CS 4803 / 7643: Deep Learning

Topics:

- LSTMs
- Language Modeling
- Recursive Neural Networks

Zsolt Kira

Georgia Tech

Administrivia

- Project comments have been given back (second iteration)

- **NOTE:** Updated PS3 https://piazza.com/class/jos18hnsf6t2cf?cid=267

Long Short Term Memory (LSTM)

Vanilla RNN

$$h_t = \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

LSTM

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

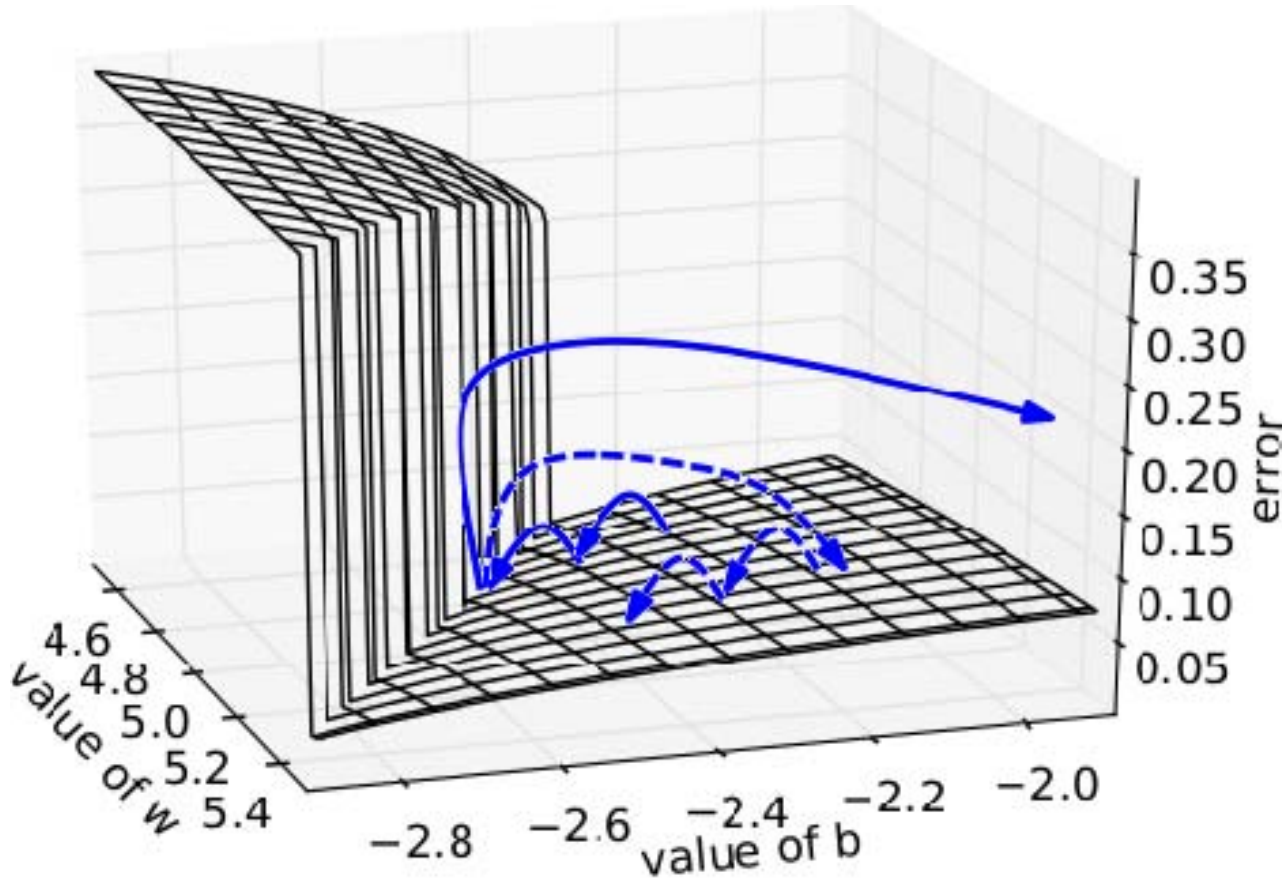
$$c_t = f \odot c_{t-1} + i \odot g$$

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

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

Illustration [Pascanu et al]

- Intuition
  - Error surface of a single hidden unit RNN; High curvature walls
  - Solid lines: standard gradient descent trajectories
  - Dashed lines: gradient rescaled to fix problem



Fix #1

- Pseudocode

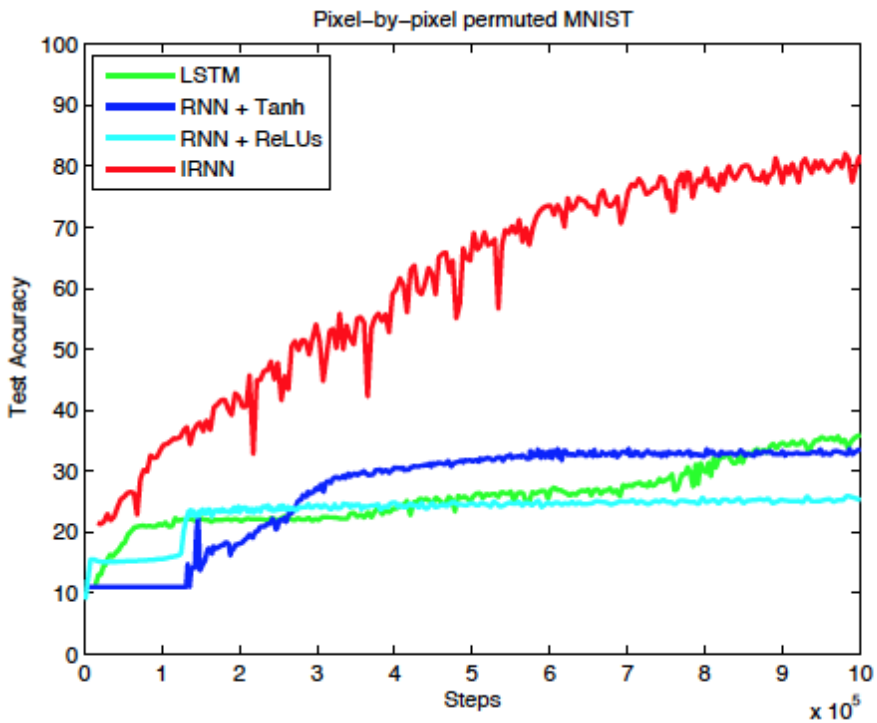
Algorithm 1 Pseudo-code for norm clipping the gradients whenever they explode

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$
end if

Image Credit: Richard Socher

Fix #2

- Smart Initialization and ReLus
  - [Socher et al 2013]
  - *A Simple Way to Initialize Recurrent Networks of Rectified Linear* Units, Le et al. 2015



