Language Technologies Institute

Carnegie Mellon University

Advanced Multimodal Machine Learning

Lecture 4.1: Recurrent Networks

Louis-Philippe Morency
Tadas Baltrusaitis

Lecture Objectives

- Word representations & distributional hypothesis
  - Learning neural representations (e.g., Word2vec)
- Language models
- Sequence modeling tasks
- Recurrent neural networks
- Backpropagation through time
- Gates recurrent neural networks
  - Long Short-Term Memory (LSTM) model

Administrative Stuff

Upcoming Schedule

- Pre-proposal (tomorrow Wednesday 2/5 at 9am)
- First project assignment:
  - Proposal presentation (2/21 and 2/23)
  - First project report (Sunday 3/5)
- Second project assignment
  - Midterm presentations (4/6 and 4/8)
  - Midterm report (Sunday 4/9)
- Final project assignment
  - Final presentation (5/2 & 5/4)
  - Final report (Sunday 5/7)

Distributed Semantics

Possible ways of representing words

Given a text corpus containing 100,000 unique words

- Classic binary word representation: [0; 0; 0; 0;….; 0; 0; 1; 0;…; 0; 0]

  100,000d vector

  - Only non-zero at the index of the word

- Classic word feature representation: [5; 1; 0; 0;….; 0; 20; 1; 0;…; 3; 0]

  300d vector

  - Manually define 300 "good" features (e.g., ends on –ing)

- Learned word representation: [0,1; 0,0003; 0;….; 0,02; 0.08; 0,05]

  300d vector

  - This 300-dimension vector should approximate the "meaning" of the word

The Distributional Hypothesis

- Distribution Hypothesis (DH) [Lenci 2008]
  - At least certain aspects of the meaning of lexical expressions depend on their distributional properties in the linguistic contexts
  - The degree of semantic similarity between two linguistic expressions α and β is a function of the similarity of the linguistic contexts in which α and β can appear
- Weak and strong DH
  - Weak view as a quantitative method for semantic analysis and lexical resource induction
  - Strong view as a cognitive hypothesis about the form and origin of semantic representations; assuming that word distributions in context play a specific *causal role* in forming meaning representations.

What is the meaning of "bardiwac"?

- He handed her glass of bardiwac.
- Beef dishes are made to complement the bardiwacs.
- Nigel staggered to his feet, face flushed from too much bardiwac.
- Malbec, one of the lesser-known bardiwac grapes, responds well to Australia's sunshine.
- I dined off bread and cheese and this excellent bardiwac.
- The drinks were delicious: blood-red bardiwac as well as light, sweet Rhenish.

⇒ bardiwac is a heavy red alcoholic beverage made from grapes

Geometric interpretation

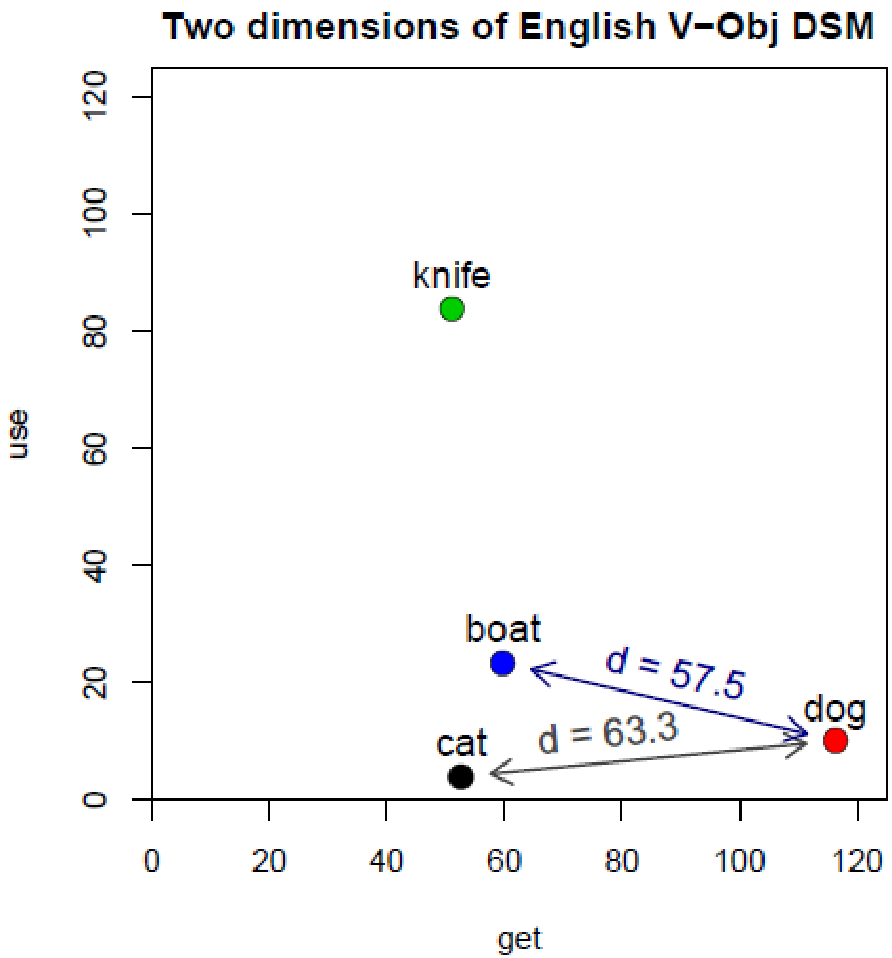
- row vector $\mathbf{x}_{dog}$ describes usage of word *dog* in the corpus
- can be seen as coordinates of point in *n*-dimensional Euclidean space R^n

	get	see	use	hear	eat	kill
knife	51	20	84	0	3	0
cat	52	58	4	4	6	26
dog	115	83	10	42	33	17
boat	59	39	23	4	0	0
cup	98	14	6	2	1	0
pig	12	17	3	2	9	27
banana	11	2	2	0	18	0

co-occurrence matrix **M**

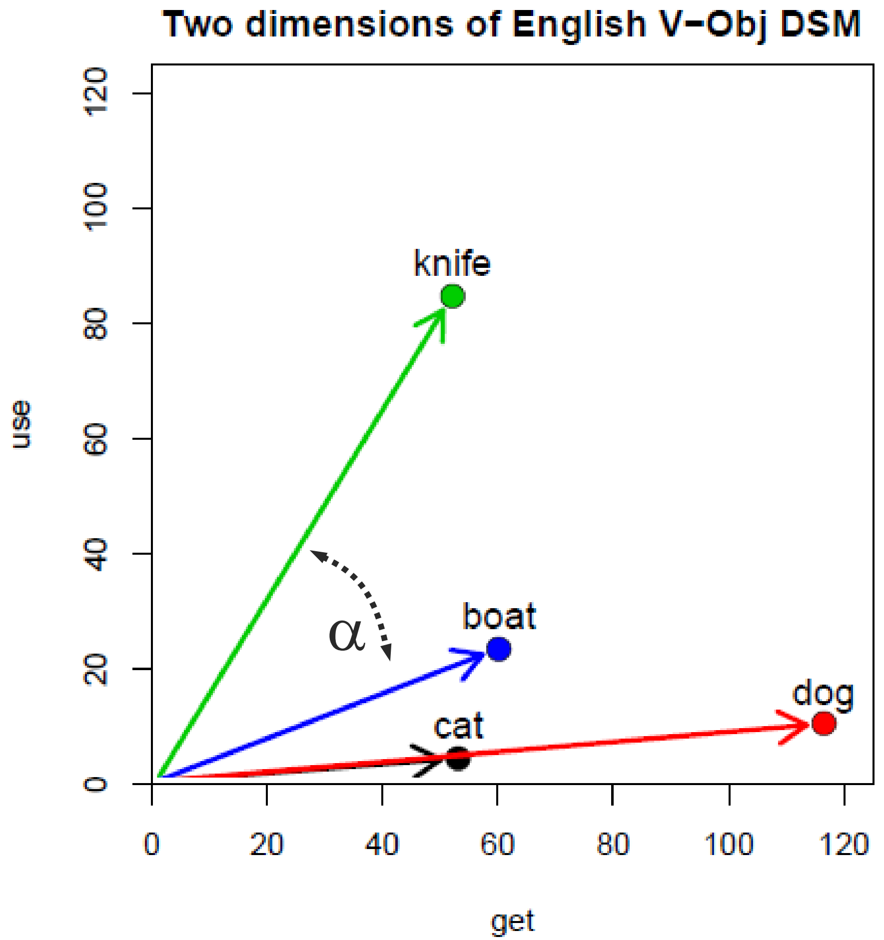
Distance and similarity

- illustrated for two dimensions: *get* and *use*: $\mathbf{x}_{\text{dog}} = (115, 10)$
- similarity = spatial proximity (Euclidean distance)
- location depends on frequency of noun ($f_{\text{dog}} \approx 2.7 \cdot f_{\text{cat}}$)

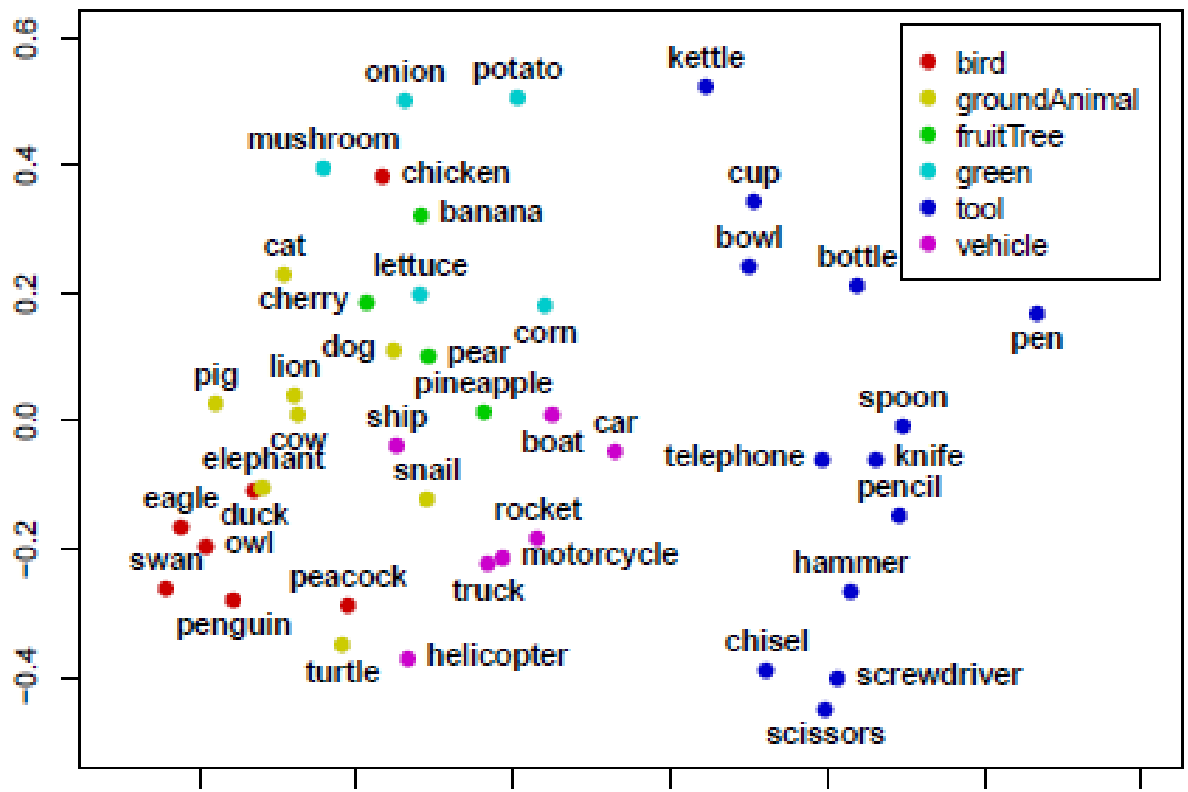


Angle and similarity

- direction more important than location
- normalise "length" $||\mathbf{x}_{dog}||$ of vector
- or use angle α as distance measure



Semantic maps



Learning Neural Representations of Words

How to learn (word) features/representations?

