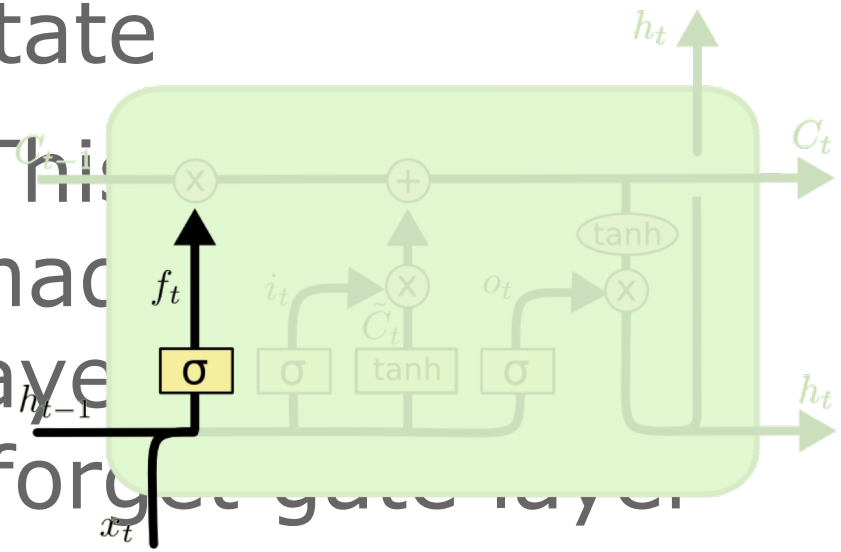


- Gates are a way to optionally let information through
- They are composed out of a sigmoid neural net layer and a pointwise multiplication through operation
- The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through
- A value of zero means “let nothing through,” while a value of one means “let everything through!”

LSTM walkthrough: forget gate

.The first step in our LSTM is to decide what information we're going to throw away from the cell state

.This is called the "forget gate layer"



.It looks at h_{t-1} and x_t, and outputs a number between 0 and 1

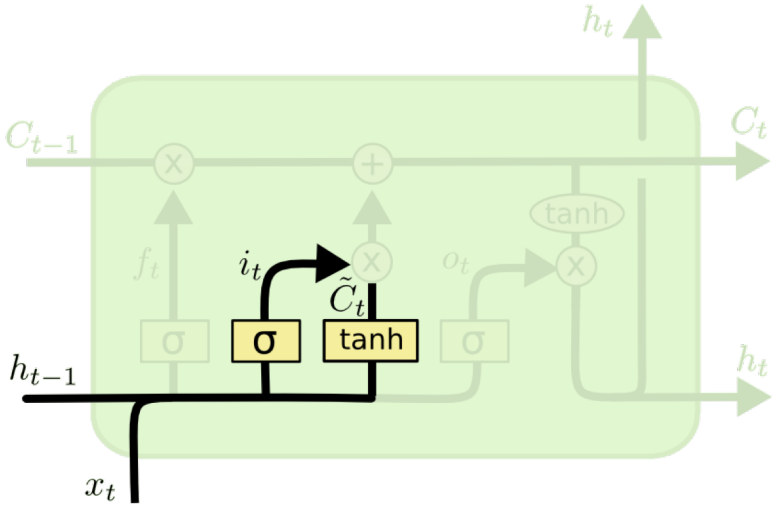
.0 – forget the cell state

1 – keep the cell state

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

LSTM walkthrough: input gate layer

- The next step is to decide what new information we're going to store in the cell state



- First, a sigmoid layer called the "input gate layer" decides which values we'll update