Meet LSTMs

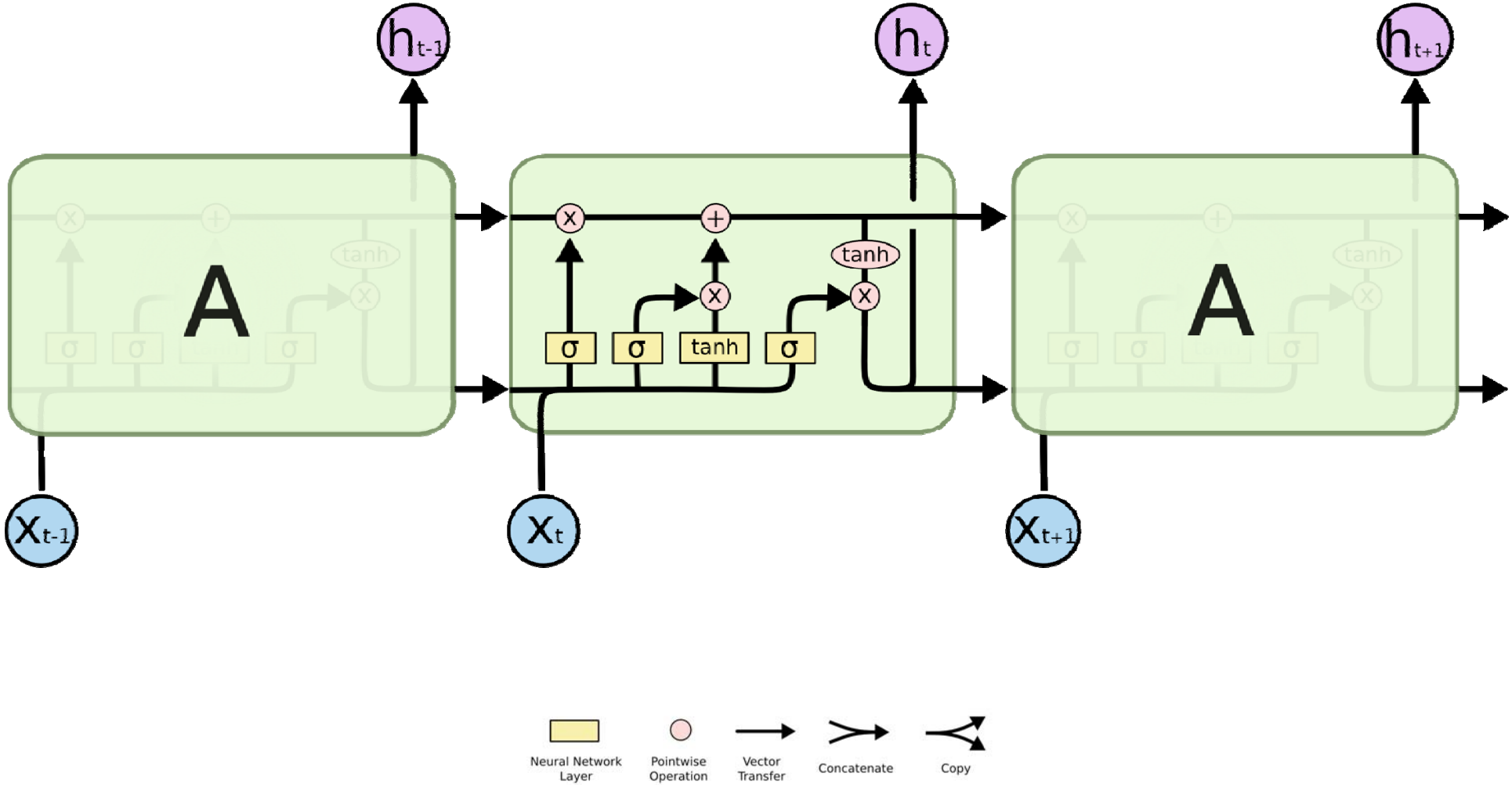


Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTMs Intuition: Memory

- Cell State / Memory

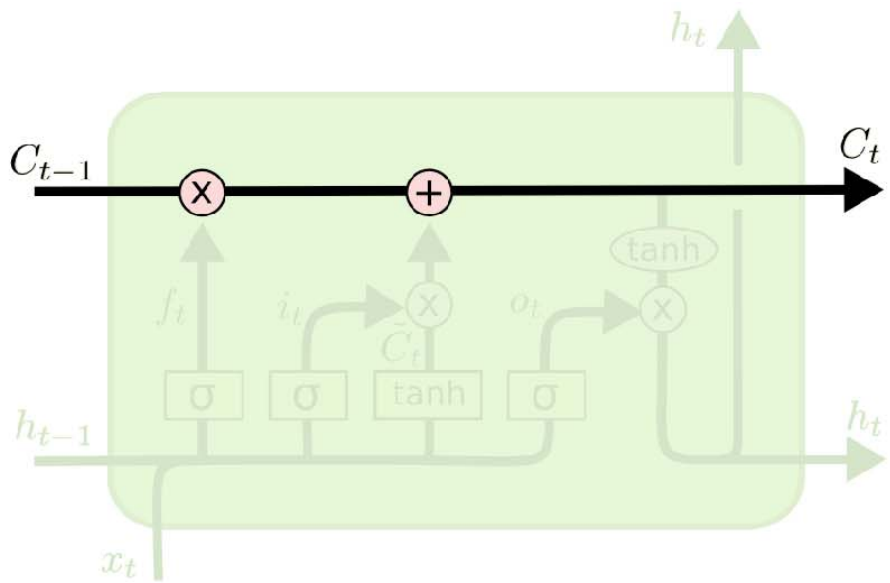
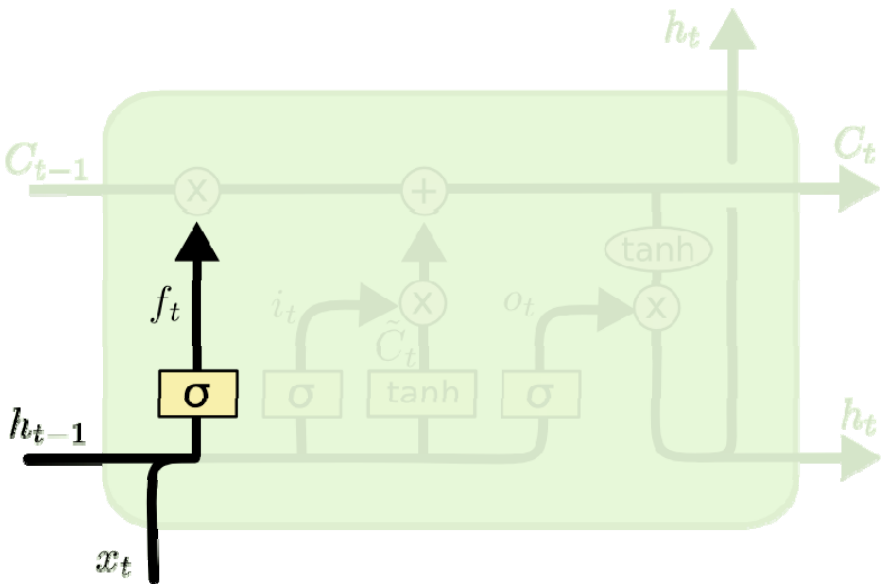


 Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTMs Intuition: Forget Gate

- Should we continue to remember this “bit” of information or not?

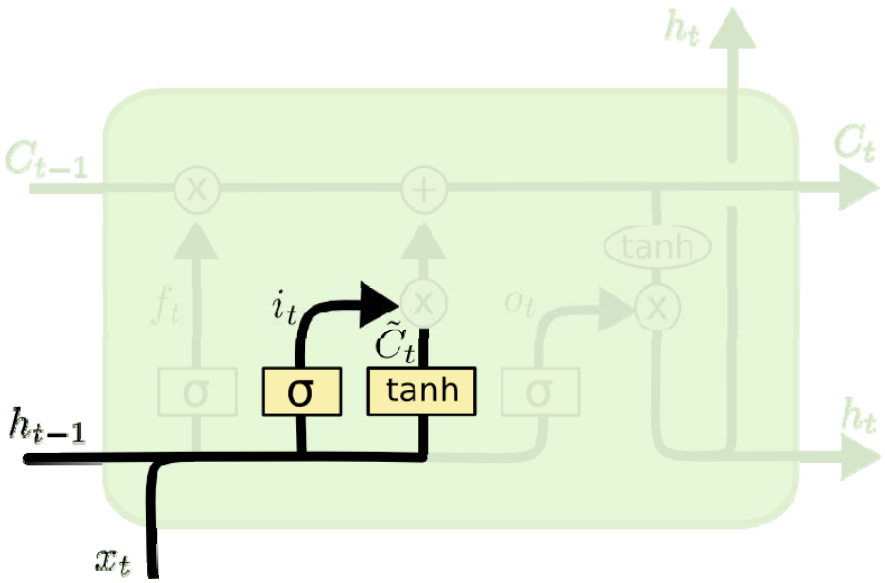


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTMs Intuition: Input Gate

- Should we update this "bit" of information or not?
  - If so, with what?



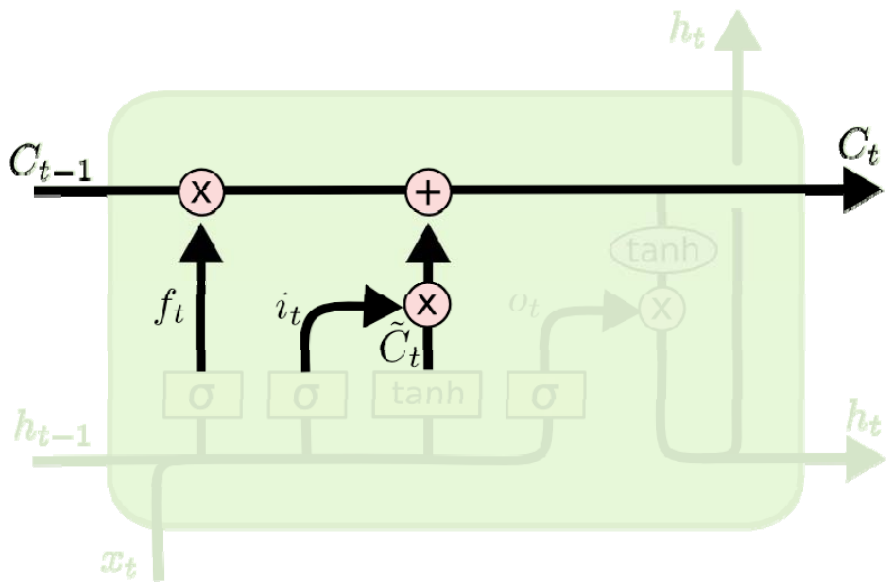
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTMs Intuition: Memory Update

- Forget that + memorize this

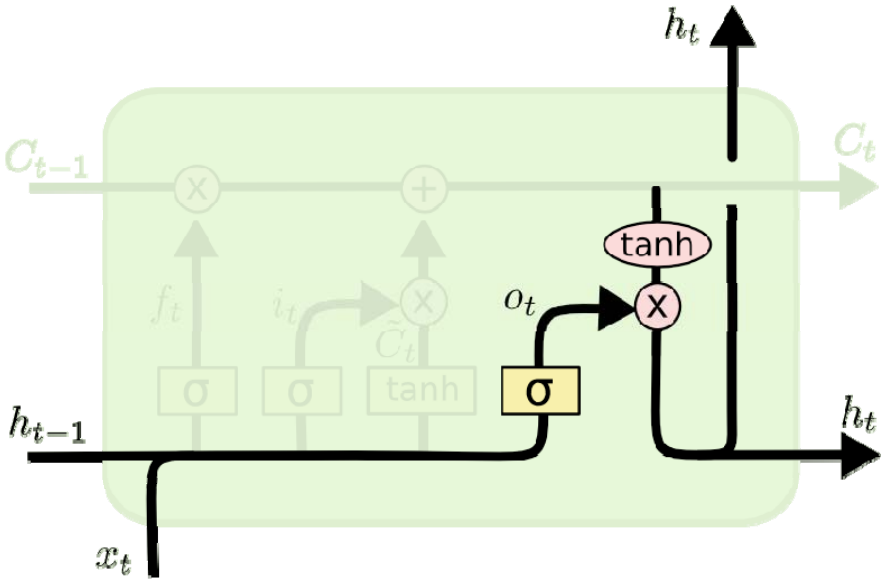


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTMs Intuition: Output Gate

- Should we output this “bit” of information to “deeper” layers?

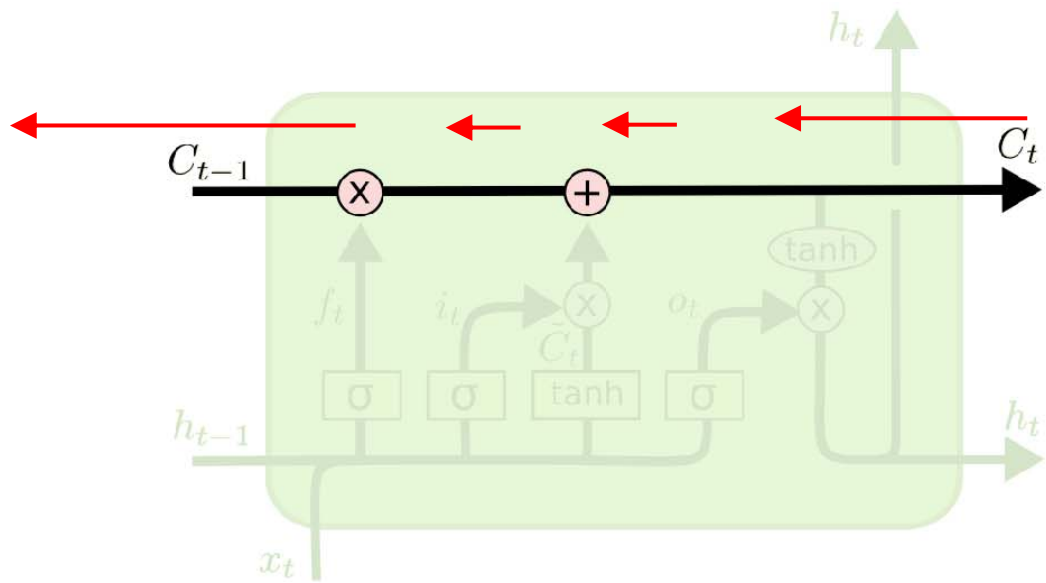


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTMs Intuition: Additive Updates



Backpropagation from c_t to c_{t-1} only elementwise multiplication by f, no matrix multiply by W

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTMs Intuition: Additive Updates