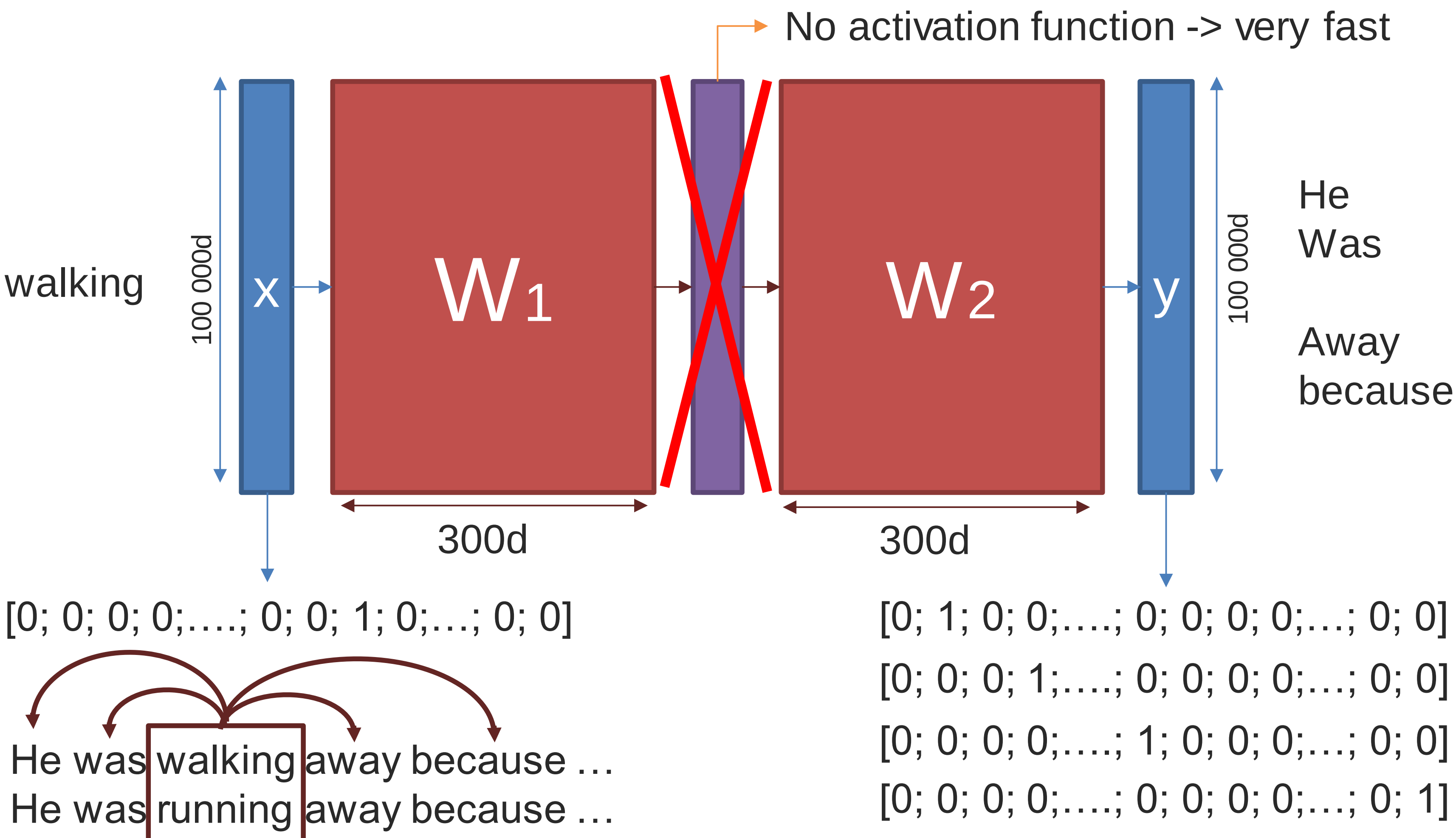
- **Distribution hypothesis:** Approximate the word meaning by its surrounding words
- Words used in a similar context will lie close together

He was walking away because …
He was running away because …

- **Instead of capturing co-occurrence counts directly, predict surrounding words of every word**

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

How to learn (word) features/representations?



Word2vec algorithm: https://code.google.com/p/word2vec/

How to use these word representations

If we would have a vocabulary of 100 000 words:

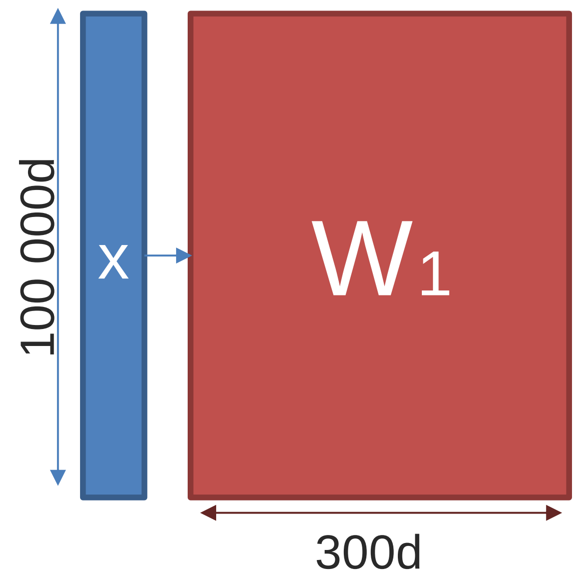
Classic NLP: 100 000 dimensional vector

Walking: [0; 0; 0; 0;….; 0; 0; 1; 0;…; 0; 0]

Running: [0; 0; 0; 0;….; 0; 0; 0; 0;…; 1; 0]

Similarity = 0.0

Transform: x'=x*W



Goal: 300 dimensional vector

Walking: [0,1; 0,0003; 0;….; 0,02; 0.08; 0,05]

Running: [0,1; 0,0004; 0;….; 0,01; 0.09; 0,05]

Similarity = 0.9

Word representations: examples

Word similarity test:

- Trained on 400 million tweets having 5 billion words

Input: running	Cosine similarity
runnin	0.758099
runing	0.702119
Running	0.69014
runnning	0.669039
sprinting	0.587385
runnung	0.578426
run	0.576671
walking/running	0.563114
runin	0.556682
walking	0.542137

Input: :)	Cosine similarity
:))	0.885355
=)	0.836011
:D	0.818340
;))	0.814380
(:	0.809806
:)))	0.808298
:-)	0.798115
:))))	0.777765
;)	0.772422
:-))	0.758584

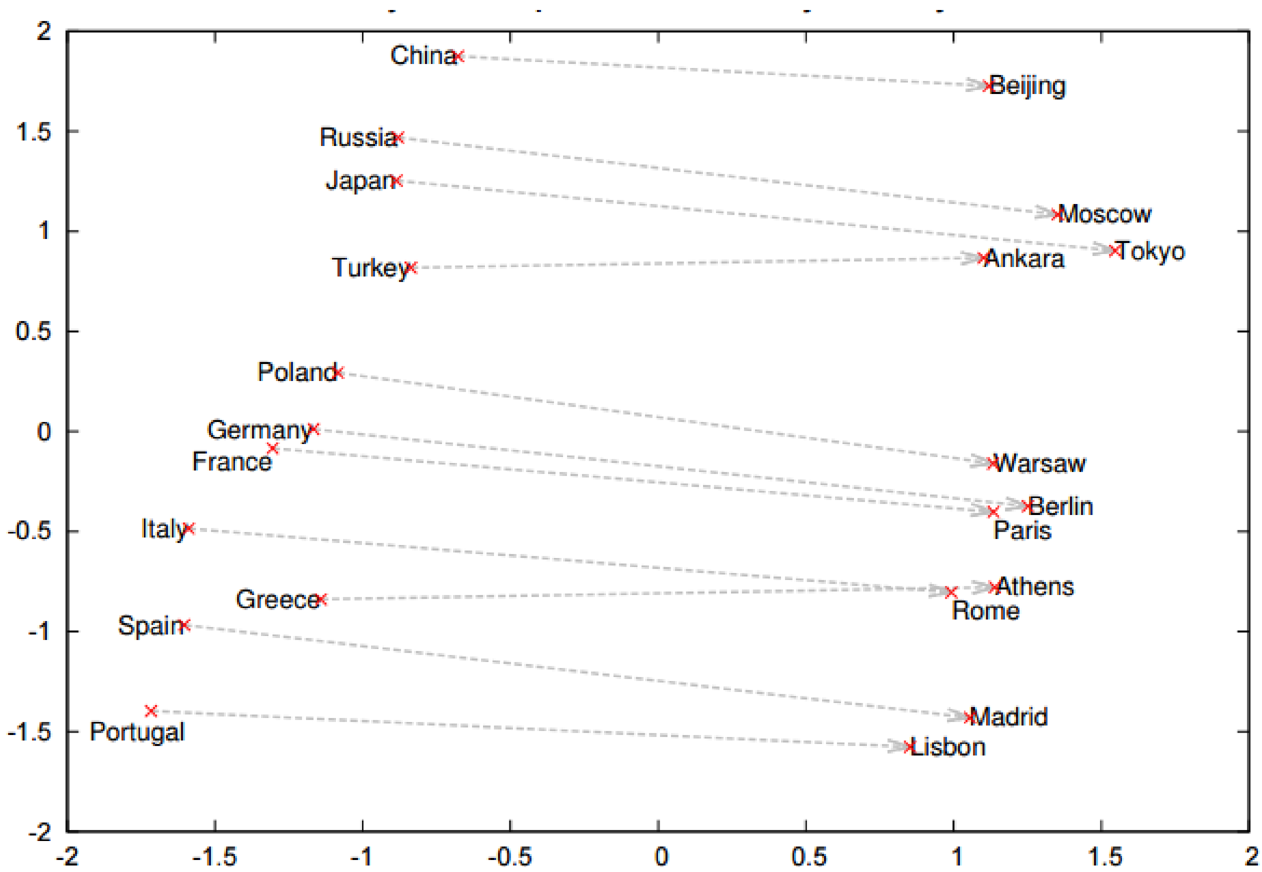
Vector space models of words

- While learning these word representations, we are actually building a vector space in which all words reside with certain relationships between them
- Encodes both syntactic and semantic relationships
- This vector space allows for algebraic operations:

$$\mathrm{Vec}(\mathrm{king}) - \mathrm{vec}(\mathrm{man}) + \mathrm{vec}(\mathrm{woman}) \approx \mathrm{vec}(\mathrm{queen})$$

Why linear algebra is working?

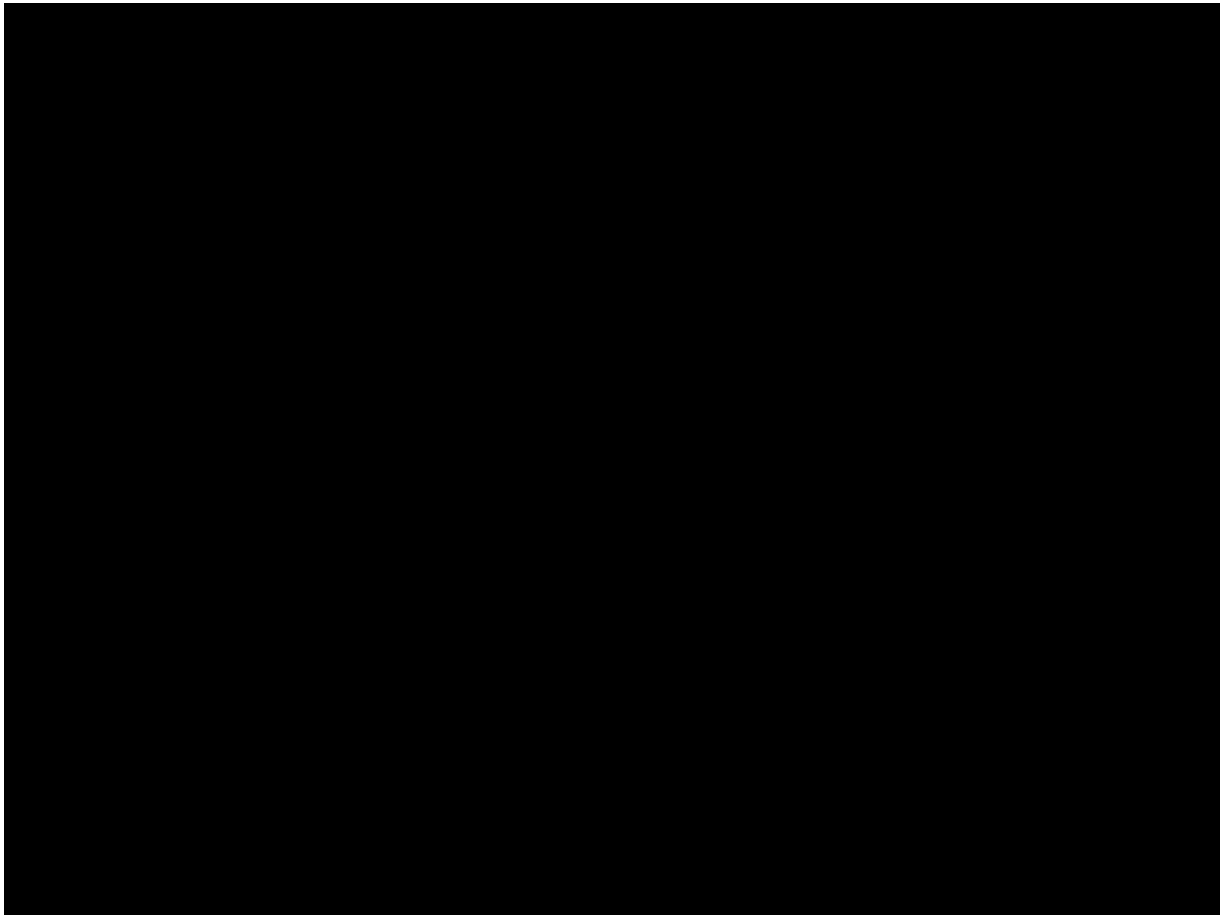
Vector space models of words: semantic relationships



Trained on the Google news corpus with over 300 billion words

Language Models

Example of Language Model



Transcription

- **Couric:** You've cited Alaska's proximity to Russia as part of your foreign policy experience. What did you mean by that?
- **Sarah Palin:** That Alaska has a very narrow maritime border between a foreign country, Russia, and, on our other side, the land-boundary that we have with Canada. It's funny that a comment like that was kinda made to … I don't know, you know … reporters.
- **Couric:** Mocked?
- **Palin:** Yeah, mocked, I guess that's the word, yeah.
- **Couric:** Well, explain to me why that enhances your foreign-policy credentials.

Language Models: N-Grams

- Estimate probability of each word given prior context.
  - P(mock | that was kinda made to)
- An N-gram model uses only N–1 words of prior context.
  - Unigram: P(mock)
  - Bigram: P(mock | to)
  - Trigram: P(mock | made to)
- The *Markov assumption* is the presumption that the future behavior of a dynamical system only depends on its recent history.

N-Gram Model Formulas

- Word sequences

$$w_1^n = w_1 \dots w_n$$

- Chain rule of probability

$$P(w_1^n) = P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1^2) \dots P(w_n \mid w_1^{n-1}) = \prod_{k=1}^{n} P(w_k \mid w_1^{k-1})$$

- Bigram approximation

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k \mid w_{k-1})$$

- N-gram approximation

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k \mid w_{k-N+1}^{k-1})$$

Estimating Probabilities

- N-gram conditional probabilities can be estimated from raw text based on the *relative frequency* of word sequences.