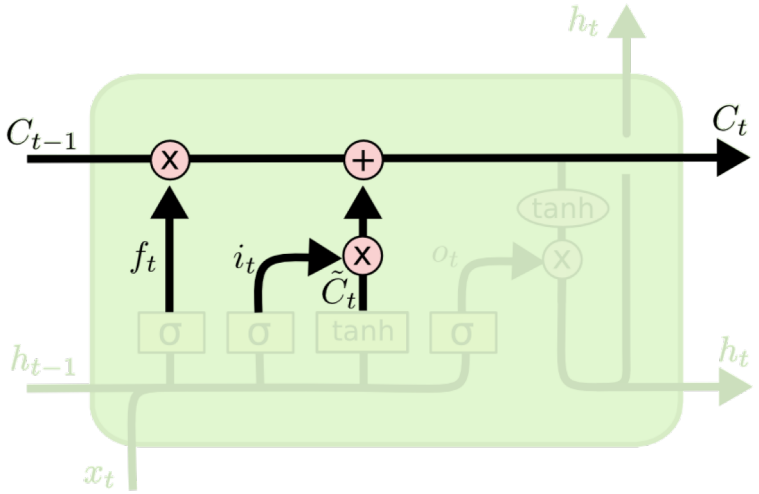
- Next, a tanh layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state.

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

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

LSTM walkthrough: update cell state

- We multiply the old state by f_t, forgetting the things we decided to forget earlier

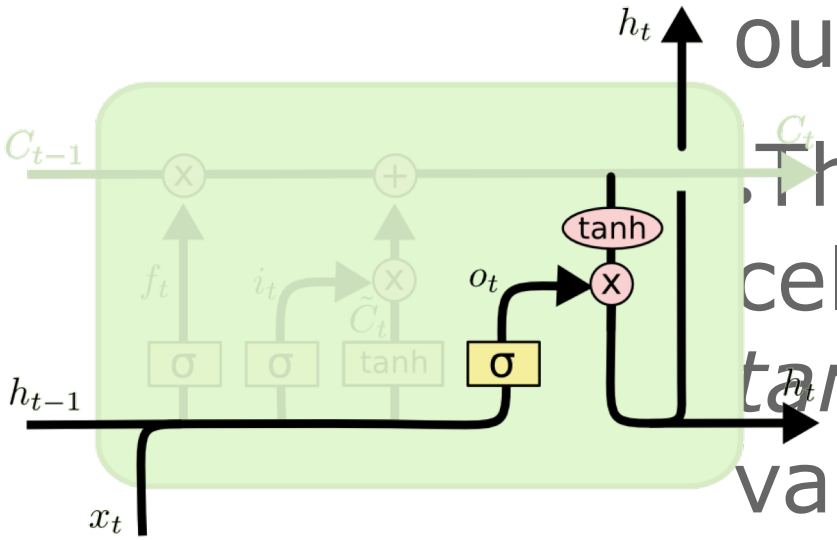


- Then we add $i_t * \hat{C}_t$
- This is the new candidate values, scaled by how much we decided to update each state value.

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

LSTM walkthrough: compute output

- Output will be based on our cell state, but will be a filtered version



- First, we run a sigmoid layer which decides what parts of the cell state we're going to output

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

- Then, we put the cell state through *tanh* (to push the values to be between −1 and 1)

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

LSTM vs RNN

- When we talk about remarkable results achieved with RNNs, we really usually talk about LSTMs
- They really work better than raw RNNs for most things
- GRU is similar to LSTM, but a bit simpler
- GRU might work better than LSTM if your training data is rather limited (and it’s faster)
- LSTM formulas seem intimidating

RNN strengths and weaknesses

- RNNs, particularly LSTMs have much more modeling power than feed-forward NNs
- However, they can take time to tune
  - Often, initial experiments produce disappointing results
  - Sensitive to learning rate, model initialization, scale of the features, etc
  - Tuning them is even deeper black magic than for feed-forward NNs
- They also require a lot of training data

Teacher-student training

- Sometimes, teacher-student training can help to learn a good LSTM
- Idea:
  - Train a feed-forward NN (like CNN) for the task (e.g., for sentiment analysis)
  - Now, train the LSTM, but instead of using real labels as targets, use **soft labels** produced by the feed-forward NN (i.e., the probability distribution predicted by the simpler model)
  - In other words, train the RNN to **mimic**

Pre-training/Transfer

- Idea:
  - First train the RNN for the simpler task that has a lot of training data
  - Then, use the pretrained RNN as part of a new model for the more complex task
- Example: LM -> Sentence Classifier
  - First train a RNN to do language modeling (predicting the next word) – lots of data available
  - Then use the trained RNN as a starting point for training a sentence classification

Questions?