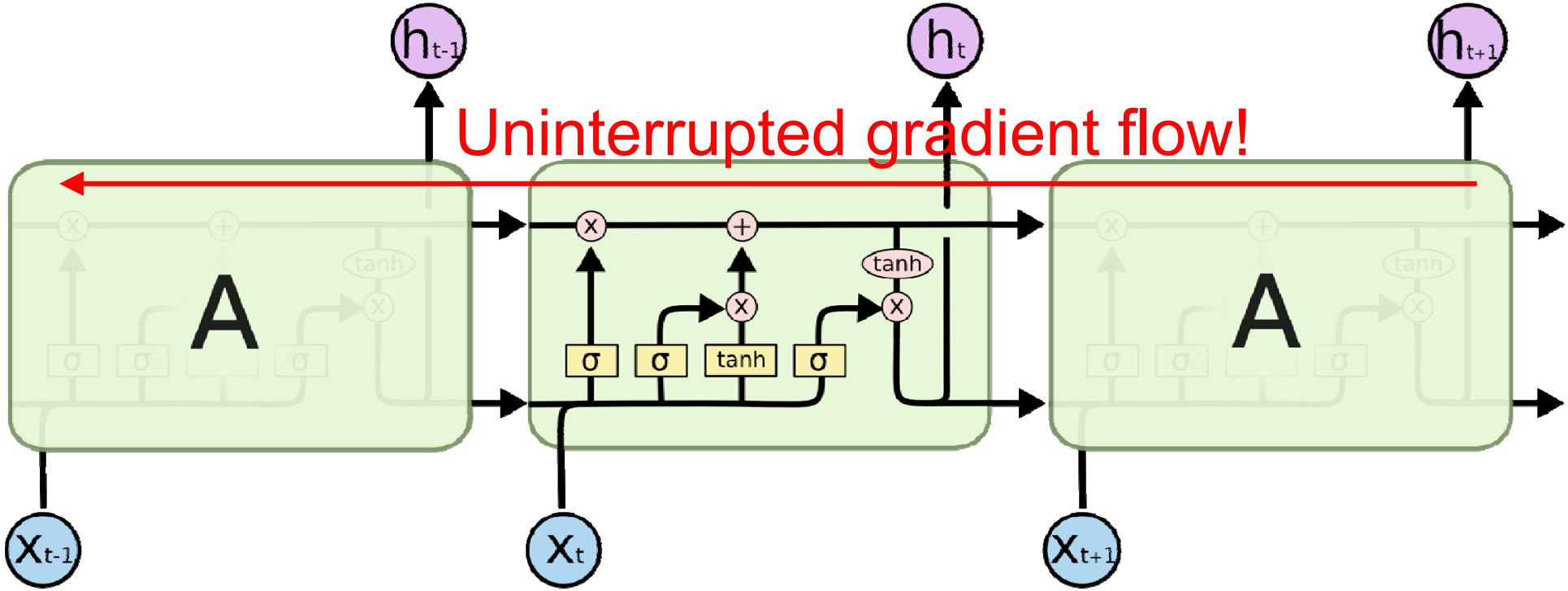
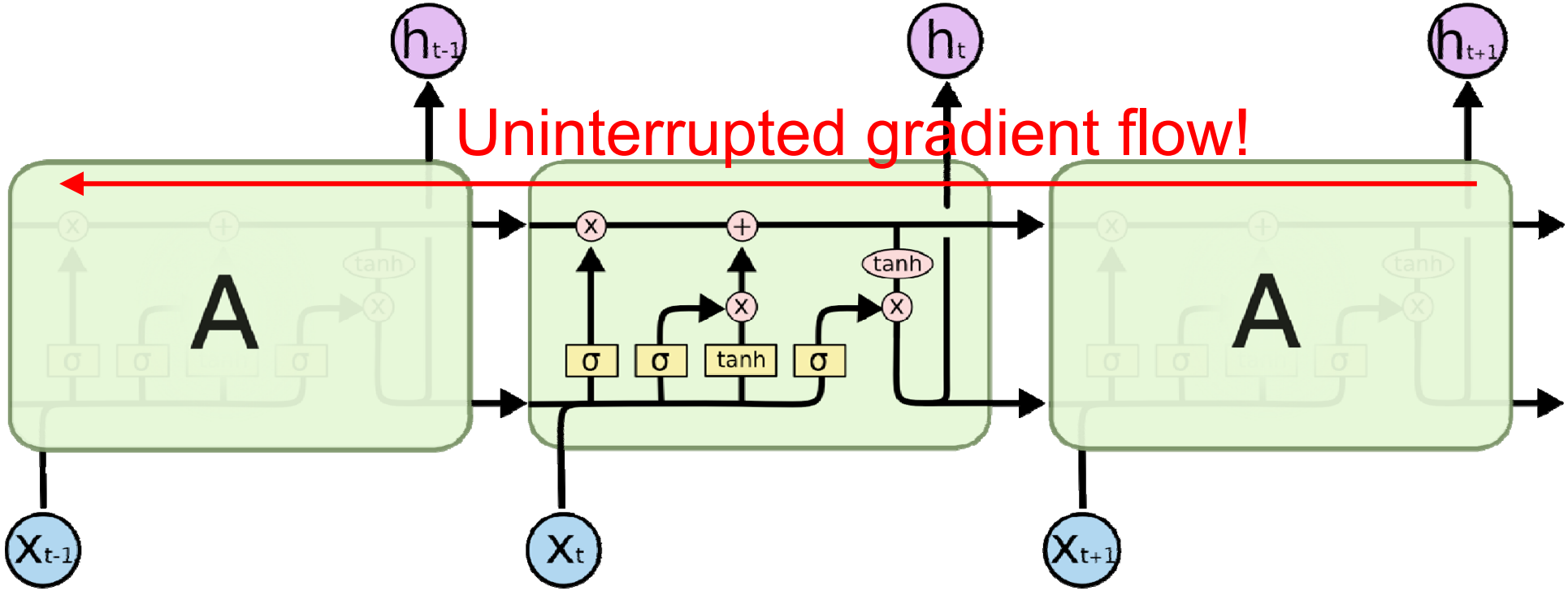


Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTMs Intuition: Additive Updates



Similar to ResNet!

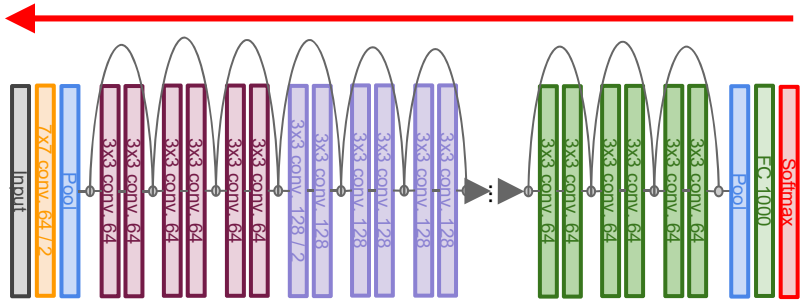


Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTMs

- A pretty sophisticated cell

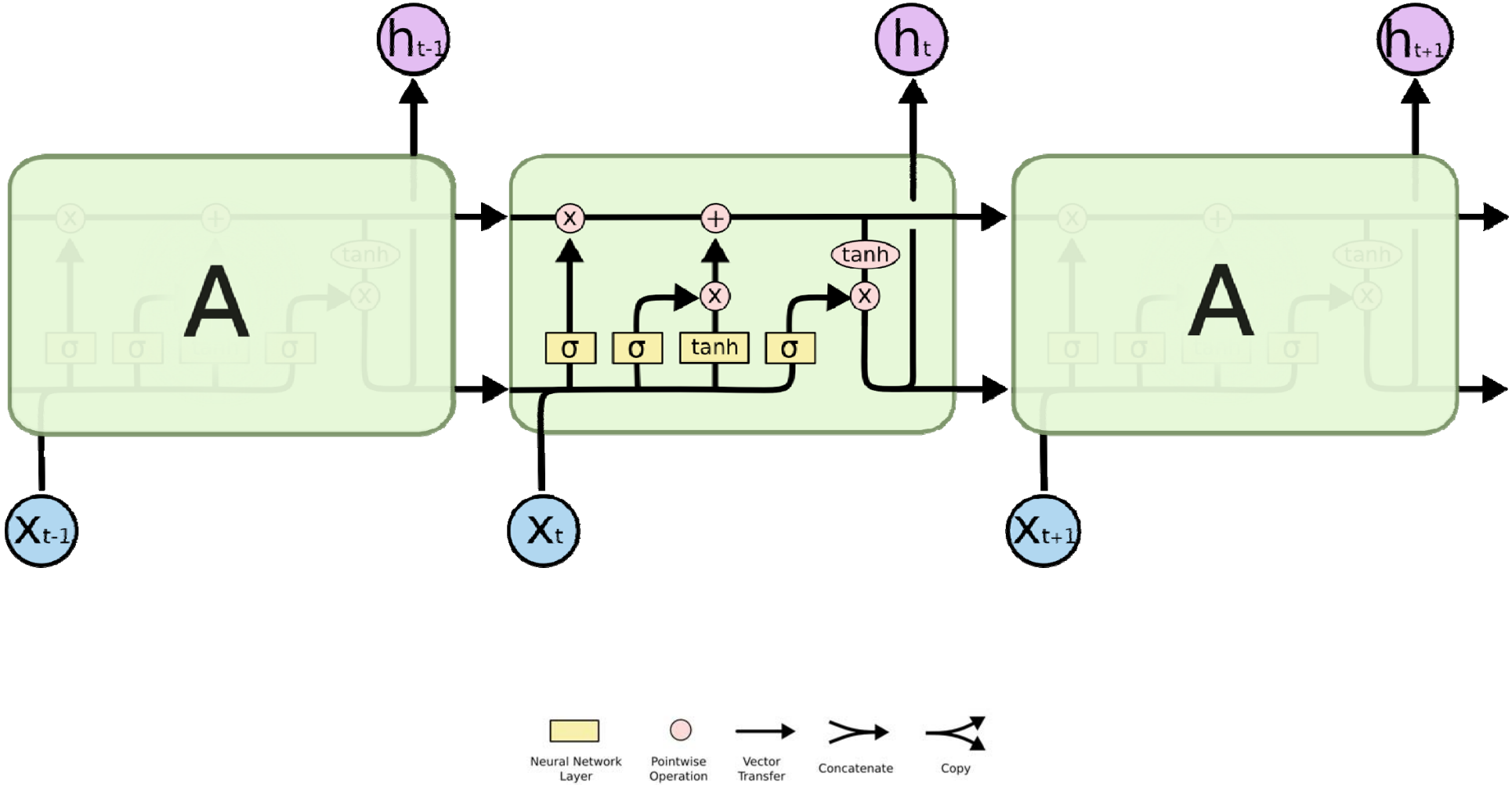
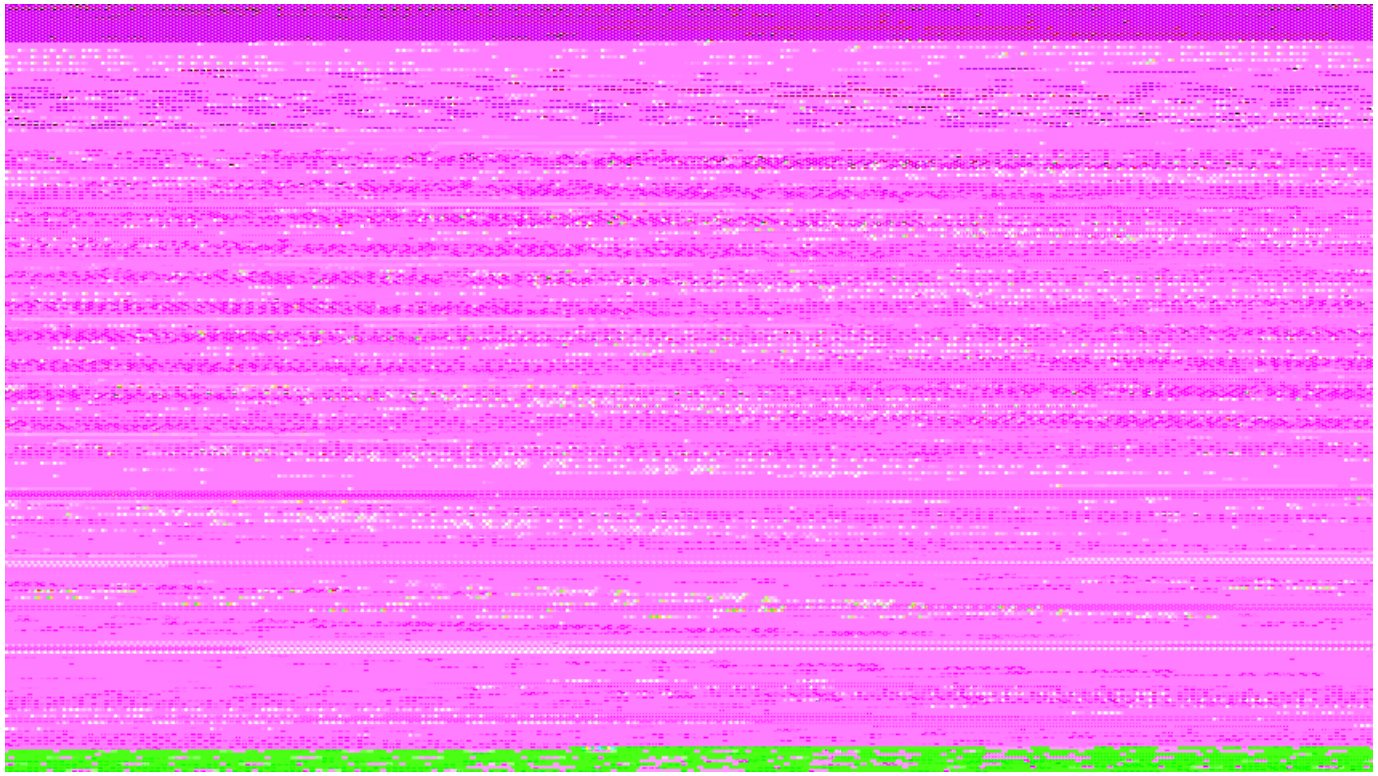


Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

Visualization

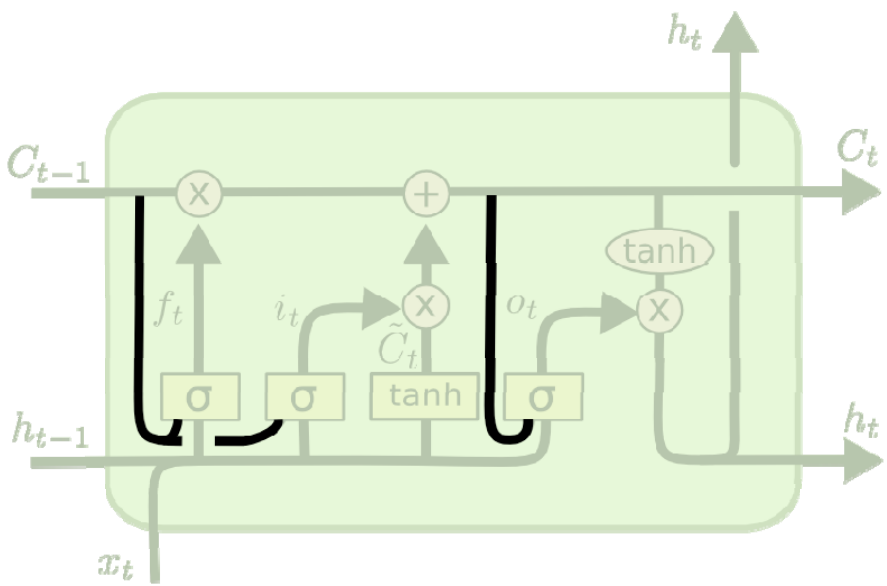
RNN **LSTM**



http://imgur.com/gallery/vaNahKE

LSTM Variants #1: Peephole Connections

- Let gates see the cell state / memory

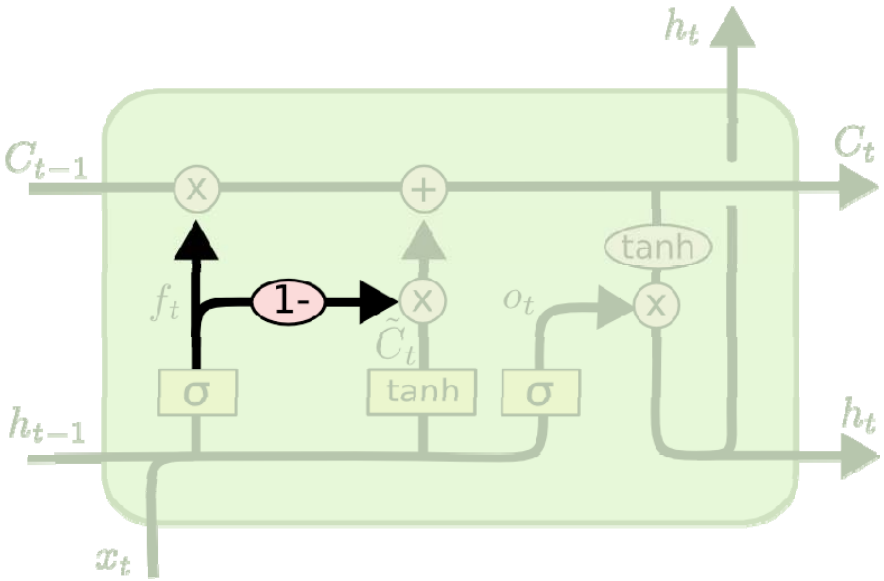


$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTM Variants #2: Coupled Gates

- Only memorize new if forgetting old

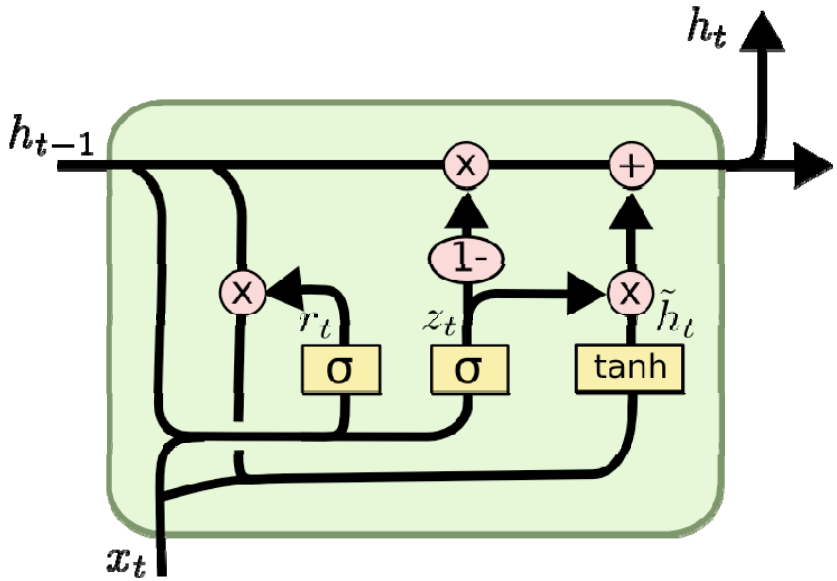


$$C_t = f_t * C_{t-1} + (1 - f_t) * \tilde{C}_t$$

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTM Variants #3: Gated Recurrent Units

- Changes:
  - No explicit memory; memory = hidden output
  - Z = memorize new and forget old



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

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

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

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

Image Credit: Christopher Olah (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

Other RNN Variants

[*An Empirical Exploration of Recurrent Network Architectures,* Jozefowicz et al., 2015]

MUT1:

$$
\begin{aligned}
z &= \mathrm{sigm}(W_{\mathrm{xz}} x_t + b_{\mathrm{z}}) \\
r &= \mathrm{sigm}(W_{\mathrm{xr}} x_t + W_{\mathrm{hr}} h_t + b_{\mathrm{r}}) \\
h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + \tanh(x_t) + b_{\mathrm{h}}) \odot z \\
&+ h_t \odot (1 - z)
\end{aligned}
$$

MUT2:

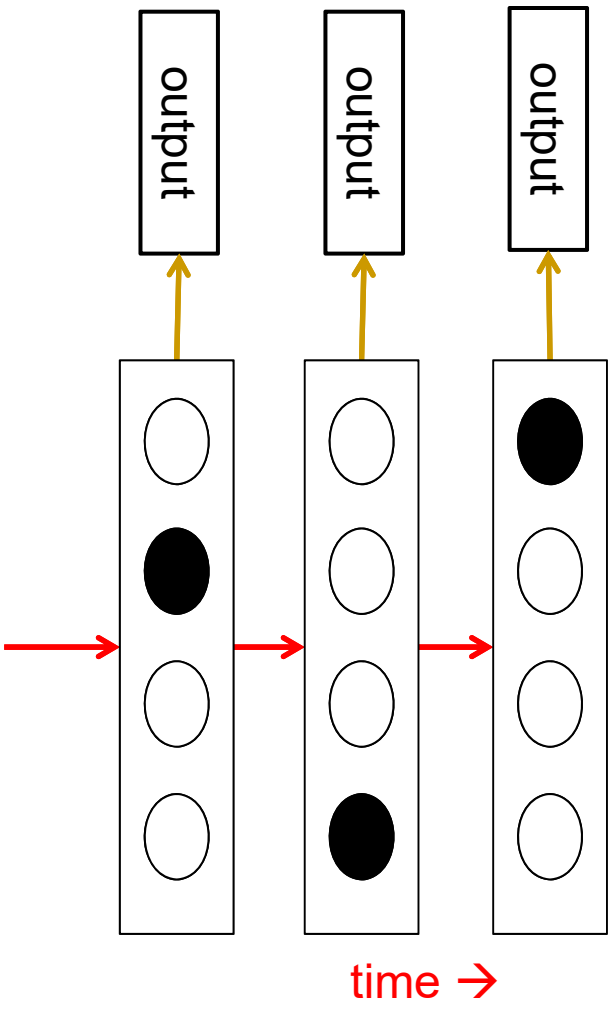
$$
\begin{aligned}
z &= \mathrm{sigm}(W_{\mathrm{xz}} x_t + W_{\mathrm{hz}} h_t + b_{\mathrm{z}}) \\
r &= \mathrm{sigm}(x_t + W_{\mathrm{hr}} h_t + b_{\mathrm{r}}) \\
h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + W_{xh} x_t + b_{\mathrm{h}}) \odot z \\
&+ h_t \odot (1 - z)
\end{aligned}
$$

MUT3:

$$
\begin{aligned}
z &= \mathrm{sigm}(W_{\mathrm{xz}} x_t + W_{\mathrm{hz}} \tanh(h_t) + b_{\mathrm{z}}) \\
r &= \mathrm{sigm}(W_{\mathrm{xr}} x_t + W_{\mathrm{hr}} h_t + b_{\mathrm{r}}) \\
h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + W_{xh} x_t + b_{\mathrm{h}}) \odot z \\
&+ h_t \odot (1 - z)
\end{aligned}
$$

Hidden Markov Models (computer scientists love them!)

- Hidden Markov Models have a discrete one-of-N hidden state. Transitions between states are stochastic and controlled by a transition matrix. The outputs produced by a state are stochastic.
  - We cannot be sure which state produced a given output. So the state is “hidden”.
  - It is easy to represent a probability distribution across N states with N numbers.
- To predict the next output we need to infer the probability distribution over hidden states.
  - HMMs have efficient algorithms for inference and learning.



A fundamental limitation of HMMs

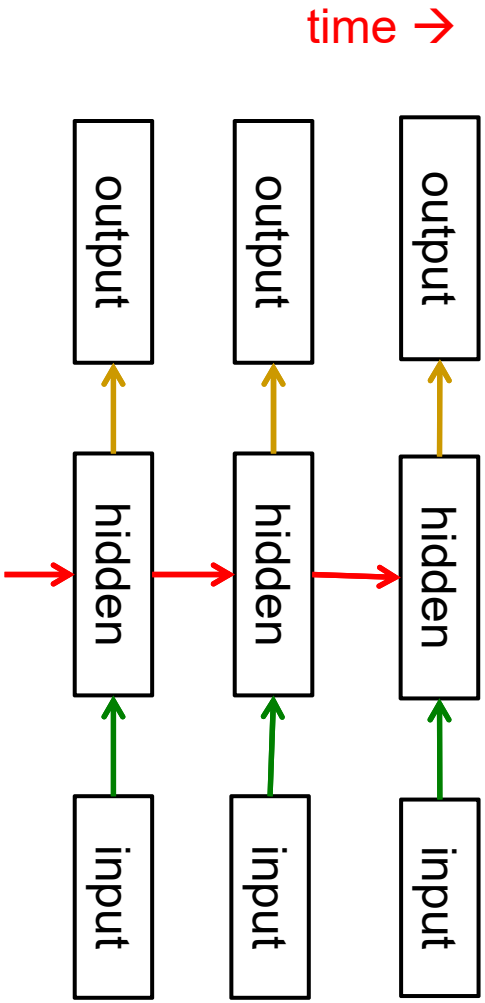
- What happens when an HMM generates data?
  - At each time step it must select one of its hidden states.
  - With N hidden states it can only remember log(N) bits about what it generated so far.

- Consider the information that the first half of an utterance contains about the second half:
  - The syntax needs to fit (e.g. number and tense agreement).
  - The semantics needs to fit. The intonation needs to fit.
  - The accent, rate, volume, and vocal tract characteristics must all fit.

- All these aspects combined could be 100 bits of information that the first half of an utterance needs to convey to the second half. 2^100 is big!