Bigram: $$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

N-gram: $$P(w_n \mid w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1}w_n)}{C(w_{n-N+1}^{n-1})}$$

Application: Speech Recognition

$$\underset{wordsequence}{\arg\max}\ P(wordsequence \mid acoustics) =$$

$$\underset{wordsequence}{\arg\max}\ \frac{P(acoustics \mid wordsequence) \times P(wordsequence)}{P(acoustics)}$$

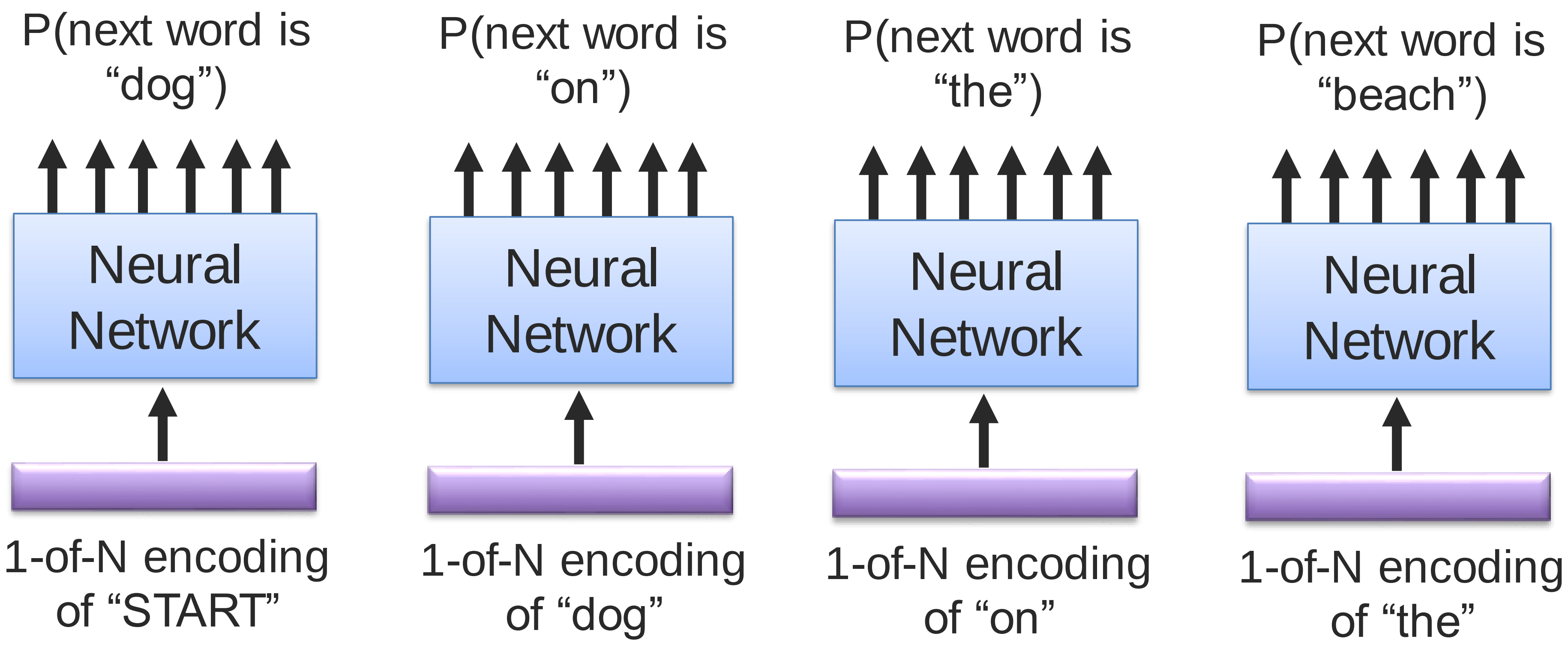
$$\underset{wordsequence}{\arg\max}\ P(acoustics \mid wordsequence) \times \underbrace{P(wordsequence)}_{\text{Language model}}$$

Neural-based Unigram Language Model (LM)

P("dog on the beach")

=P(dog|START)P(on|dog)P(the|on)P(beach|the)

P(b|a): not from count, but the NN that can predict the next word.

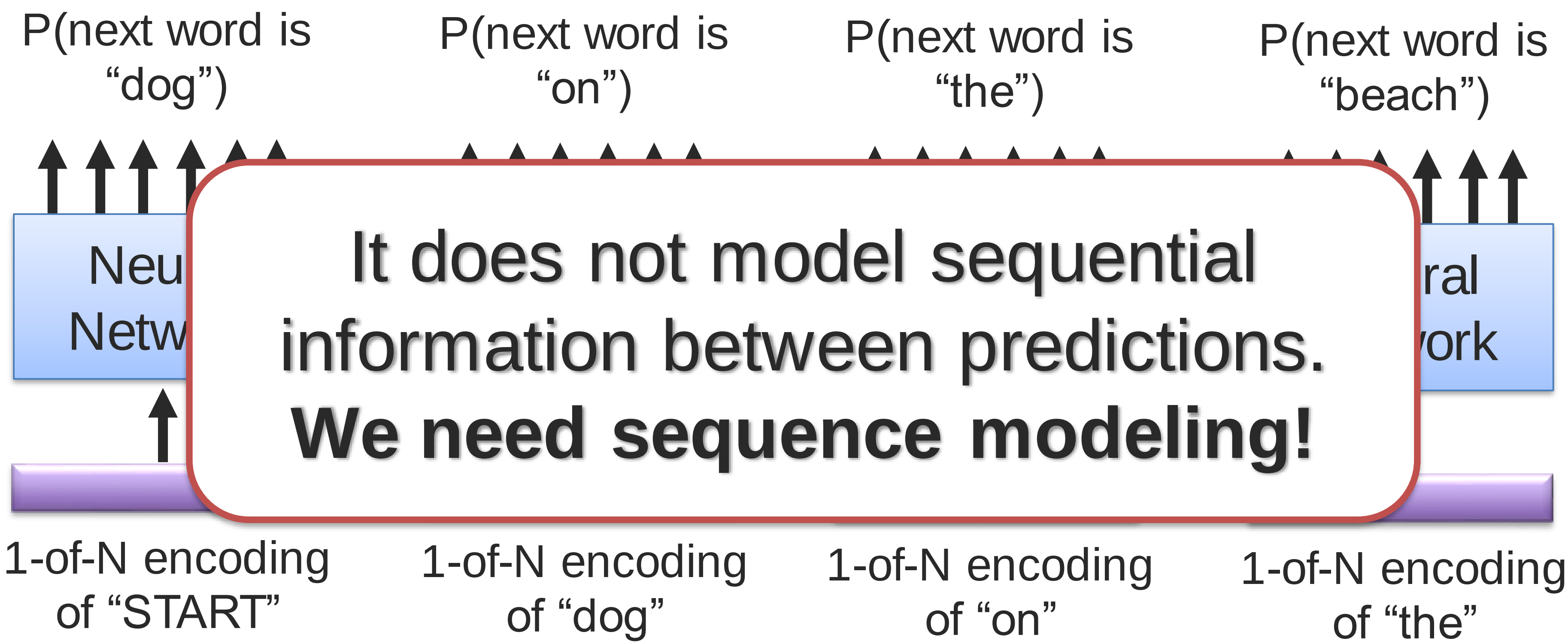


Neural-based Unigram Language Model (LM)

P("dog on the beach")

=P(dog|START)P(on|dog)P(the|on)P(beach|the)

P(b|a): not from count, but the NN that can predict the next word.



It does not model sequential information between predictions.

We need sequence modeling!

Sequence Modeling Tasks

Sequence Modeling: Language Model

Masterful!

By Antony Witheyman - January 12, 2006

Ideal for anyone with an interest in disguises who likes to see the subject tackled in a humourous manner.

0 of 4 people found this review helpful

Prediction → Language Model

Next word?

Ideal for anyone with an interest in disguises

Sequence Modeling: Sequence Prediction

Masterful!

By Antony Witheyman - January 12, 2006

Ideal for anyone with an interest in disguises who likes to see the subject tackled in a humourous manner.

0 of 4 people found this review helpful

Prediction →

Part-of-speech ?
(noun, verb,…)

POS?	POS?	POS?	POS?	POS?	POS?	POS?	POS?
↑	↑	↑	↑	↑	↑	↑	↑
Ideal	for	anyone	with	an	interest	in	disguises

Sequence Modeling: Sequence Label Prediction

Masterful!

By Antony Witheyman - January 12, 2006

Ideal for anyone with an interest in disguises who likes to see the subject tackled in a humourous manner.

0 of 4 people found this review helpful

Prediction

Sentiment ?
(positive or negative)

Sentiment label?

Ideal for anyone with an interest in disguises

Sequence Modeling: Sequence Representation

Masterful!

By Antony Witheyman - January 12, 2006

Ideal for anyone with an interest in disguises who likes to see the subject tackled in a humourous manner.

0 of 4 people found this review helpful

Learning → Sequence representation

[0,1; 0,0004; 0;….; 0,01; 0.09; 0,05]

Ideal for anyone with an interest in disguises

Sequence Modeling

★★★★★ **Masterful!**

By Antony Witheyman - January 12, 2006

Ideal for anyone with an interest in disguises who likes to see the subject tackled in a humourous manner.

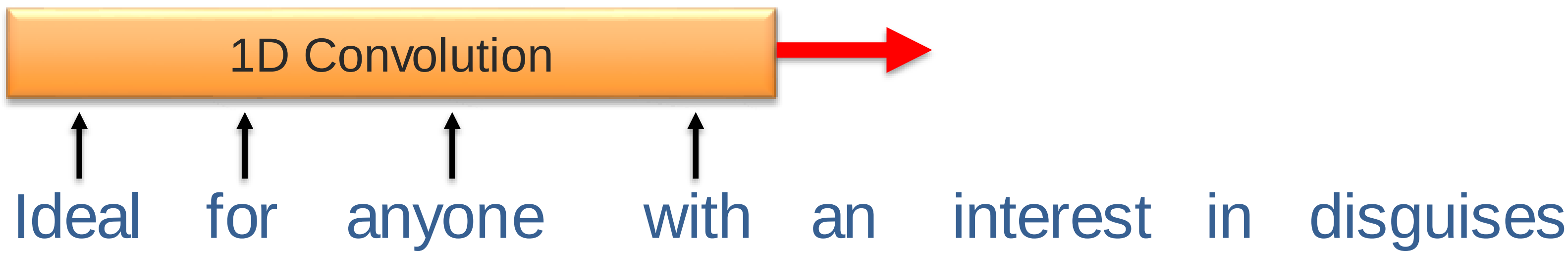
0 of 4 people found this review helpful

- Part-of-speech ?
  (noun, verb,…)
- Sentiment ?
  (positive or negative)
- Language Model
- Sequence representation

Main Challenges:

- Sequences of variable lengths (e.g., sentences)
- Keep the number of parameters at a minimum
- Take advantage of possible redundancy

Time-Delay Neural Network



Main Challenges:

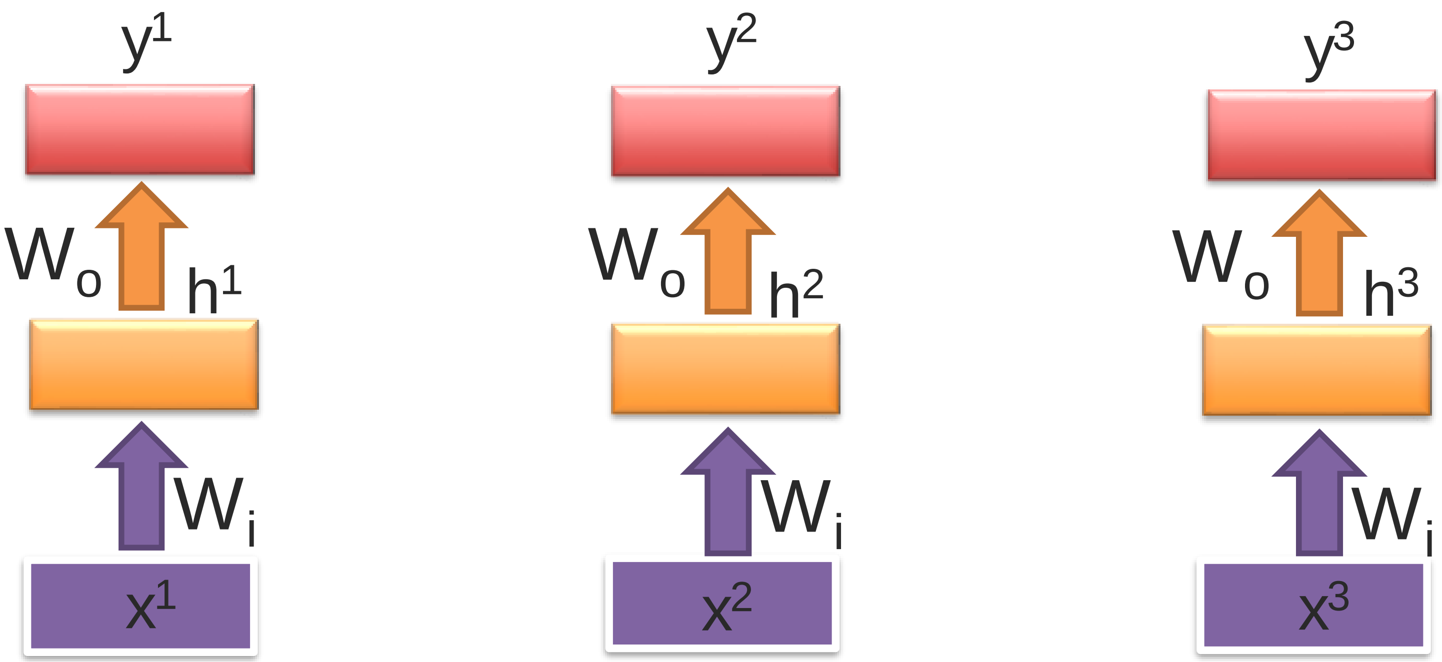
- Sequences of variable lengths (e.g., sentences)
- Keep the number of parameters at a minimum
- Take advantage of possible redundancy

Recurrent Neural Networks

Sequence Prediction (or Unigram Language Model)

Input data: $x^1 \quad x^2 \quad x^3 \quad \ldots\ldots$ (x^i are vectors)

Output data: $y^1 \quad y^2 \quad y^3 \quad \ldots\ldots$ (y^i are vectors)



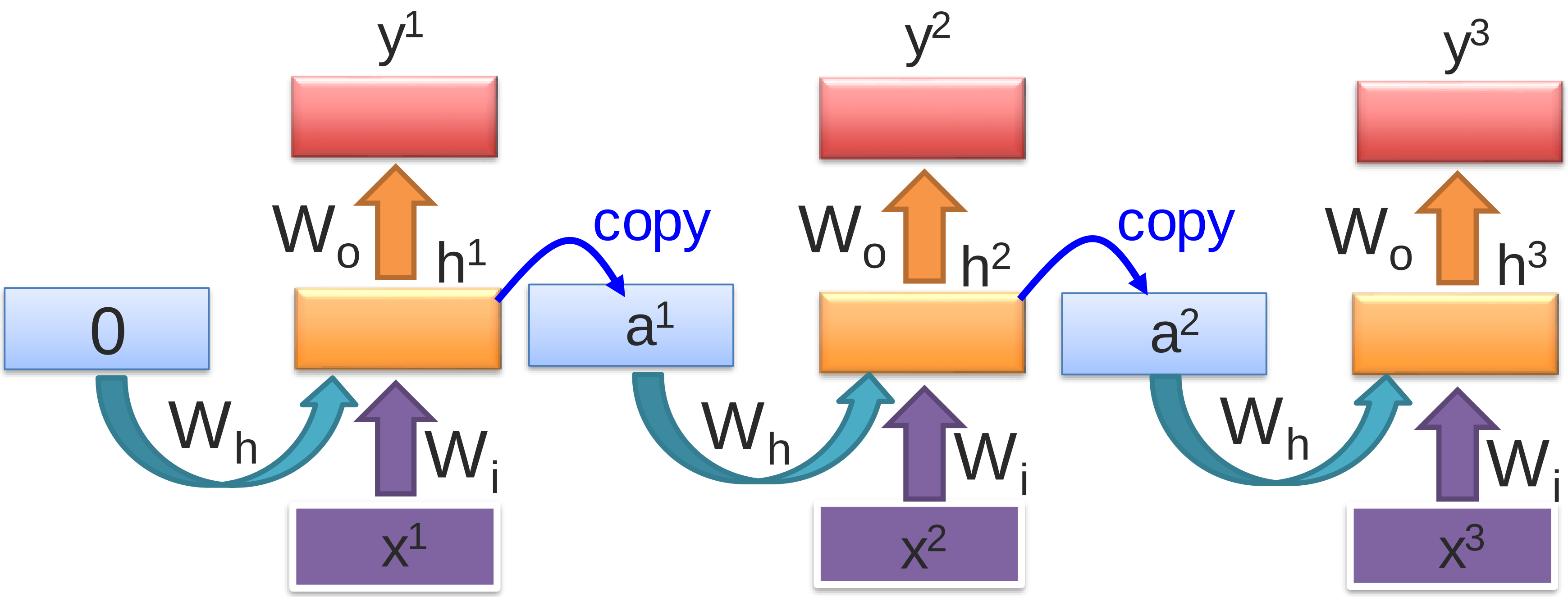
How can we include temporal dynamics?

Elman Network for Sequence Prediction

(or Unigram Language Model)

Input data: x^1 x^2 x^3 …… (x^i are vectors)

Output data: y^1 y^2 y^3 …… (y^i are vectors)

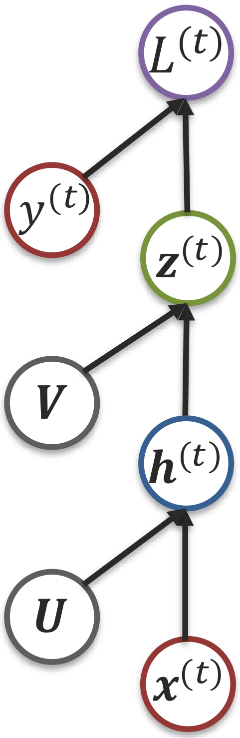


The same model parameters are used again and again.

Can be trained using backpropagation

Recurrent Neural Network

Feedforward Neural Network



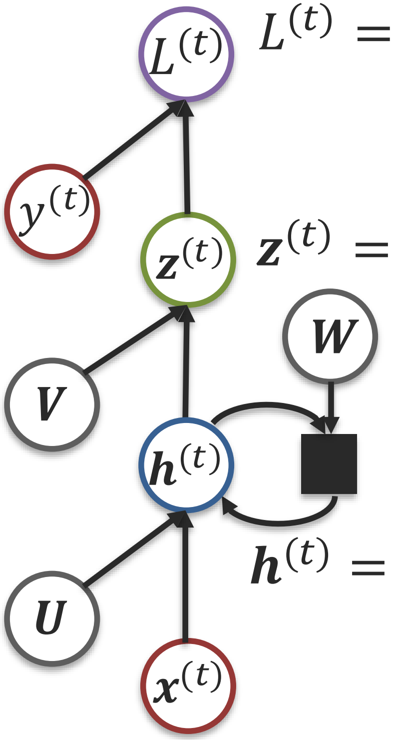
$$L^{(t)} = -logP(Y = y^{(t)} | \boldsymbol{z}^{(t)})$$

$$\boldsymbol{z}^{(t)} = matmult(\boldsymbol{h}^{(t)}, \boldsymbol{V})$$

$$\boldsymbol{h}^{(t)} = tanh(\boldsymbol{U}\boldsymbol{x}^{(t)})$$

Recurrent Neural Networks

$$L = \sum_t L^{(t)}$$



$$L^{(t)} = -logP(Y = y^{(t)} | \boldsymbol{z}^{(t)})$$

$$\boldsymbol{z}^{(t)} = matmult(\boldsymbol{h}^{(t)}, \boldsymbol{V})$$

$$\boldsymbol{h}^{(t)} = tanh(\boldsymbol{U}\boldsymbol{x}^{(t)} + \boldsymbol{W}\boldsymbol{h}^{(t-1)})$$

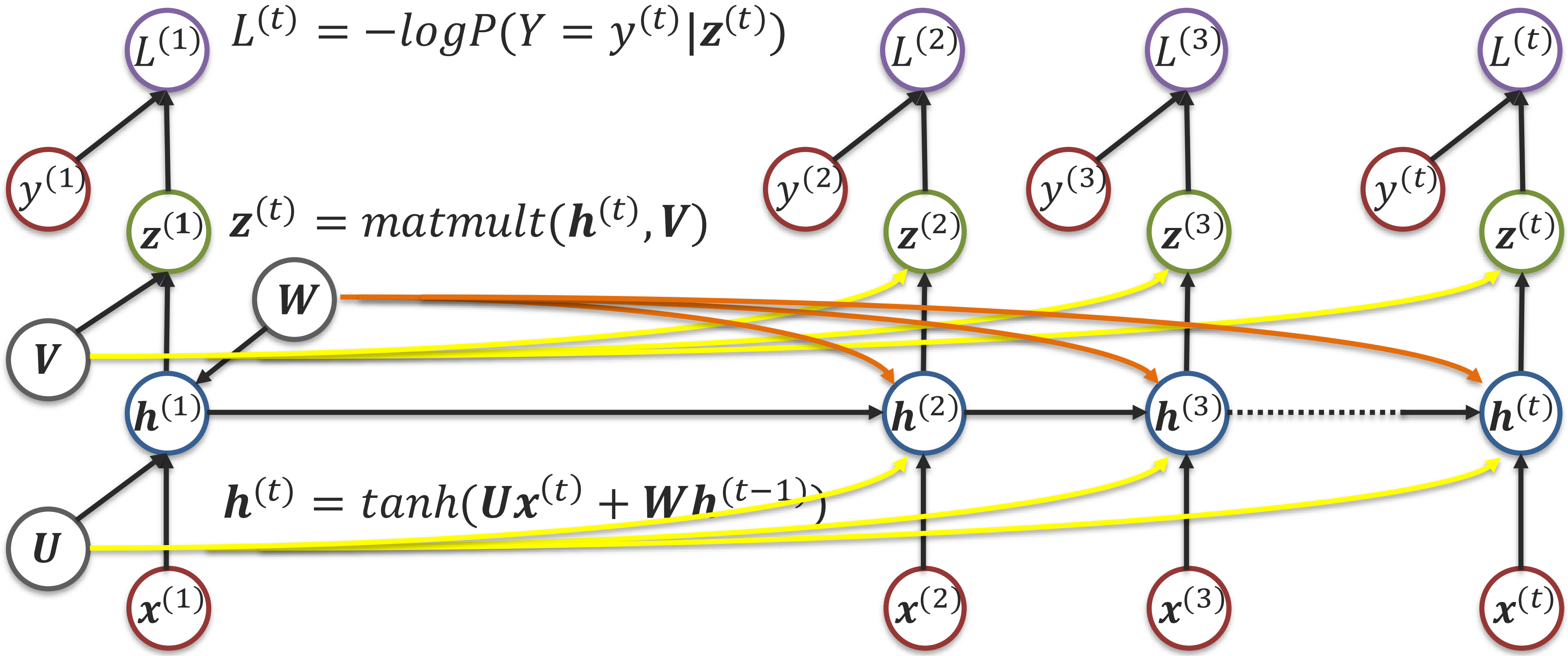
Recurrent Neural Networks - Unrolling

$$L = \sum_t L^{(t)}$$

$$L^{(t)} = -logP(Y = y^{(t)} | \boldsymbol{z}^{(t)})$$

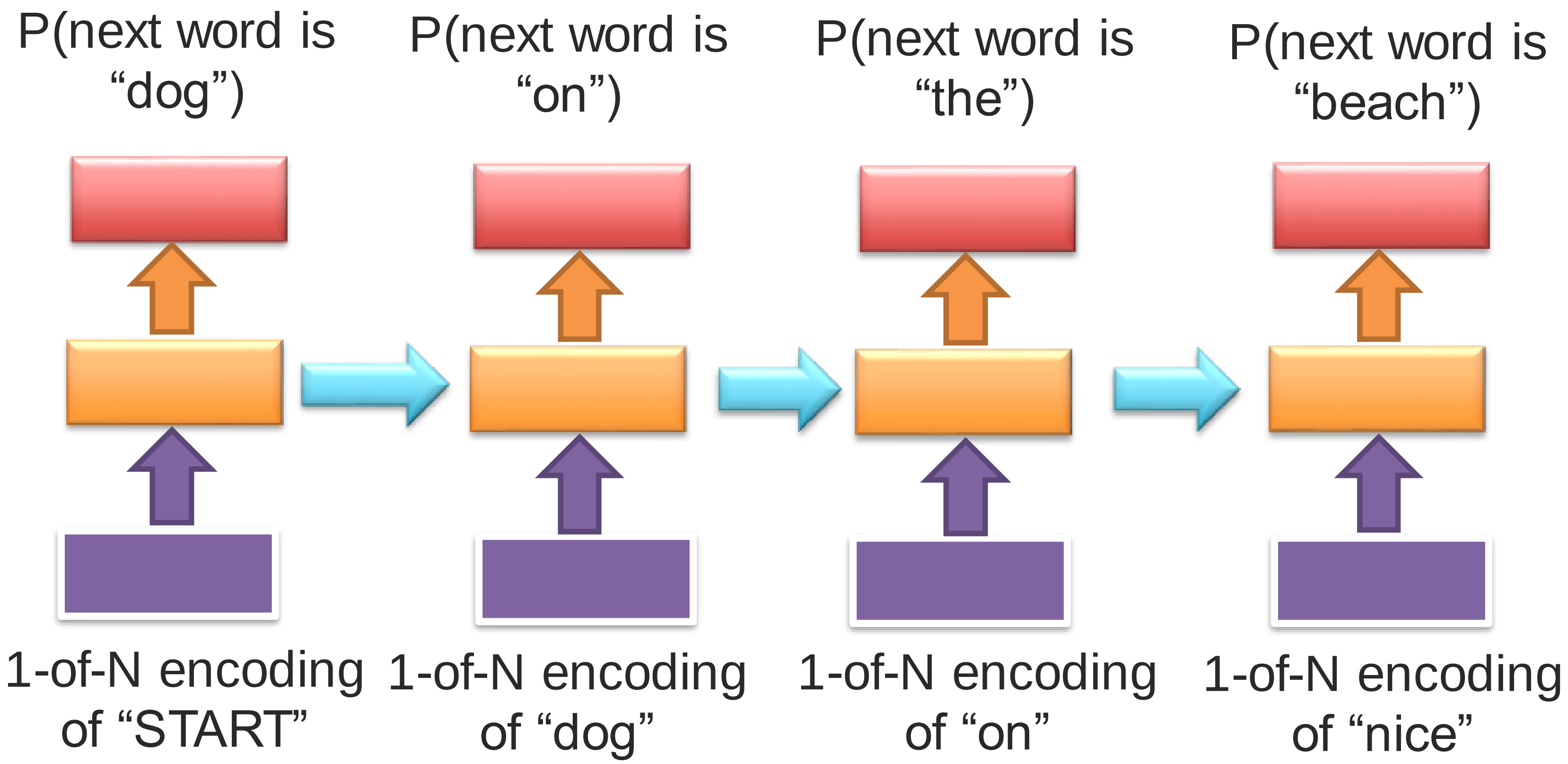
$$\boldsymbol{z}^{(t)} = matmult(\boldsymbol{h}^{(t)}, \boldsymbol{V})$$

$$\boldsymbol{h}^{(t)} = tanh(\boldsymbol{U}\boldsymbol{x}^{(t)} + \boldsymbol{W}\boldsymbol{h}^{(t-1)})$$