Recurrent neural networks

- RNNs are very powerful, because they combine two properties:
  - Distributed hidden state that allows them to store a lot of information about the past efficiently.
  - Non-linear dynamics that allows them to update their hidden state in complicated ways.
- With enough neurons and time, RNNs can compute anything that can be computed by your computer.
  - Universal approximation machine (Turing machine)



Language Modeling

Importance of text processing

- Text processing core to several companies (Google, Facebook, Yahoo, etc.)

- Lots of applications
  - Spam detection
  - Ad, movie, music recommendations
  - Email categorization
  - Machine Translation
  - Speech recognition
  - Image annotation
  - Generation of news articles

NLP Tasks

- Sentiment analysis
- Morphology
- Parsing
- Semantic parsing
- Paraphrase
- Analogies
- Language modeling
- Named entity recognition
- Part of speech tagging
- ...

Previous Approaches

- Previous approaches dominated by:
  - N-grams
  - Word classes
  - Bag-of-words representations
- Standard application of ML techniques
  - Logistic regression
  - SVMs
- Sound familiar?
  - Deep learning is prime candidate for text: Rich hierarchical structure that can be represented
  - Not as simple as you think

N-Grams

- Standard approach to language modeling

- Task: compute probability of a sentence *W*

$$P(W) = \prod_i P(w_i | w_1 \dots w_{i-1})$$

- Often simplified to trigrams:

$$P(W) = \prod_i P(w_i | w_{i-2,} w_{i-1})$$

N-gram example

$$P(\text{"}this\ is\ a\ sentence\text{"}) = P(this) \times P(is|this) \times P(a|this, is) \times P(sentence|is, a)$$

- The probabilities are estimated from counts:

$$P(a|this, is) = \frac{C(this\ is\ a)}{C(this\ is)}$$

- Smoothing is used to redistribute probability to unseen events (this avoids zero probabilities)

A Bit of Progress in Language Modeling (Goodman, 2001)

One-hot representations

- Simple way how to encode discrete concepts, such as words

Example:

```
vocabulary = (Monday, Tuesday, is, a, today)
Monday  = [1 0 0 0 0]
Tuesday = [0 1 0 0 0]
is      = [0 0 1 0 0]
a       = [0 0 0 1 0]
today   = [0 0 0 0 1]
```

Also known as 1-of-N (where in our case, N would be the size of the vocabulary)

Bag of Words representations

- Sum of one-hot codes
- Ignores order of words

Example:

```
vocabulary = (Monday, Tuesday, is, a, today)
Monday Monday       = [2 0 0 0 0]
today is a Monday   = [1 0 1 1 1]
today is a Tuesday  = [0 1 1 1 1]
is a Monday today   = [1 0 1 1 1]
```

Can be extended to bag-of-N-grams to capture local ordering of words

Representations using Distributional Similarity

You can get a lot of value by representing a word by means of its neighbors

"You shall know a word by the company it keeps"

(J. R. Firth 1957: 11)

One of the most successful ideas of modern statistical NLP

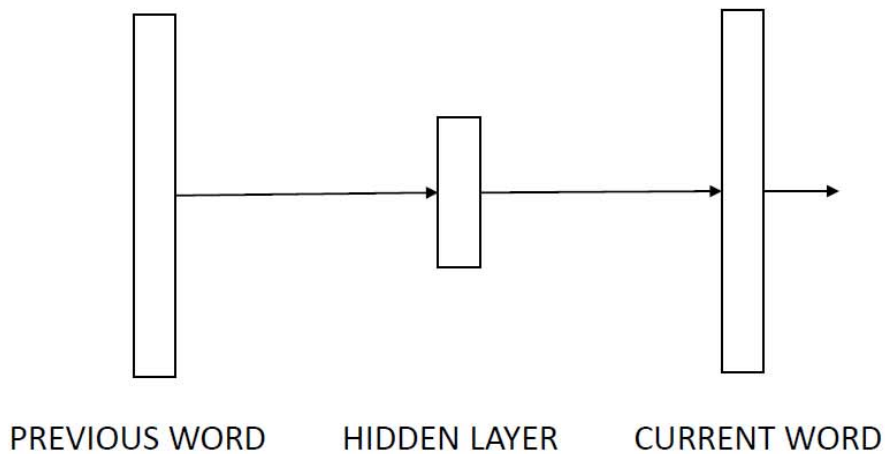
government debt problems turning into	banking	crises as has happened in
saying that Europe needs unified	banking	regulation to replace the hodgepodge

↖ These words will represent *banking* ↗

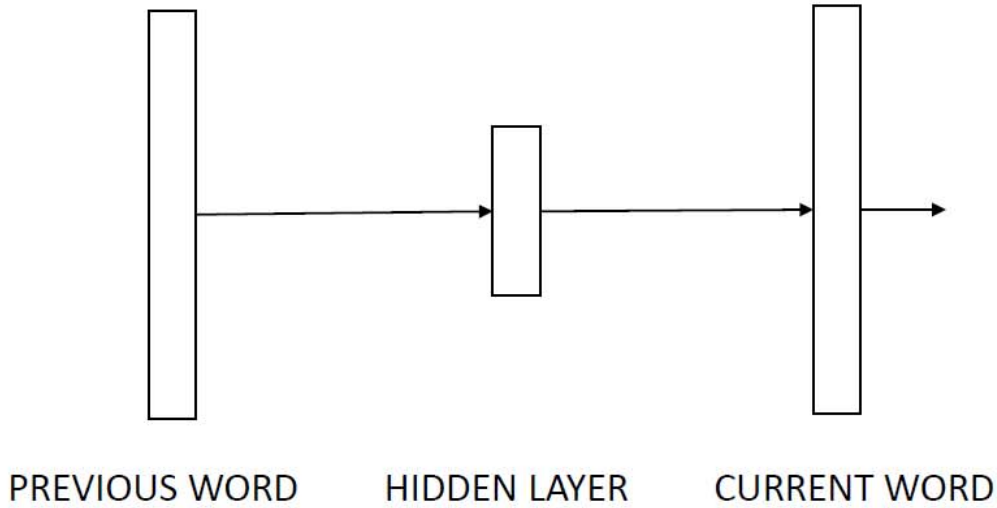
You can vary whether you use local or large context to get a more syntactic or semantic clustering

Vector representations

- Instead of a sparse one-hot vector, represent words as a **dense** vector



- Bigram neural language model
- Previous word is used to predict the current word by going through hidden layer (classifier with as many outputs as there are words in the vocabulary)



- The input is encoded as one-hot
- The model will learn compressed, continuous representations of words (usually the matrix of weights between the input and hidden layer)

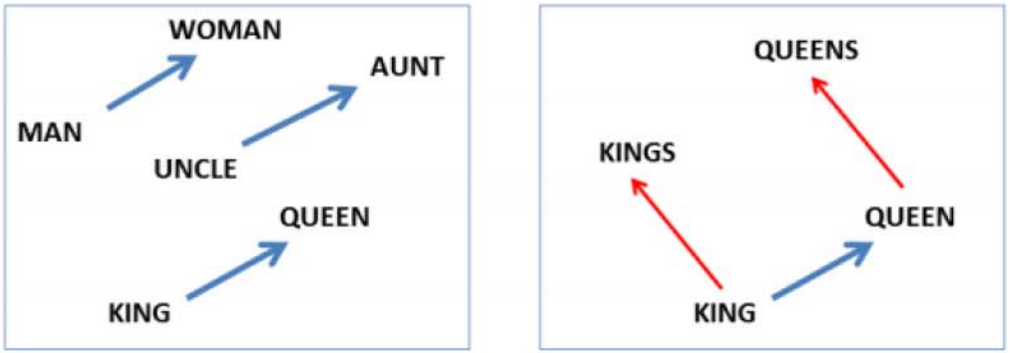
- We call the vectors in the matrix between the input and hidden layer *word vectors* (also known as *word embeddings*)
- Each word is associated with a real valued vector in N-dimensional space (usually N = 50 – 1000)
- The word vectors have some similar properties to word classes; however, many degrees of similarity are captured

Word vectors training and use

- These word vectors can be subsequently used as features in many NLP tasks (Collobert et al, 2011)
- As word vectors can be trained on huge text datasets, they provide generalization for systems trained with limited amount of supervised data
- More complex model architectures can be used for obtaining the word vectors (neural net language model with multi-task learning (Collobert & Weston, 2008))

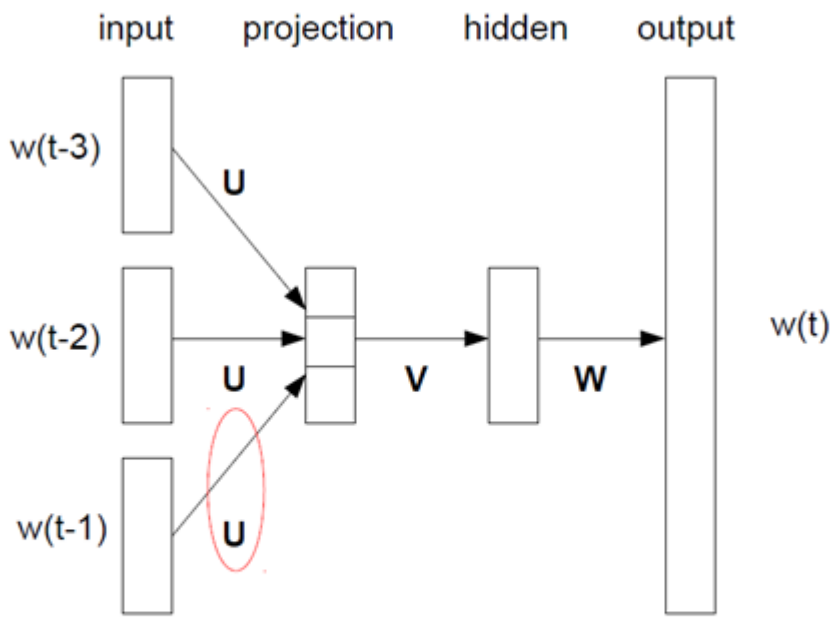
Linguistic regularities in word vectors

- Recently, it was shown that word vectors capture many linguistic properties (gender, tense, plurality, even semantic concepts like "capital city of")
- We can do nearest neighbor search around result of vector operation "King – man + woman" and obtain "Queen" (*Linguistic regularities in continuous space word representations* (Mikolov et al, 2013))



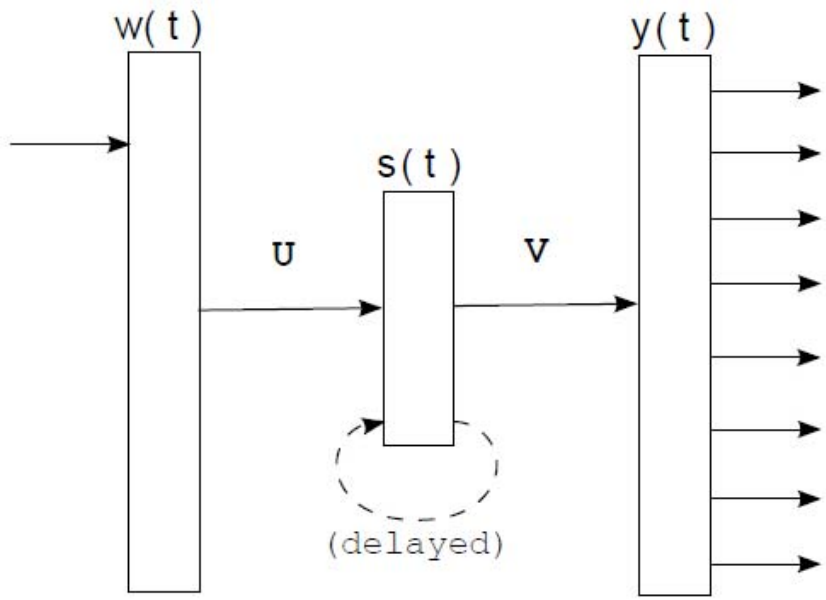
Various architectures for word vectors

- Neural net based word vectors were traditionally trained as part of neural network language model (Bengio et al, 2003)
- This models consists of input layer, projection layer, hidden layer and output layer