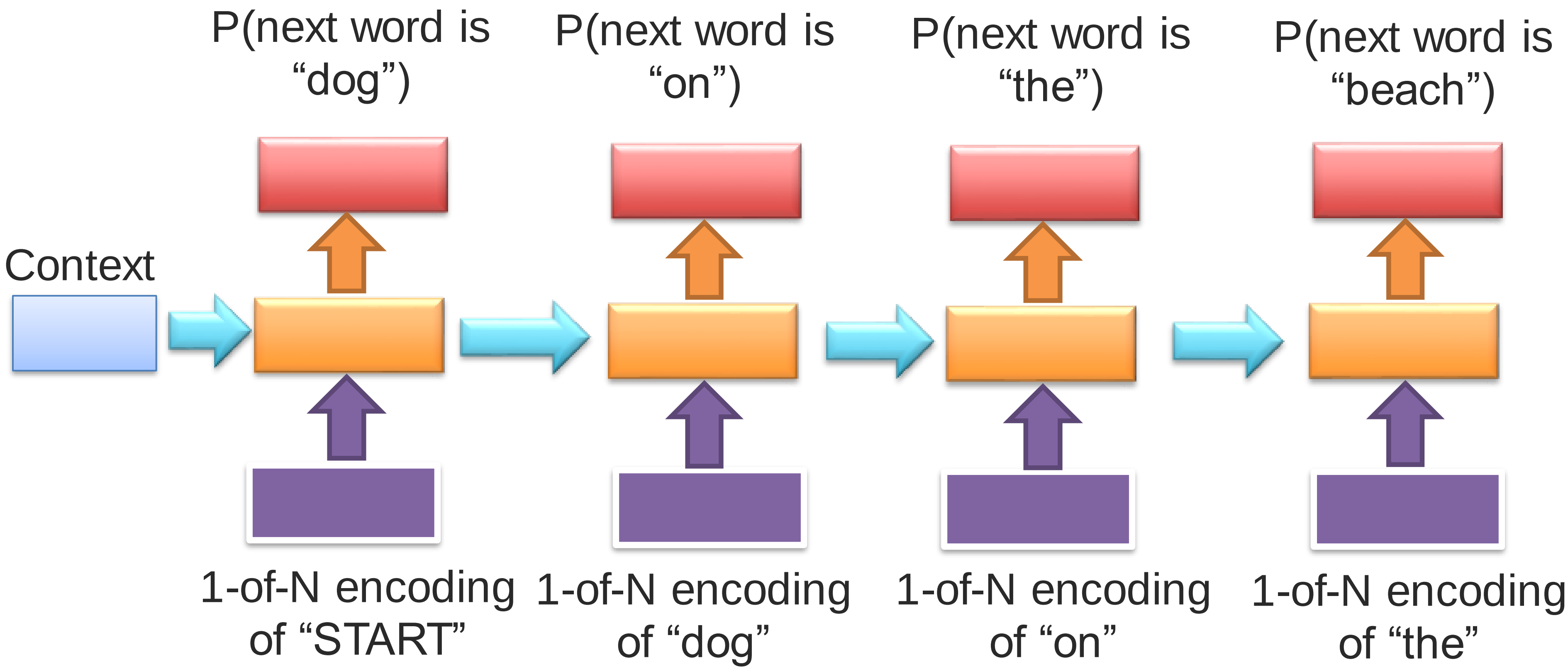
Same model parameters are used for all time parts.

RNN-based Language Model



- Models long-term information

RNN-based Sentence Generation (Decoder)



➢ Models long-term information

Sequence Modeling: Sequence Prediction

Masterful!

By Antony Witheyman - January 12, 2006

Ideal for anyone with an interest in disguises who likes to see the subject tackled in a humourous manner.

0 of 4 people found this review helpful

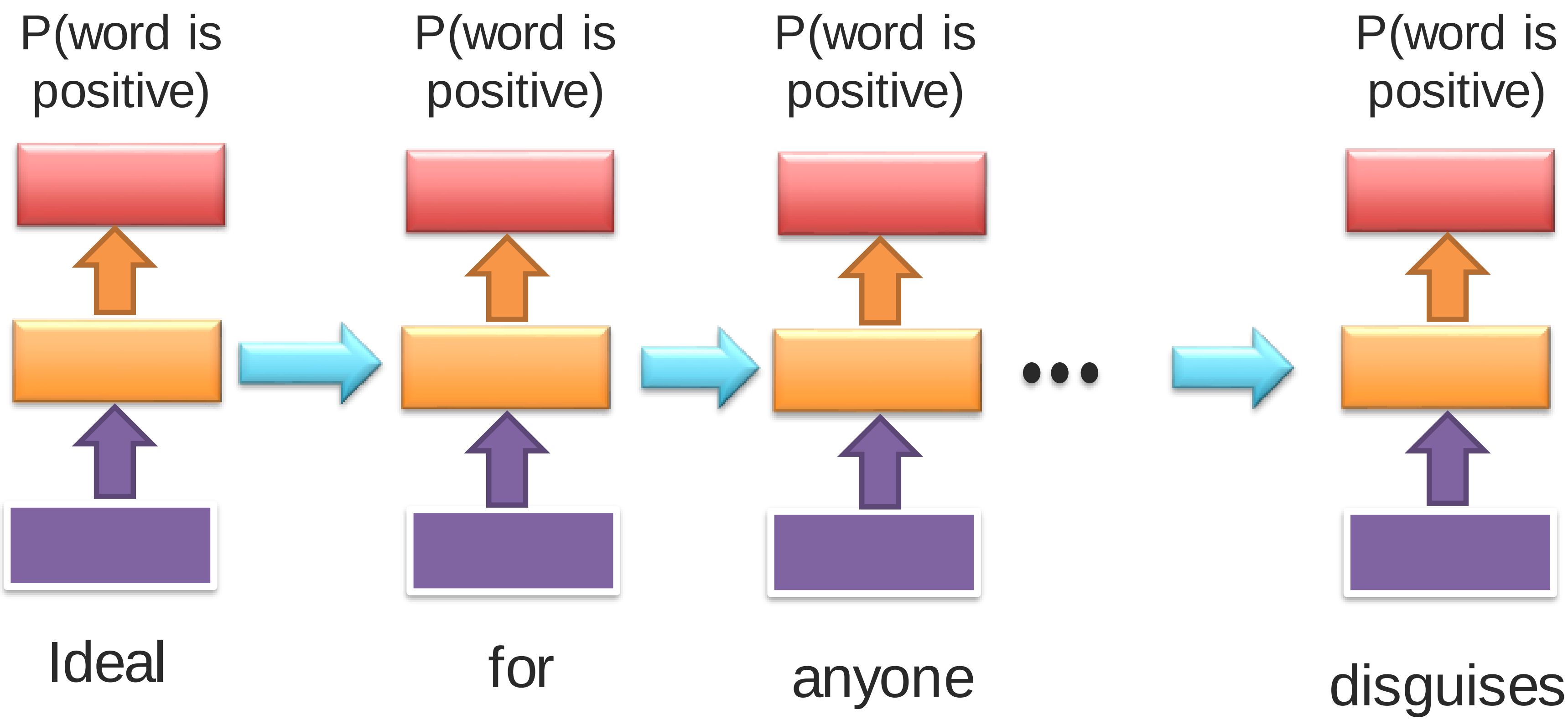
Prediction

Sentiment ?
(positive or negative)

Sentiment label?

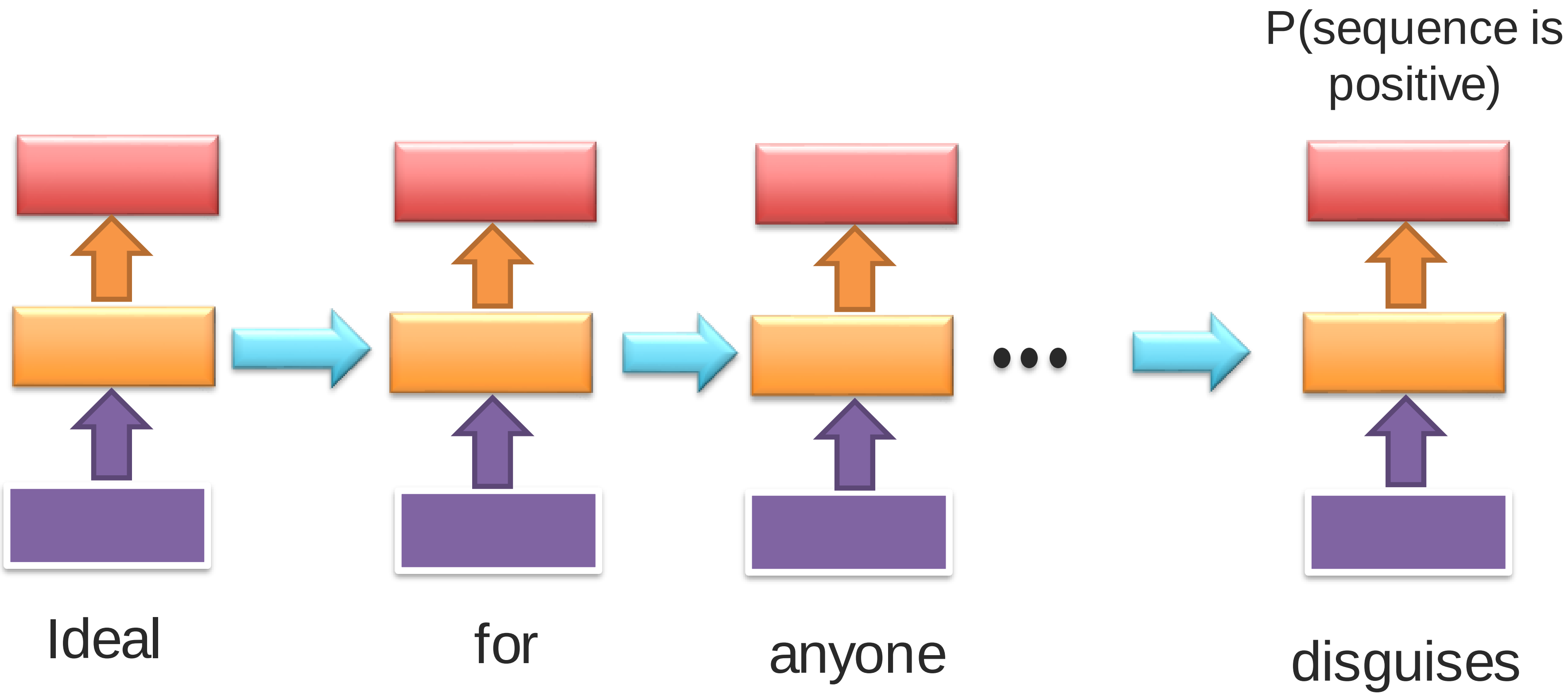
Ideal for anyone with an interest in disguises

RNN for Sequence Prediction



$$L = \sum_{t} L^{(t)} = \sum_{t} -logP(Y = y^{(t)} | \mathbf{z}^{(t)})$$

RNN for Sequence Prediction



$$L = L^{(N)} = -logP(Y = y^{(N)}|\boldsymbol{z}^{(N)})$$

Sequence Modeling: Sequence Representation

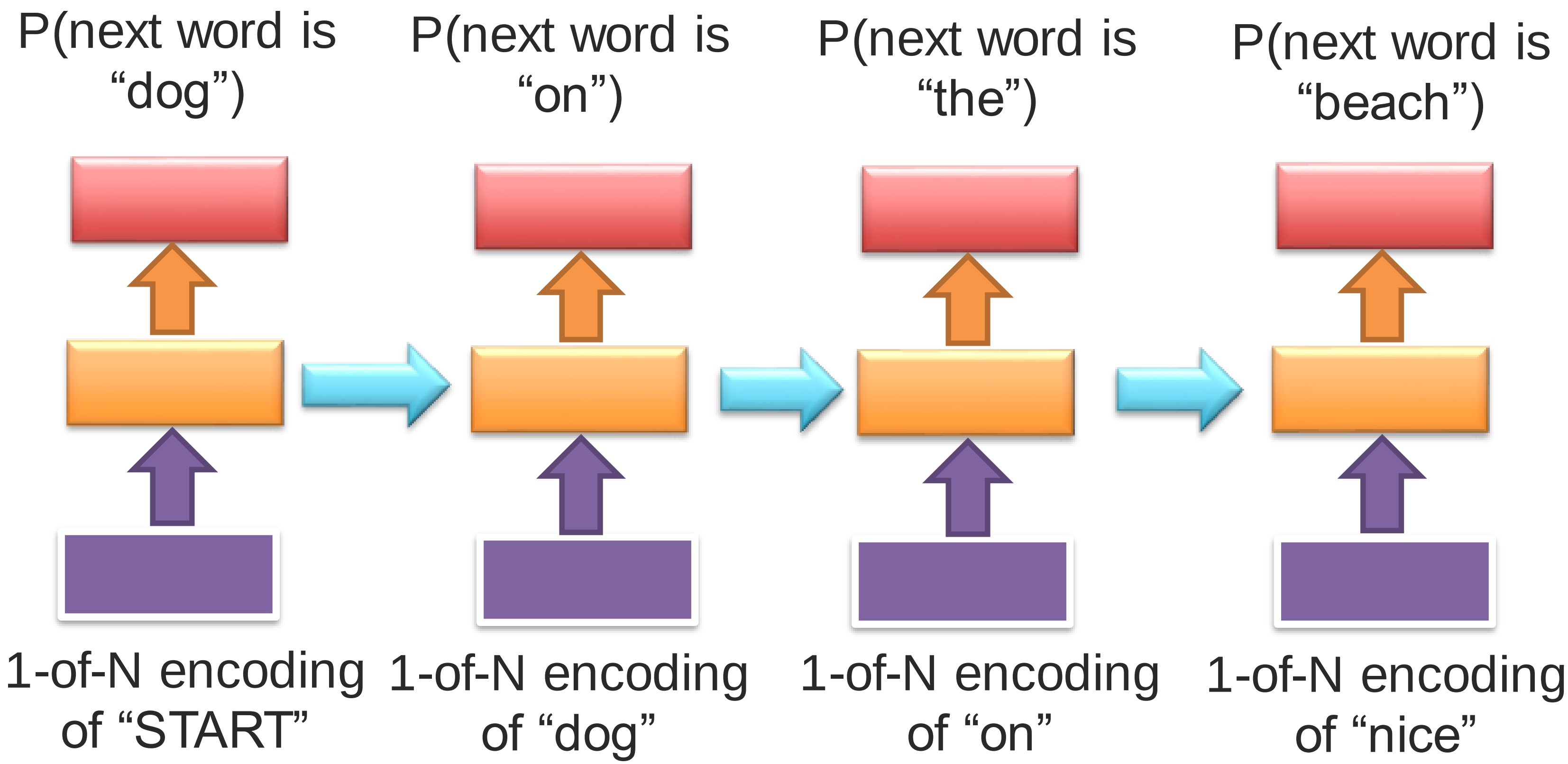

Masterful!
By Antony Witheyman - January 12, 2006
Ideal for anyone with an interest in disguises who likes to see the subject tackled in a humourous manner.
0 of 4 people found this review helpful