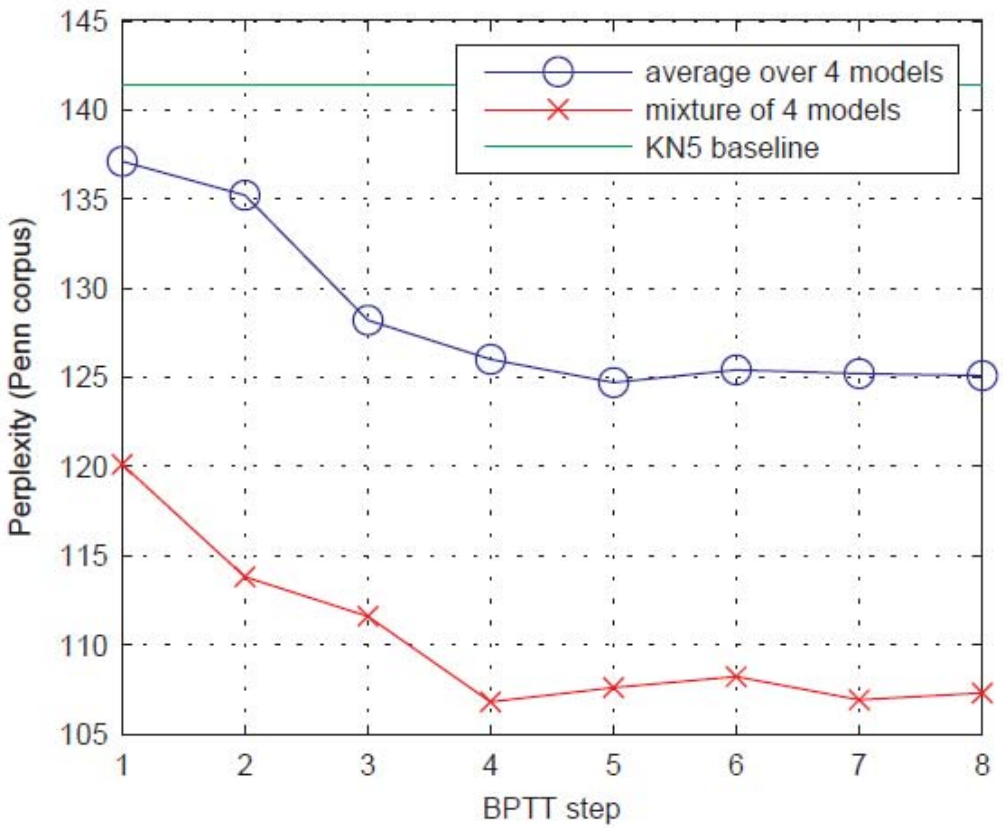
- Four-gram neural net language model architecture (Bengio 2001)
- The training is done using stochastic gradient descent and backpropagation

Word representations using RNNs



- Input layer $\mathbf{w}$ and output layer $\mathbf{y}$ have the same dimensionality as the vocabulary
- Hidden layer $\mathbf{s}$ is orders of magnitude smaller
- $\mathbf{U}$ is the matrix of weights between input and hidden layer, $\mathbf{V}$ is the matrix of weights between hidden and output layer

BPTT for different values



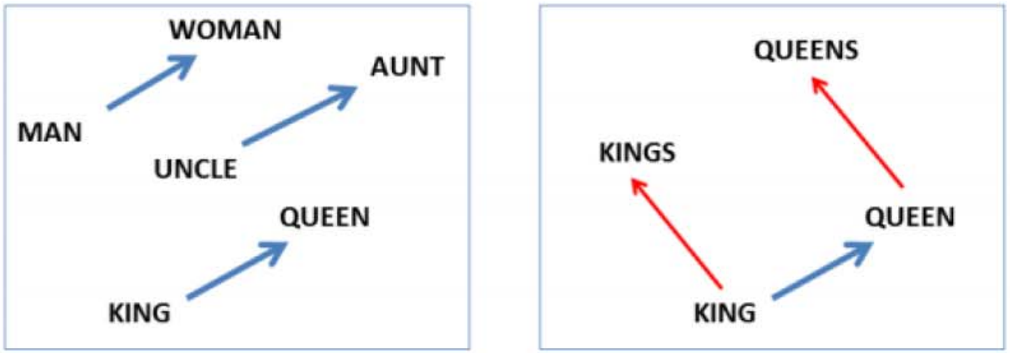
From wikipedia:

In information theory, **perplexity** is a measurement of how well a probability distribution or probability model predicts a sample. It may be used to compare probability models.

- Importance of BPTT training on Penn Corpus. BPTT=1 corresponds to standard backpropagation.

Linguistic regularities in word vectors

- Recently, it was shown that word vectors capture many linguistic properties (gender, tense, plurality, even semantic concepts like "capital city of")
- We can do nearest neighbor search around result of vector operation "King – man + woman" and obtain "Queen" (*Linguistic regularities in continuous space word representations* (Mikolov et al, 2013))



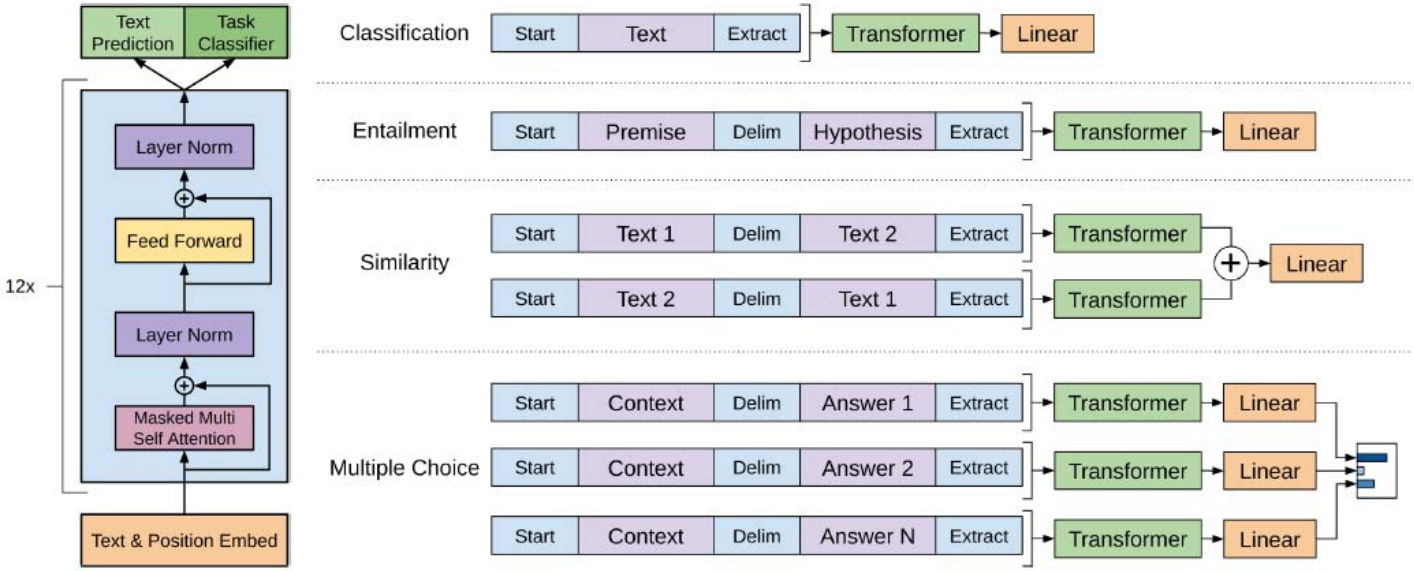
More examples

Expression	*Nearest token*
Paris - France + Italy	Rome
bigger - big + cold	colder
sushi - Japan + Germany	bratwurst
Cu - copper + gold	Au
Windows - Microsoft + Google	Android
Montreal Canadiens - Montreal + Toronto	Toronto Maple Leafs

Open AI's GPT-2

- WebText dataset – 45M link
- 8M documents after cleaning
- 40GB of text

- Tranformer Model



What is attention?

<u>Attention in neural networks:</u>

A mechanism that learns to focus on a subset of the input that is relevant to the task.

task context representation

attended feature $\longrightarrow$ $\hat{\boldsymbol{v}} = f(\boldsymbol{h}, V)$

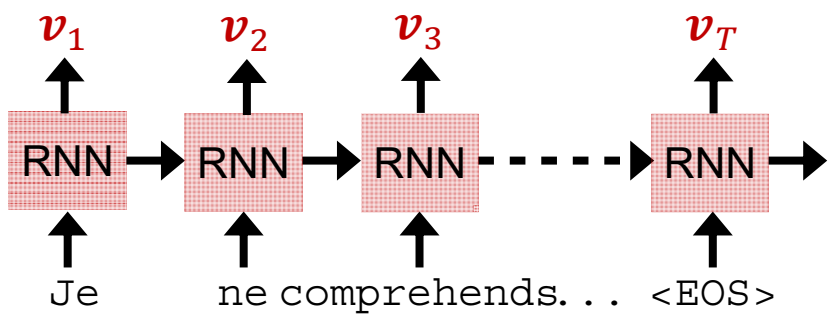
learned attention function (neural net)

set of attention candidates

46

Attention over text

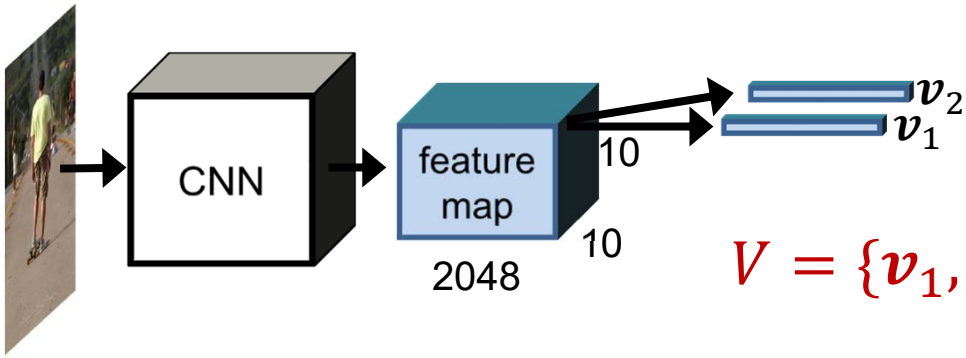
- Attention candidates, V typically defined by the hidden states of an encoder (e.g. one feature vector for each word in the input text)



$$V = \{v_1, \dots, v_T\}$$

Visual attention

- Attention candidates, V typically defined by the spatial output of a CNN (feature vectors for different parts of the image)