Learning → Sequence representation

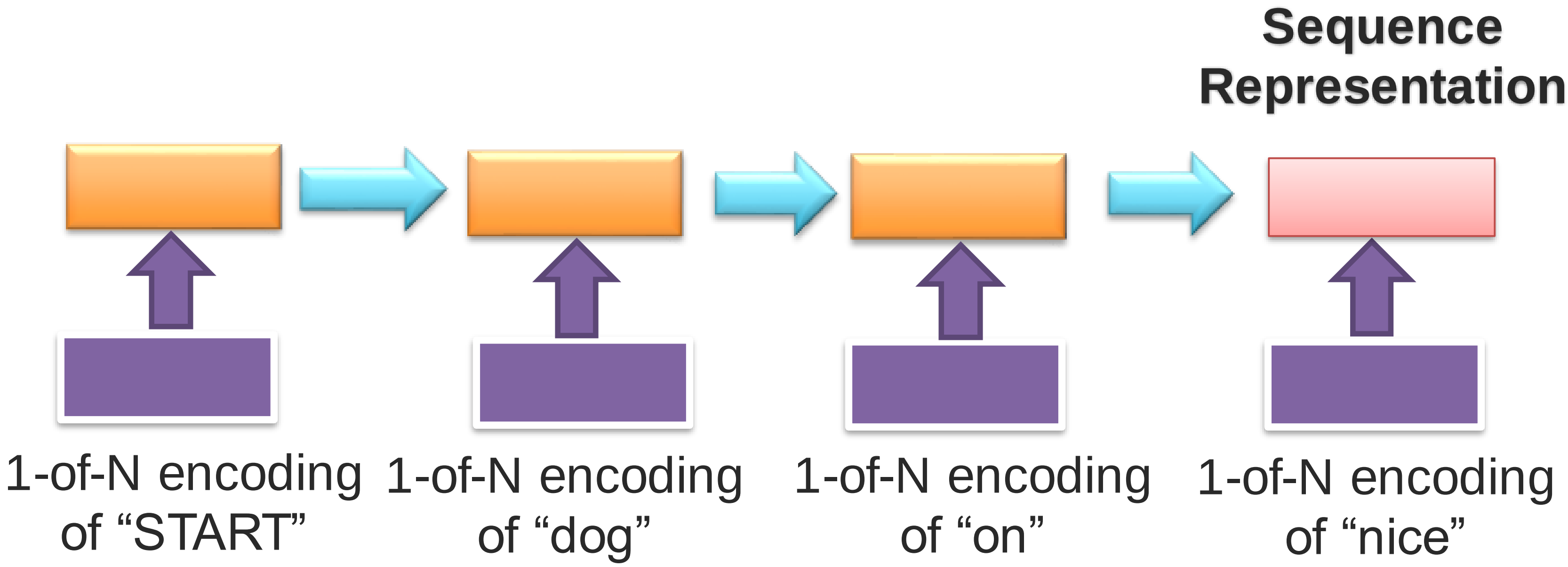
[0,1; 0,0004; 0;….; 0,01; 0.09; 0,05]

Ideal for anyone with an interest in disguises

RNN for Sequence Representation

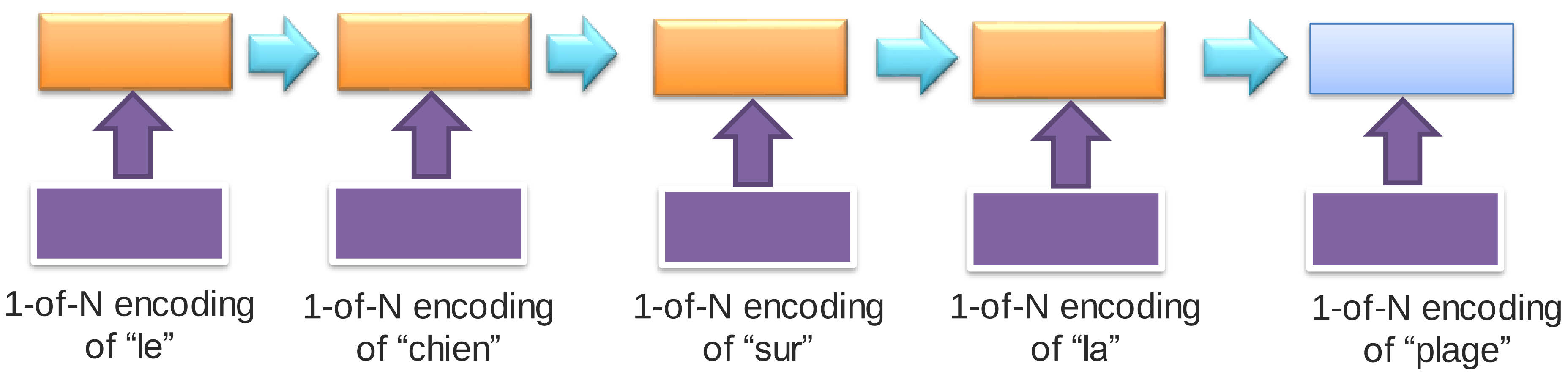


RNN for Sequence Representation (Encoder)

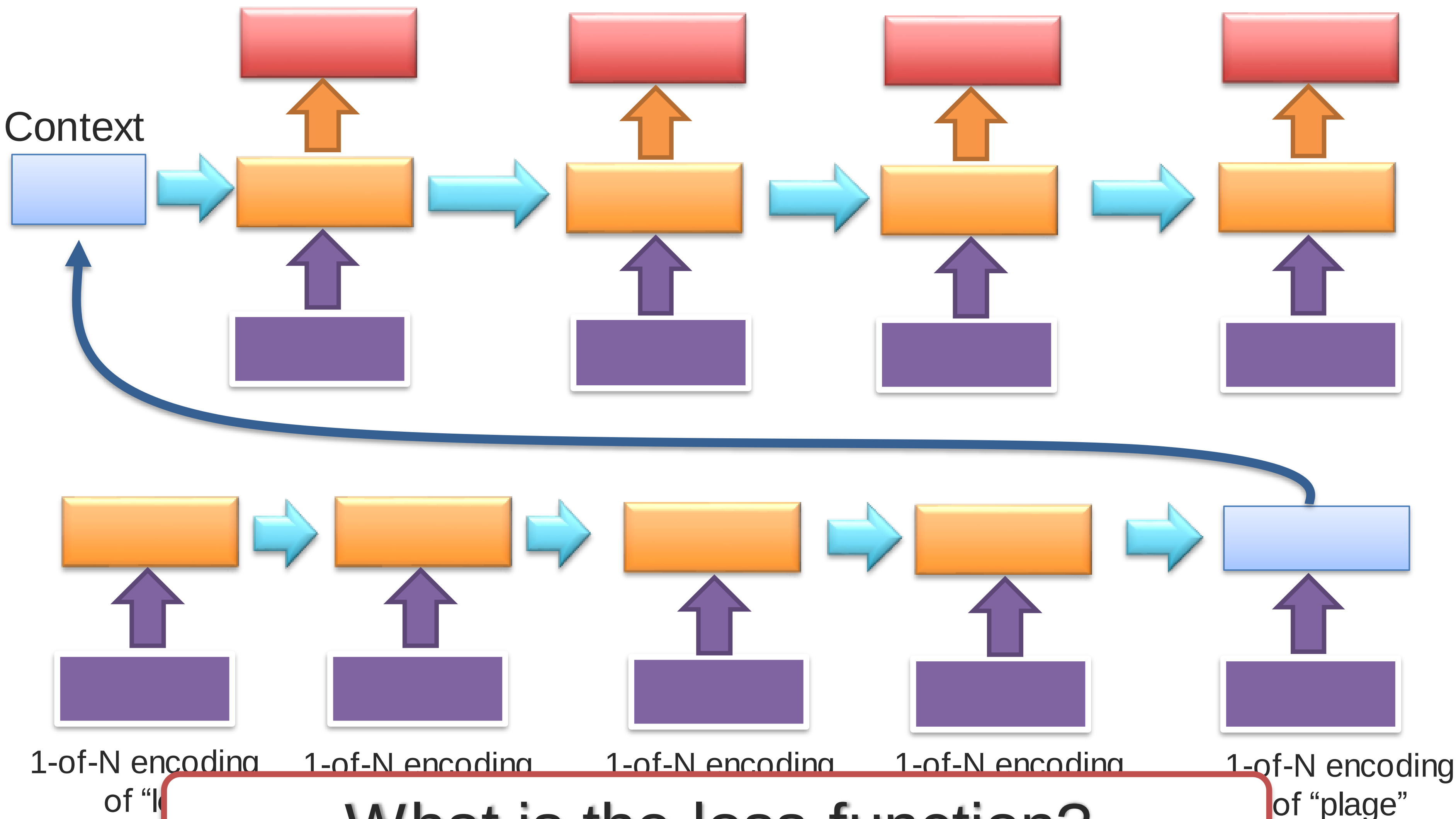


RNN-based for Machine Translation

Le chien sur la plage → The dog on the beach



Encoder-Decoder Architecture



What is the loss function?

Advanced Topics

- Character-level “language models”
  - Xiang Zhang, Junbo Zhao and Yann LeCun, Character-level Convolutional Networks for Text Classification, NIPS 2015

  http://arxiv.org/pdf/1509.01626v2.pdf

- Skip-though: embedding at the sentence level
  - Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, Sanja Fidler. Skip-Thought Vectors, NIPS 2015

  http://arxiv.org/pdf/1506.06726v1.pdf

Backpropagation Through Time

Optimization: Gradient Computation

Vector representation:

$$\nabla_{\boldsymbol{x}}\, y = \left[\frac{\partial y}{\partial x_1}, \frac{\partial y}{\partial x_2}, \frac{\partial y}{\partial x_3}\right]$$

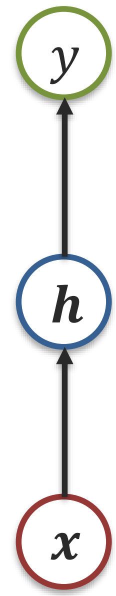
Gradient

$$\nabla_{\boldsymbol{x}}\, y = \left(\boxed{\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}}\right)^T \nabla_{\boldsymbol{h}}\, y$$

"backprop" Gradient

"local" Jacobian

(matrix of size $|h| \times |x|$ computed using partial derivatives)



$y = f(\boldsymbol{h})$

$\boldsymbol{h} = g(\boldsymbol{x})$

Backpropagation Algorithm

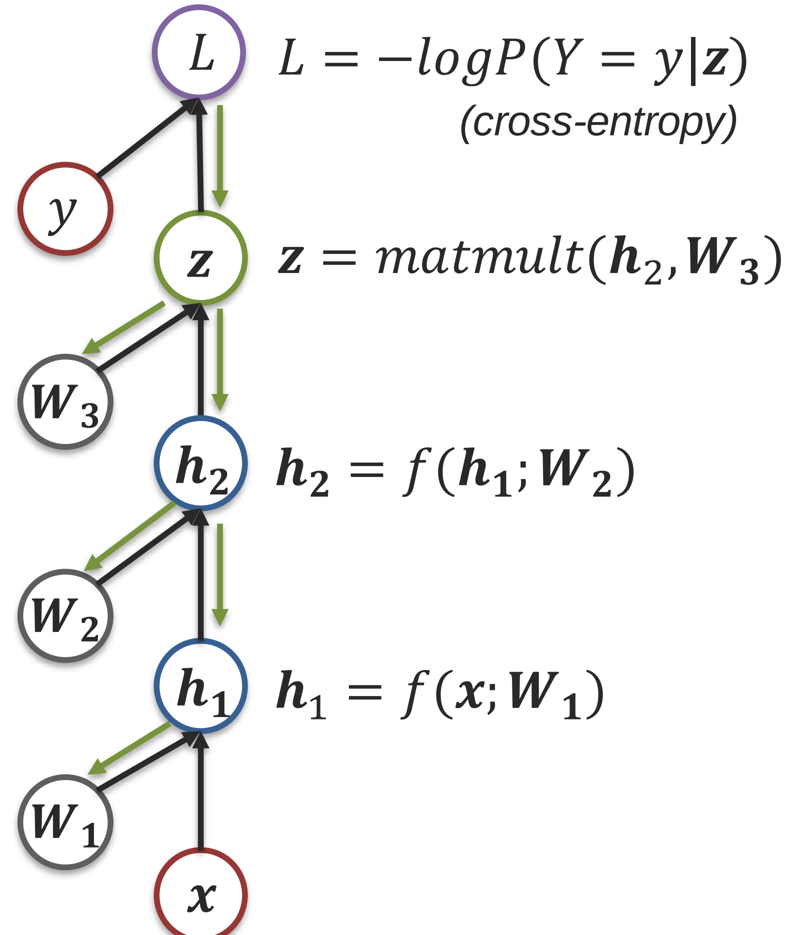
Forward pass

- Following the graph topology, compute value of each unit

Backpropagation pass

- Initialize output gradient = 1
- Compute "local" Jacobian matrix using values from forward pass
- Use the chain rule:

Gradient = "local" Jacobian x "backprop" gradient



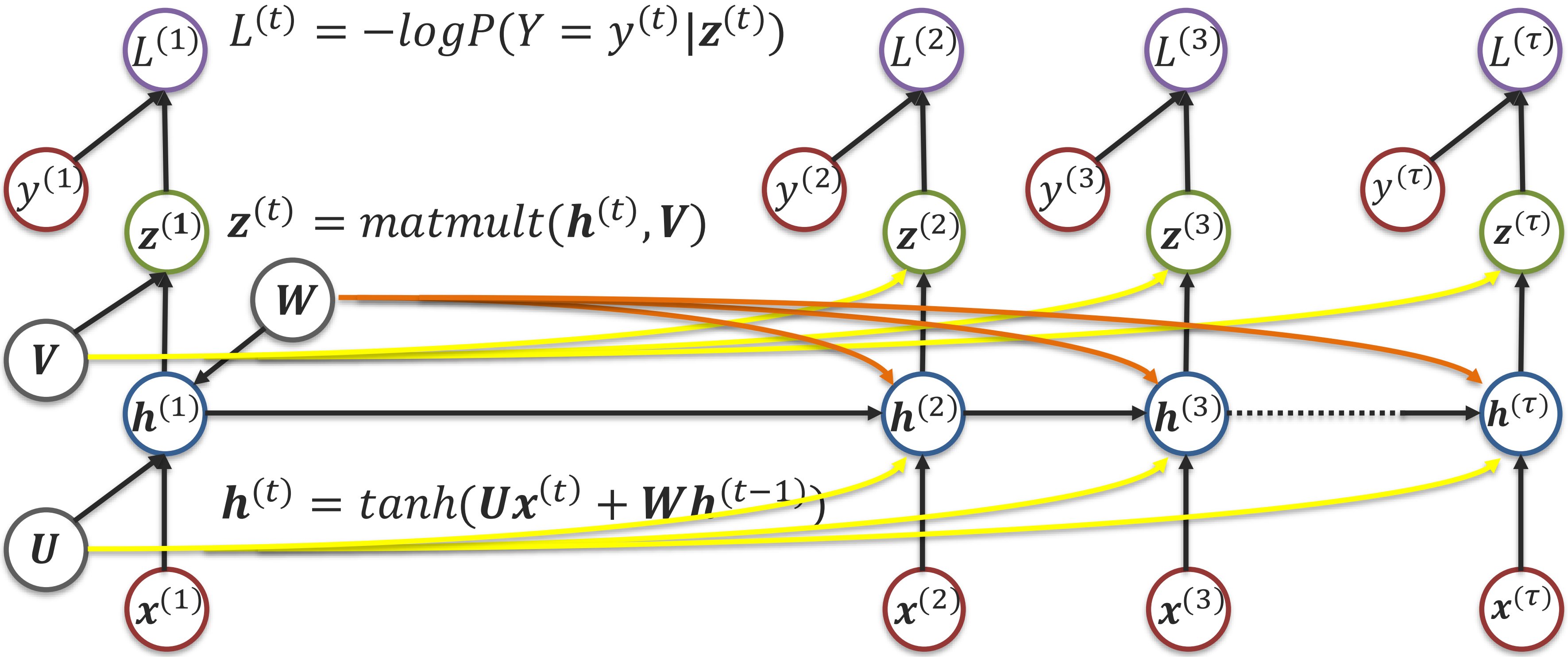
Recurrent Neural Networks

$$L = \sum_t L^{(t)}$$

$$L^{(t)} = -logP(Y = y^{(t)} | \boldsymbol{z}^{(t)})$$

$$\boldsymbol{z}^{(t)} = matmult(\boldsymbol{h}^{(t)}, \boldsymbol{V})$$

$$\boldsymbol{h}^{(t)} = tanh(\boldsymbol{U}\boldsymbol{x}^{(t)} + \boldsymbol{W}\boldsymbol{h}^{(t-1)})$$



Backpropagation Through Time

$$L = \sum_t L^{(t)} = -\sum_t logP(Y = y^{(t)} | \boldsymbol{z}^{(t)})$$

Gradient = "backprop" gradient

x "local" Jacobian

$L^{(\tau)}$ or $L^{(t)}$

$$\frac{\partial L}{\partial L^{(t)}} = 1$$

$\boldsymbol{z}^{(\tau)}$ or $\boldsymbol{z}^{(t)}$

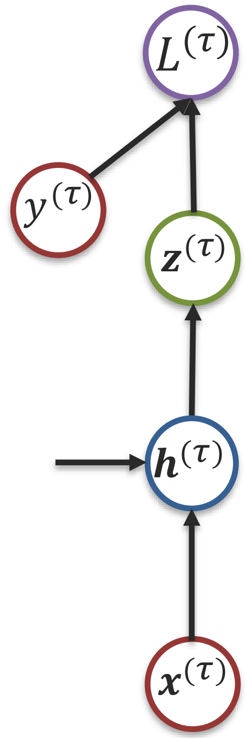
$$(\nabla_{\boldsymbol{z}^{(t)}} L)_i = \frac{\partial L}{\partial z_i^{(t)}} = \frac{\partial L}{\partial L^{(t)}} \frac{\partial L^{(t)}}{\partial z_i^{(t)}} = sigmoid(z_i^t) - \mathbf{1}_{i,y^{(t)}}$$

$\boldsymbol{h}^{(\tau)}$

$$\nabla_{\boldsymbol{h}^{(\tau)}} L = \nabla_{\boldsymbol{z}^{(\tau)}} L \frac{\partial z^{(\tau)}}{\partial \boldsymbol{h}^{(\tau)}} = \nabla_{\boldsymbol{z}^{(\tau)}} L \boldsymbol{V}$$

$\boldsymbol{h}^{(t)} \rightarrow \boldsymbol{h}^{(t+1)}$

$$\nabla_{\boldsymbol{h}^{(t)}} L = \nabla_{\boldsymbol{z}^{(t)}} L \frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{h}^{(t)}} + \nabla_{\boldsymbol{z}^{(t+1)}} L \frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}}$$



Backpropagation Through Time

$$L = \sum_t L^{(t)} = -\sum_t logP(Y = y^{(t)}|\boldsymbol{z}^{(t)})$$

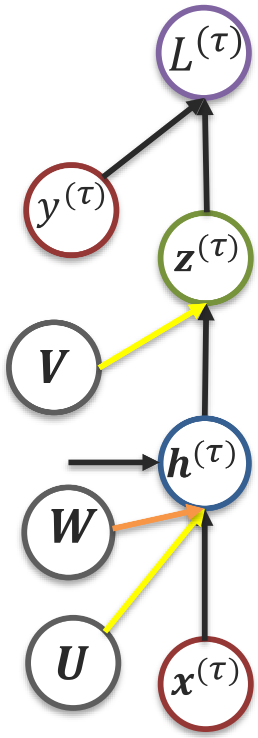
Gradient = "backprop" gradient

x "local" Jacobian

$\boldsymbol{V}$ $$\nabla_{\boldsymbol{V}} L = \sum_t (\nabla_{\boldsymbol{z}^{(t)}} L) \frac{\partial \boldsymbol{z}^{(t)}}{\partial \boldsymbol{V}}$$

$\boldsymbol{W}$ $$\nabla_{\boldsymbol{W}} L = \sum_t (\nabla_{\boldsymbol{h}^{(t)}} L) \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{W}}$$

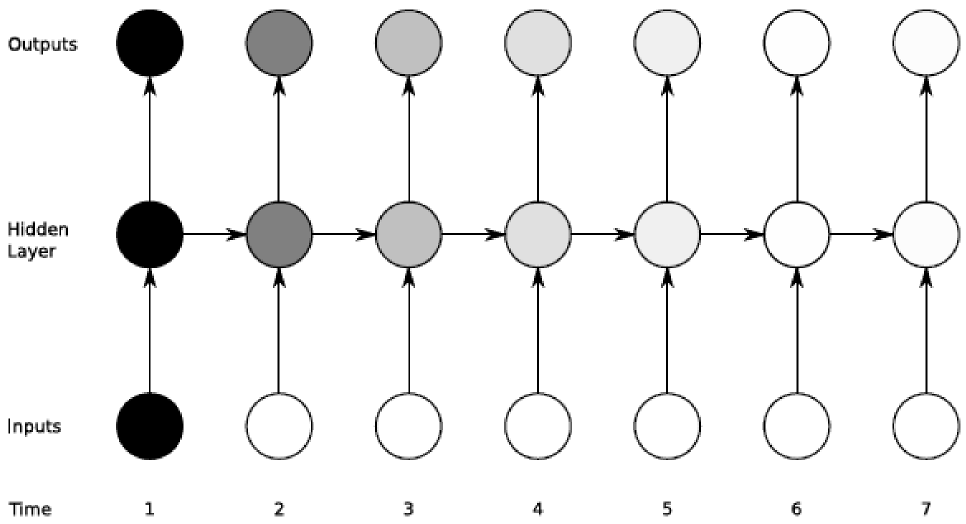
$\boldsymbol{U}$ $$\nabla_{\boldsymbol{U}} L = \sum_t (\nabla_{\boldsymbol{h}^{(t)}} L) \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{U}}$$



Long-term Dependencies

Vanishing gradient problem for RNNs:

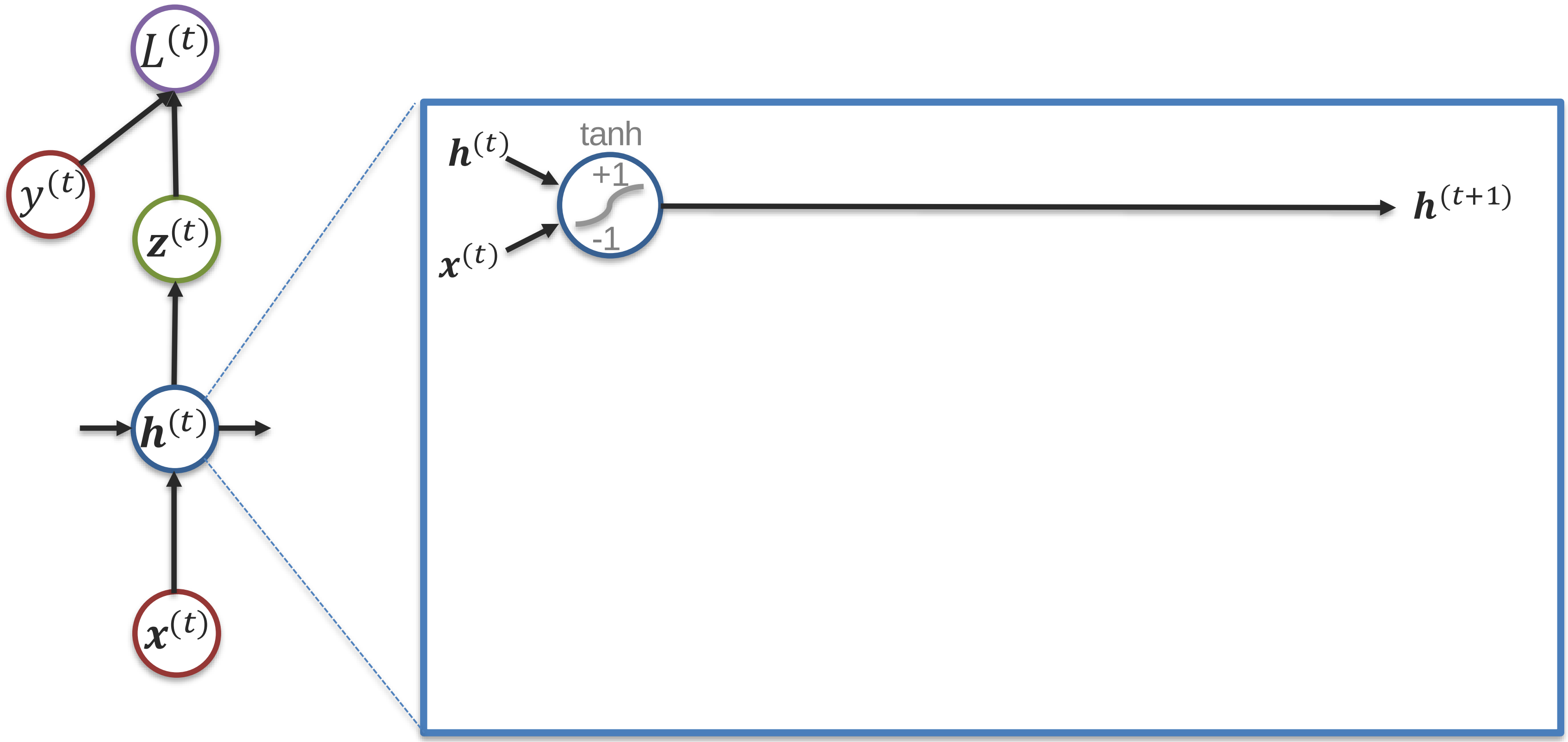
$$\boldsymbol{h}^{(t)} \sim tanh(\boldsymbol{W}\boldsymbol{h}^{(t-1)})$$



- The influence of a given input on the hidden layer, and therefore on the network output, either decays or blows up exponentially as it cycles around the network's recurrent connections.