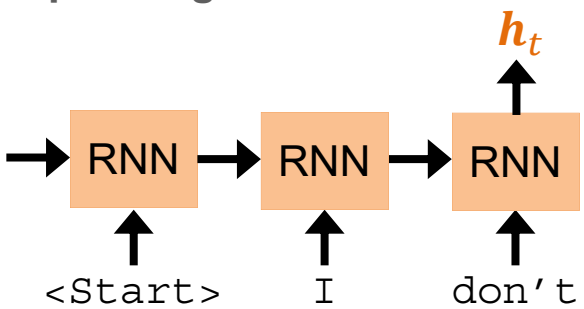


$$V = \{\boldsymbol{v}_1, \dots, \boldsymbol{v}_{100}\}$$

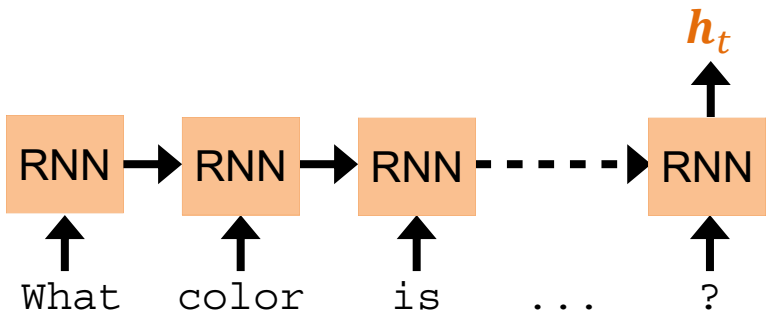
Task context

- Task context representation, $\boldsymbol{h}$, is typically an RNN encoding

Machine translation / image captioning: Decoder hidden state

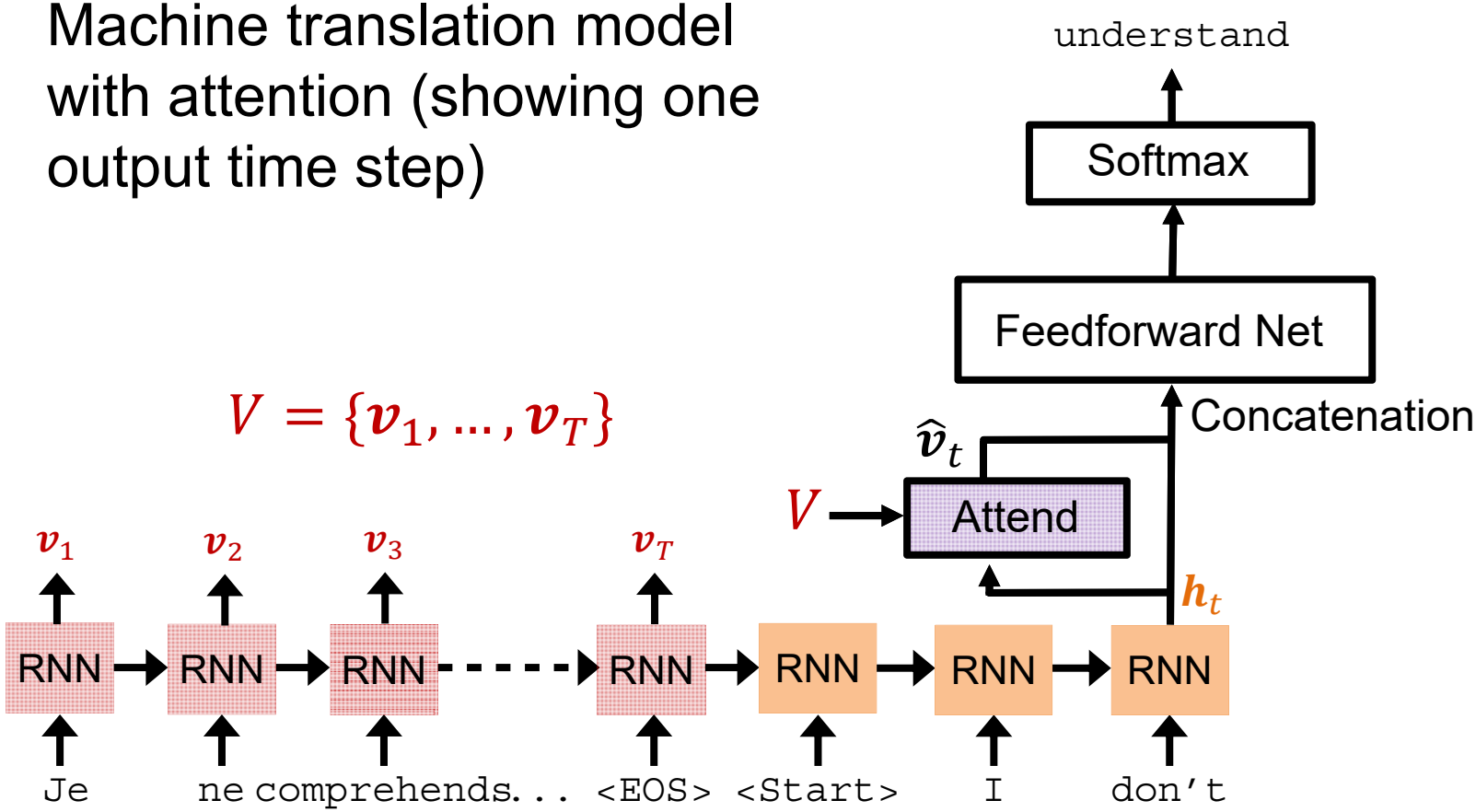


VQA: Question encoding

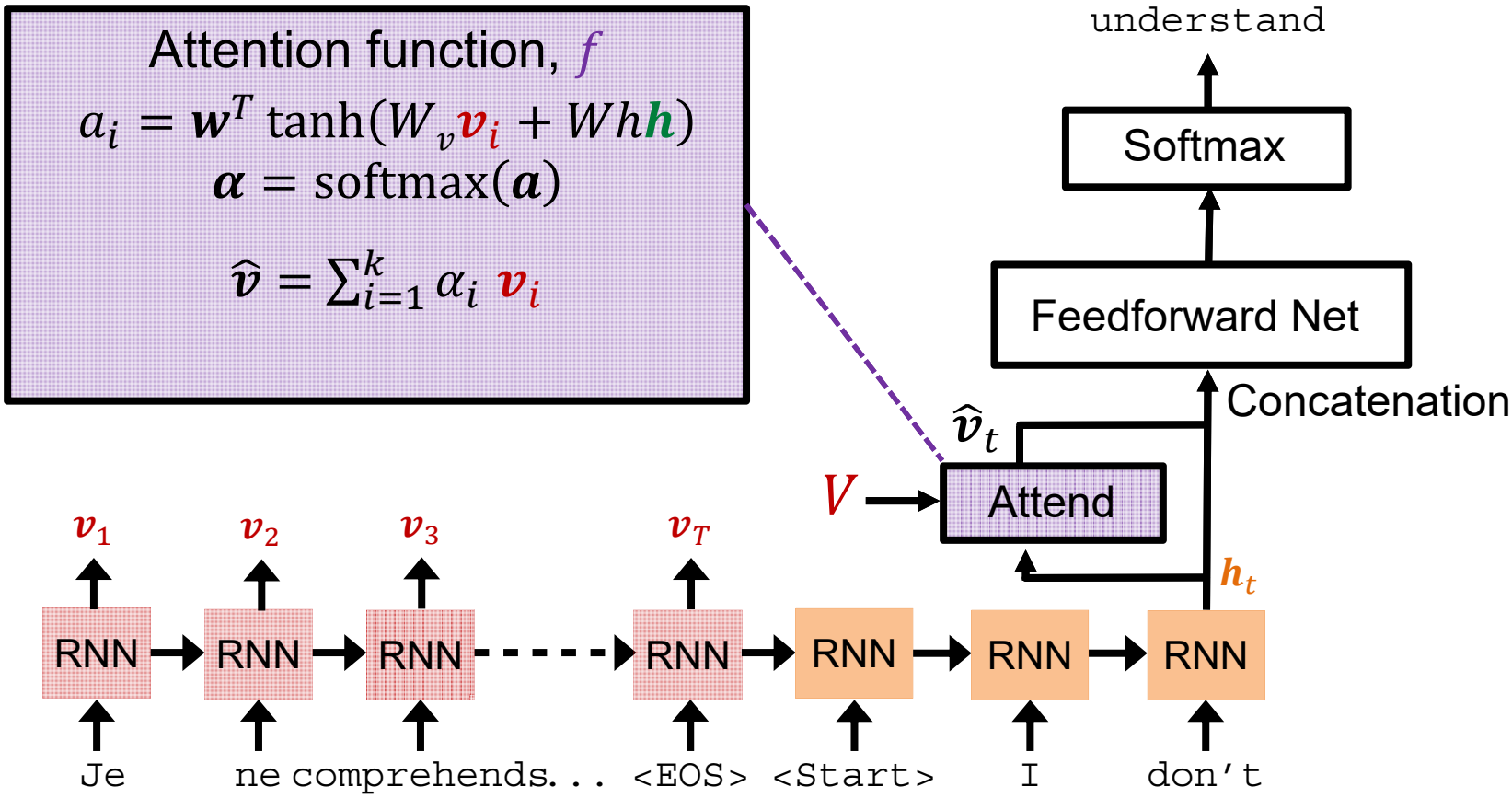


Putting it all together

- Machine translation model with attention (showing one output time step)

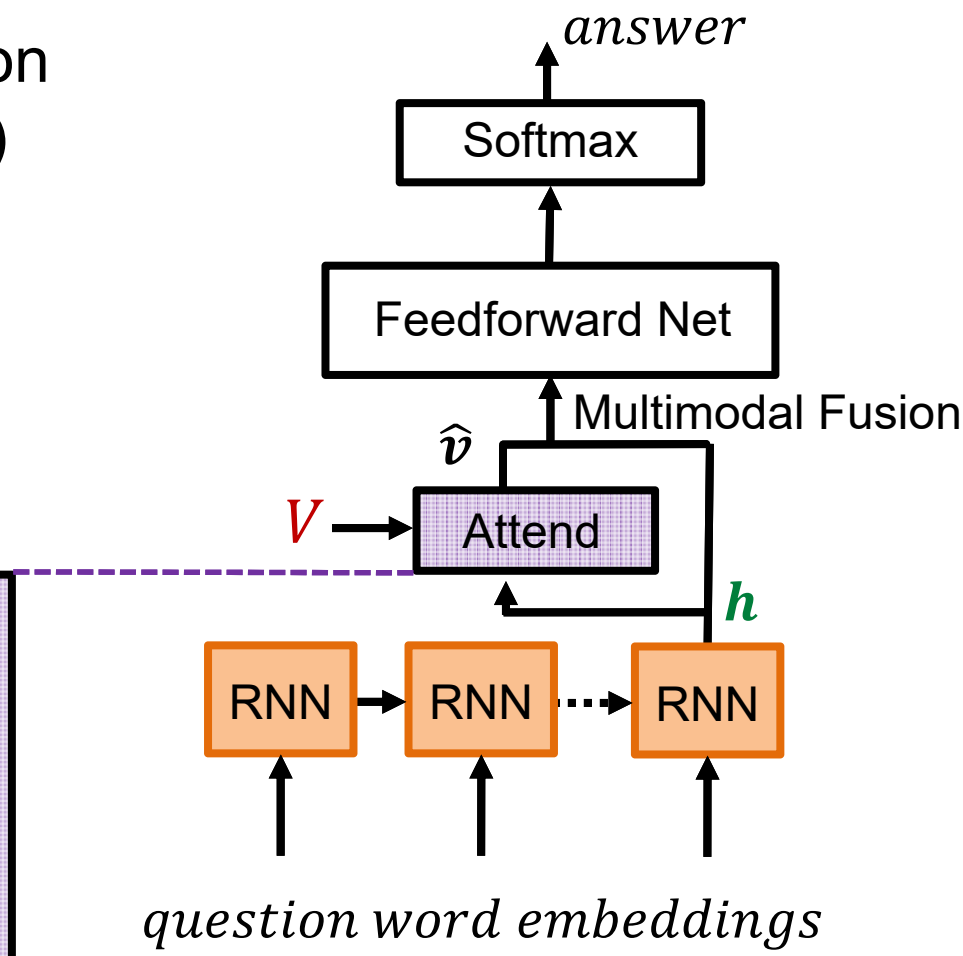


Putting it all together



Putting it all together

- VQA model with attention (showing one time step)



Attention function, f

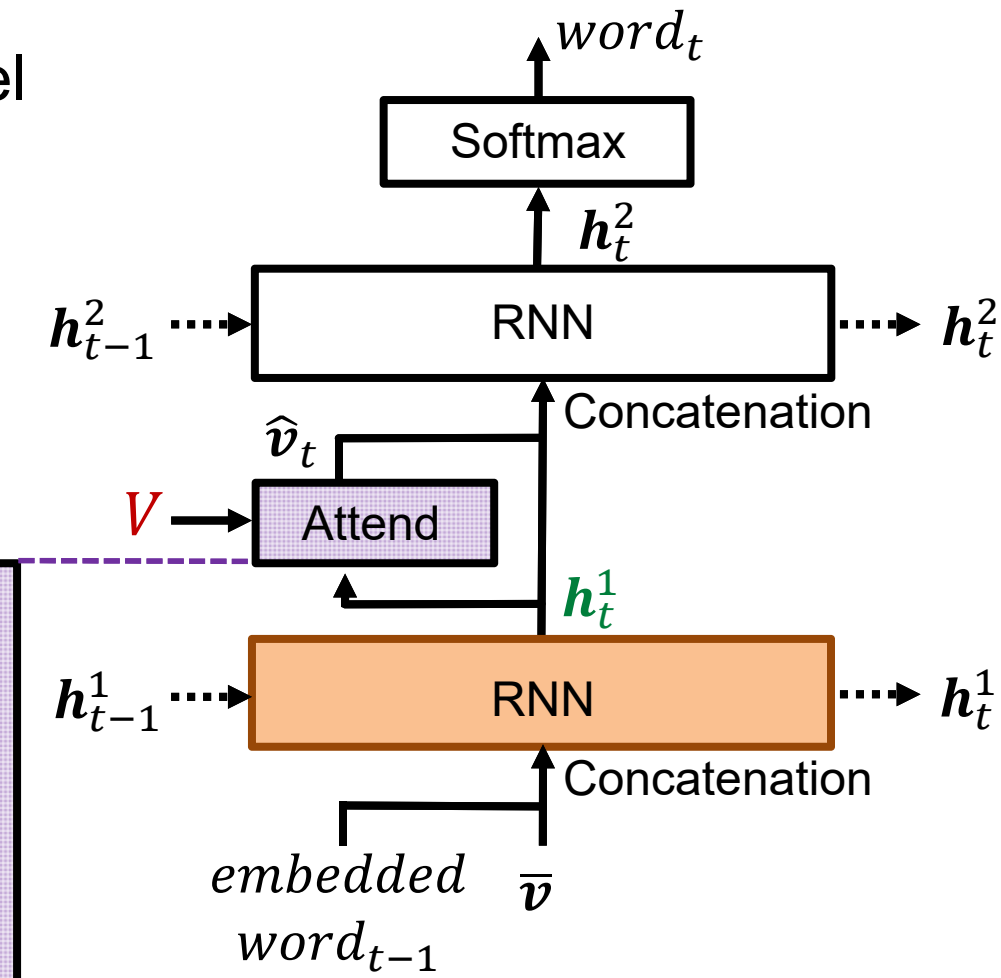
$$a_i = \boldsymbol{w}^T \tanh(W_v \boldsymbol{v}_i + Wh\boldsymbol{h})$$

$$\boldsymbol{\alpha} = \mathrm{softmax}(\boldsymbol{a})$$

$$\hat{\boldsymbol{v}} = \sum_{i=1}^{k} \alpha_i \, \boldsymbol{v}_i$$

Putting it all together

- Image captioning model with attention (showing one time step)



Attention function, f

$$a_i = \boldsymbol{w}^T \tanh(W_v \boldsymbol{v}_i + Wh\boldsymbol{h})$$

$$\boldsymbol{\alpha} = \mathrm{softmax}(\boldsymbol{a})$$

$$\hat{\boldsymbol{v}} = \sum_{i=1}^{k} \alpha_i \, \boldsymbol{v}_i$$

When Is Technology Too Dangerous to Release to the Public?

A new text-generating algorithm has reignited a long-running debate.

By AARON MAK

FEB 22, 2019 • 5:56 PM

Photo illustration by Slate. Photos by AndreyPopov/iStock/Getty Images Plus, mansum008/iStock/Getty Images Plus, and Svetlana Chernova/iStock/Getty Images Plus.

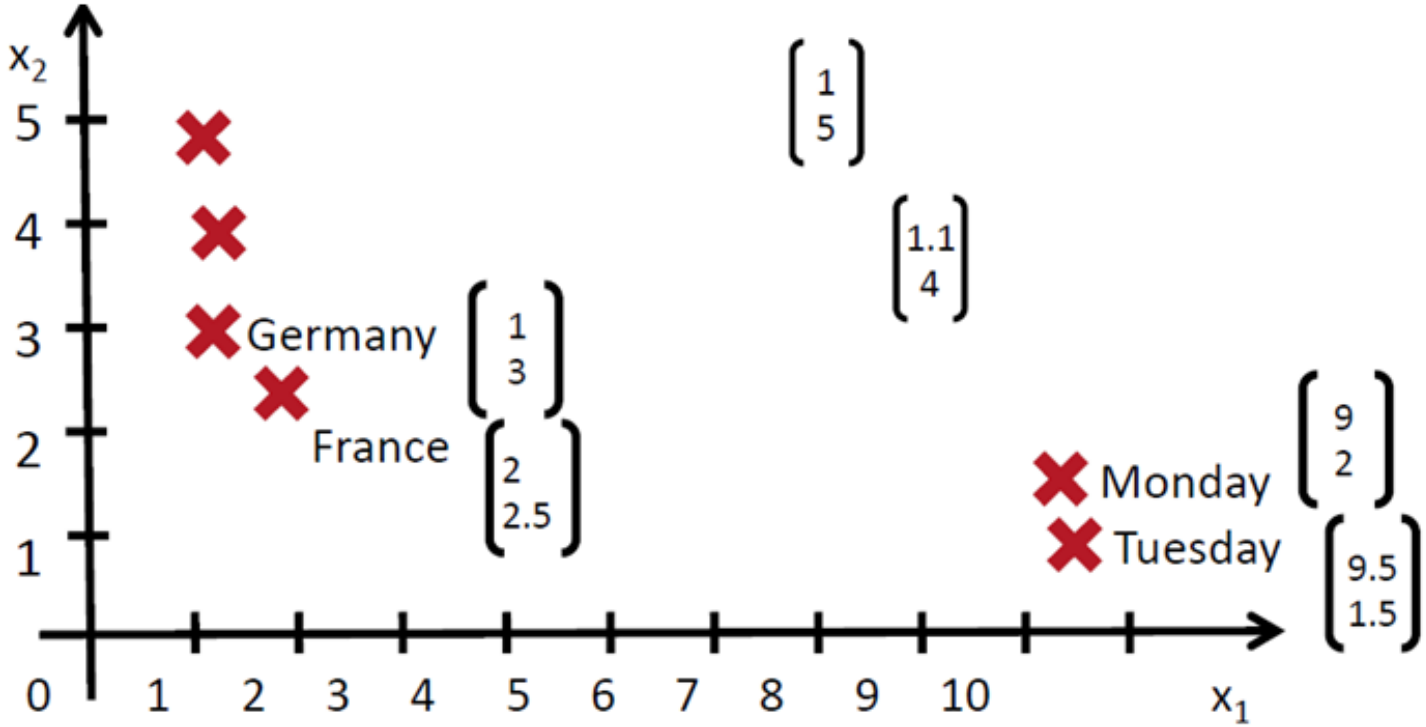
Results

Language Models are Unsupervised Multitask Learners

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	**21.8**
117M	**35.13**	45.99	**87.65**	**83.4**	**29.41**	65.85	1.16	1.17	37.50	75.20
345M	**15.60**	55.48	**92.35**	**87.1**	**22.76**	47.33	1.01	**1.06**	26.37	55.72
762M	**10.87**	**60.12**	**93.45**	**88.0**	**19.93**	**40.31**	**0.97**	**1.02**	22.05	44.575
1542M	**8.63**	**63.24**	**93.30**	**89.05**	**18.34**	**35.76**	**0.93**	**0.98**	**17.48**	42.16

Table 3. Zero-shot results on many datasets. No training or fine-tuning was performed for any of these results. PTB and WikiText-2 results are from (Gong et al., 2018). CBT results are from (Bajgar et al., 2016). LAMBADA accuracy result is from (Hoang et al., 2018) and LAMBADA perplexity result is from (Grave et al., 2016). Other results are from (Dai et al., 2019).

Building on Word Vector Space Models



the country of my birth
the place where I was born

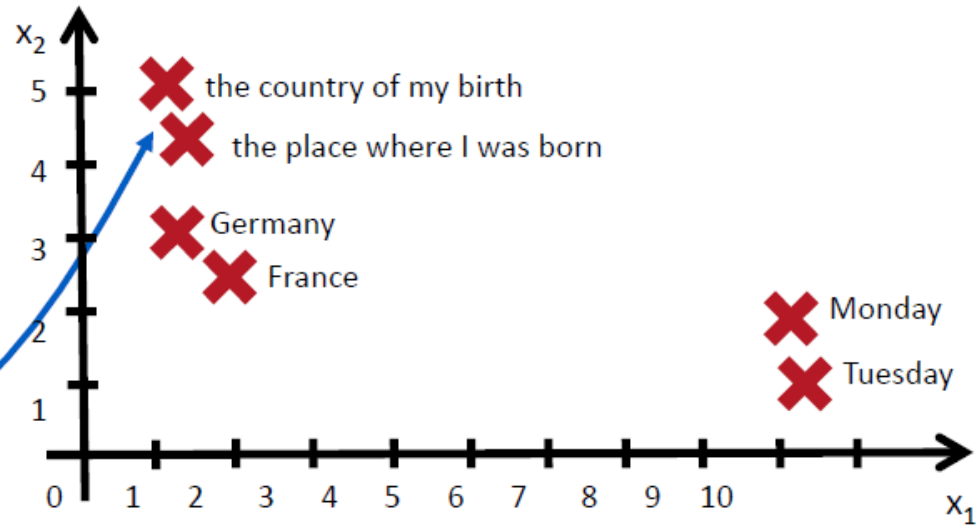
But how can we represent the meaning of longer phrases?

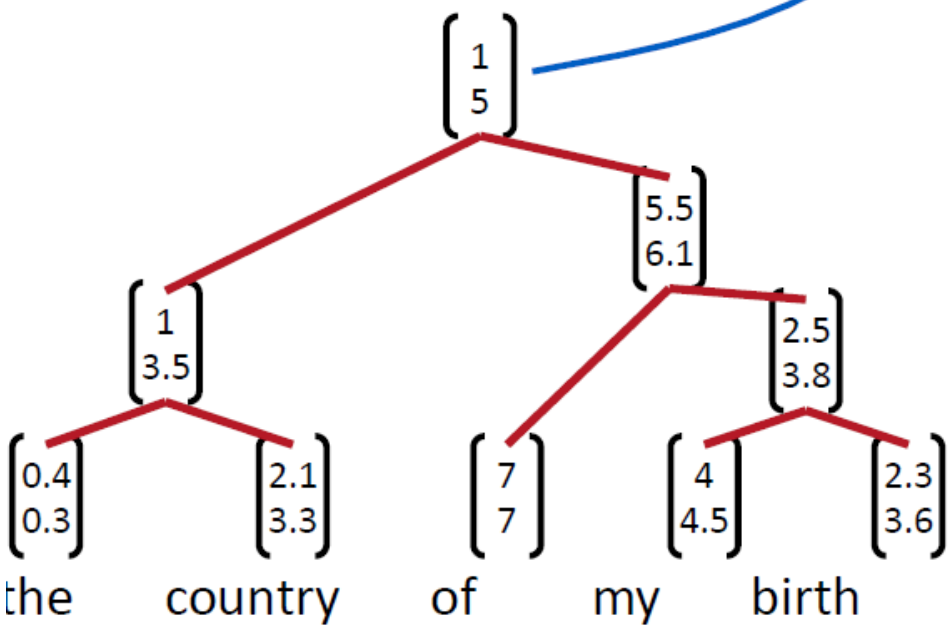
By mapping them into the same vector space!

Mapping Phrases to Vector Spaces

Use principle of compositionality

The meaning (vector) of a sentence is determined by
(1) the meanings of its words and
(2) the rules that combine them.





Models in this section can jointly learn parse trees and compositional vector representations