Gated Recurrent Neural Networks

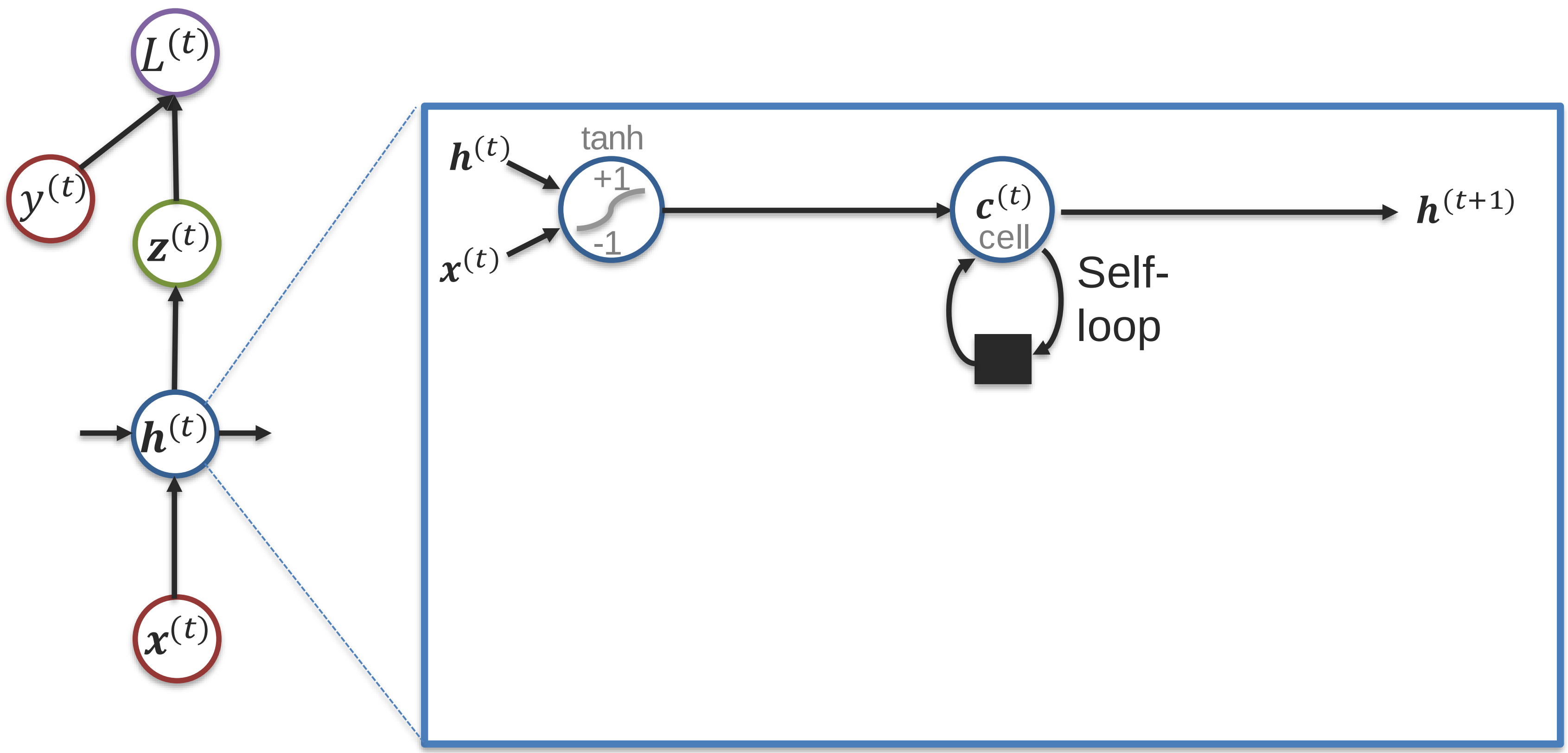
Recurrent Neural Networks



LSTM ideas: (1) "Memory" Cell and Self Loop

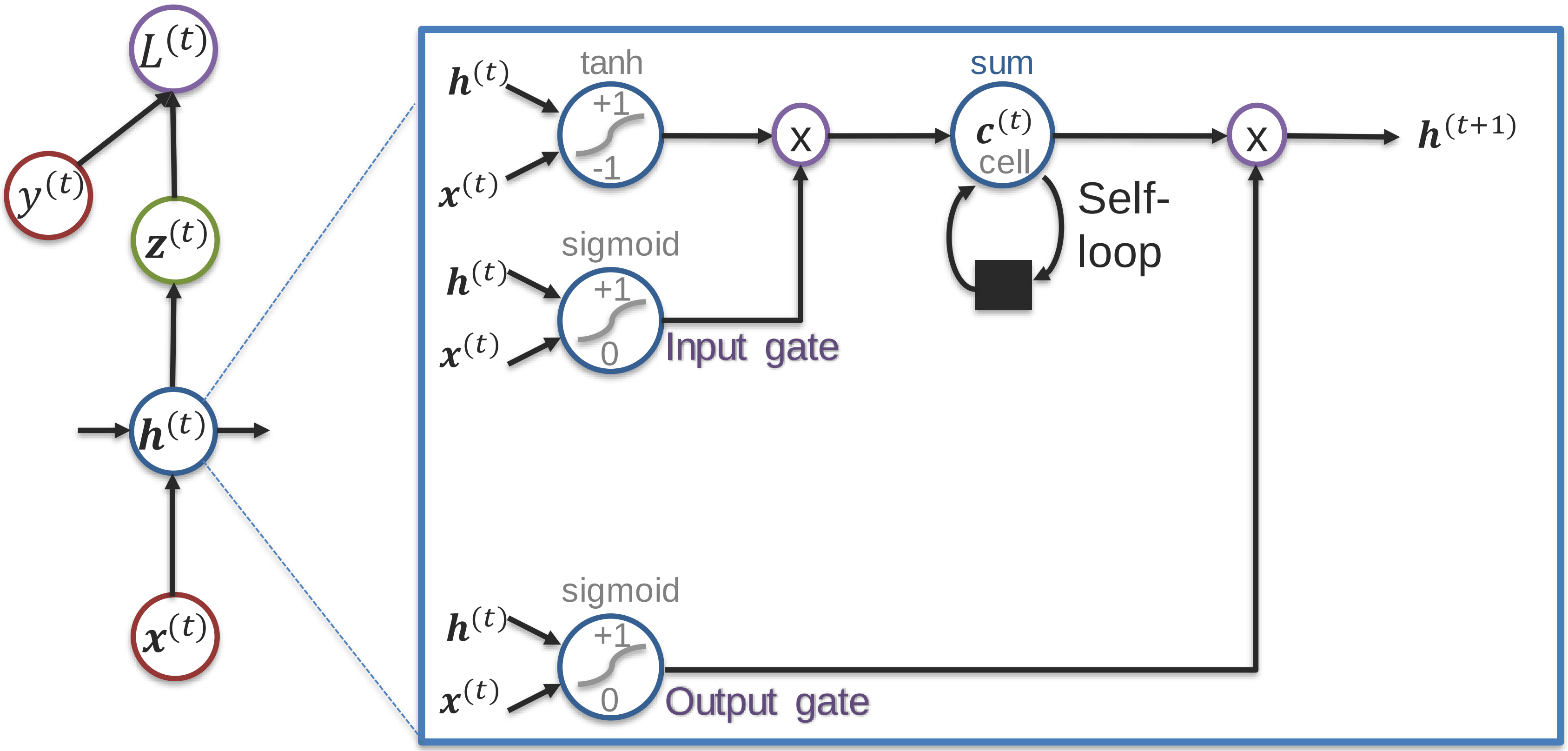
[Hochreiter and Schmidhuber, 1997]

Long Short-Term Memory (LSTM)



LSTM Ideas: (2) Input and Output Gates

[Hochreiter and Schmidhuber, 1997]

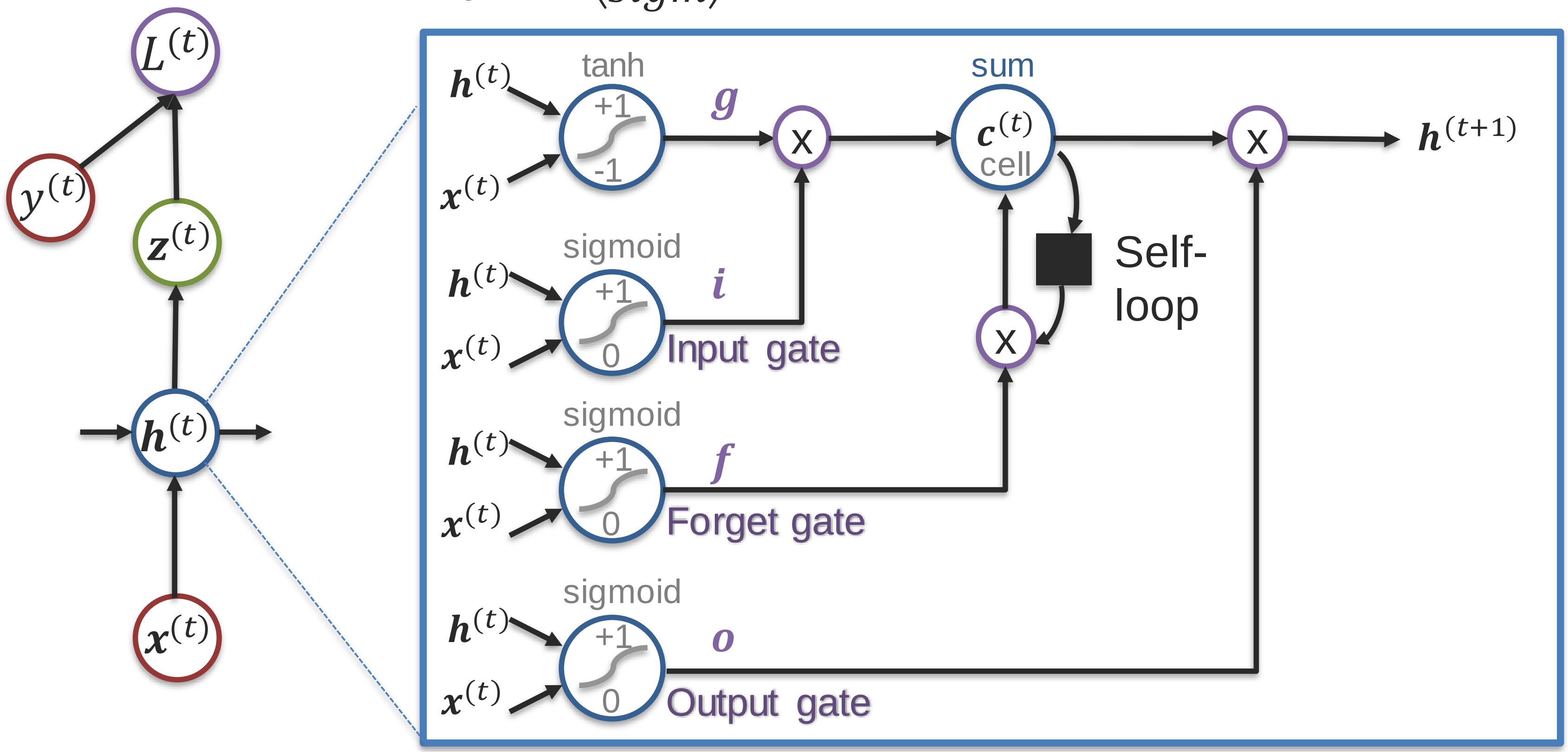


LSTM Ideas: (3) Forget Gate [Gers et al., 2000]

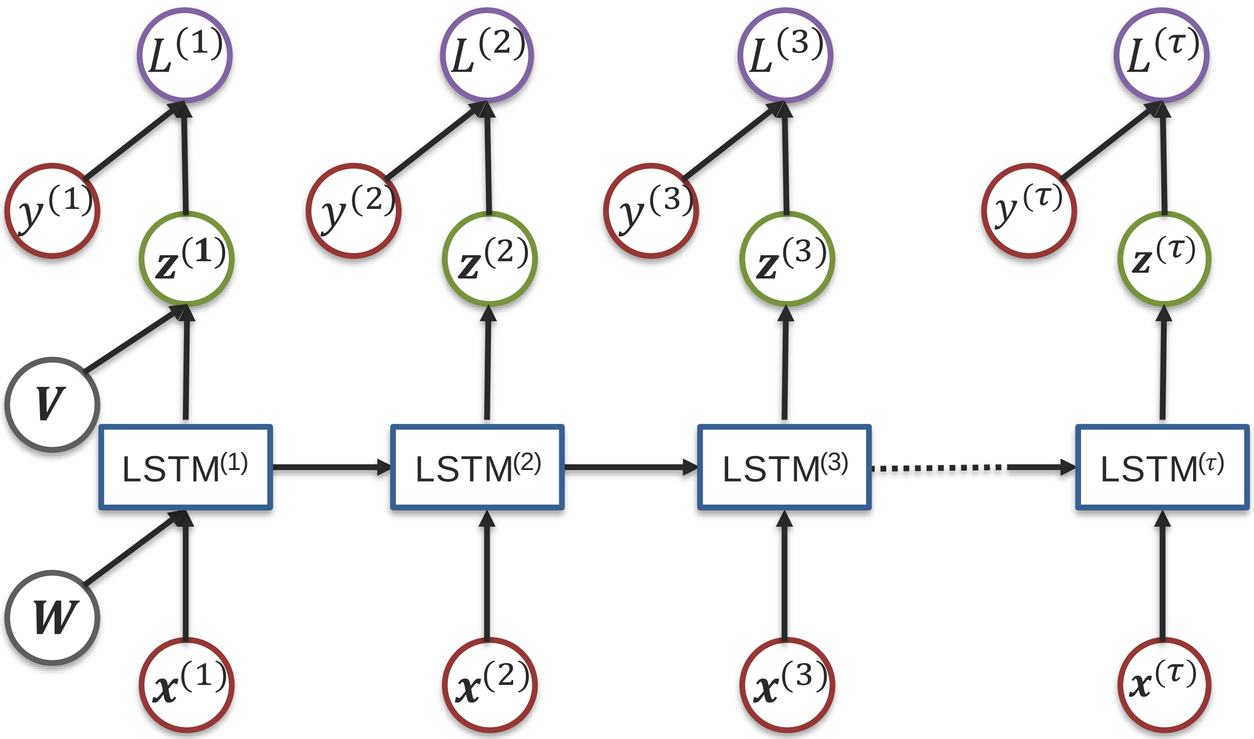
$$\begin{pmatrix} \boldsymbol{g} \\ \boldsymbol{i} \\ \boldsymbol{f} \\ \boldsymbol{o} \end{pmatrix} = \begin{pmatrix} tanh \\ sigm \\ sigm \\ sigm \end{pmatrix} W \begin{pmatrix} \boldsymbol{h}^{(t)} \\ \boldsymbol{x}^{(t)} \end{pmatrix}$$

$$\boldsymbol{c}^{(t)} = \boldsymbol{f} \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{i} \odot \boldsymbol{g}$$

$$\boldsymbol{h}^{(t)} = \boldsymbol{o} \odot \tanh(\boldsymbol{c}^{(t)})$$

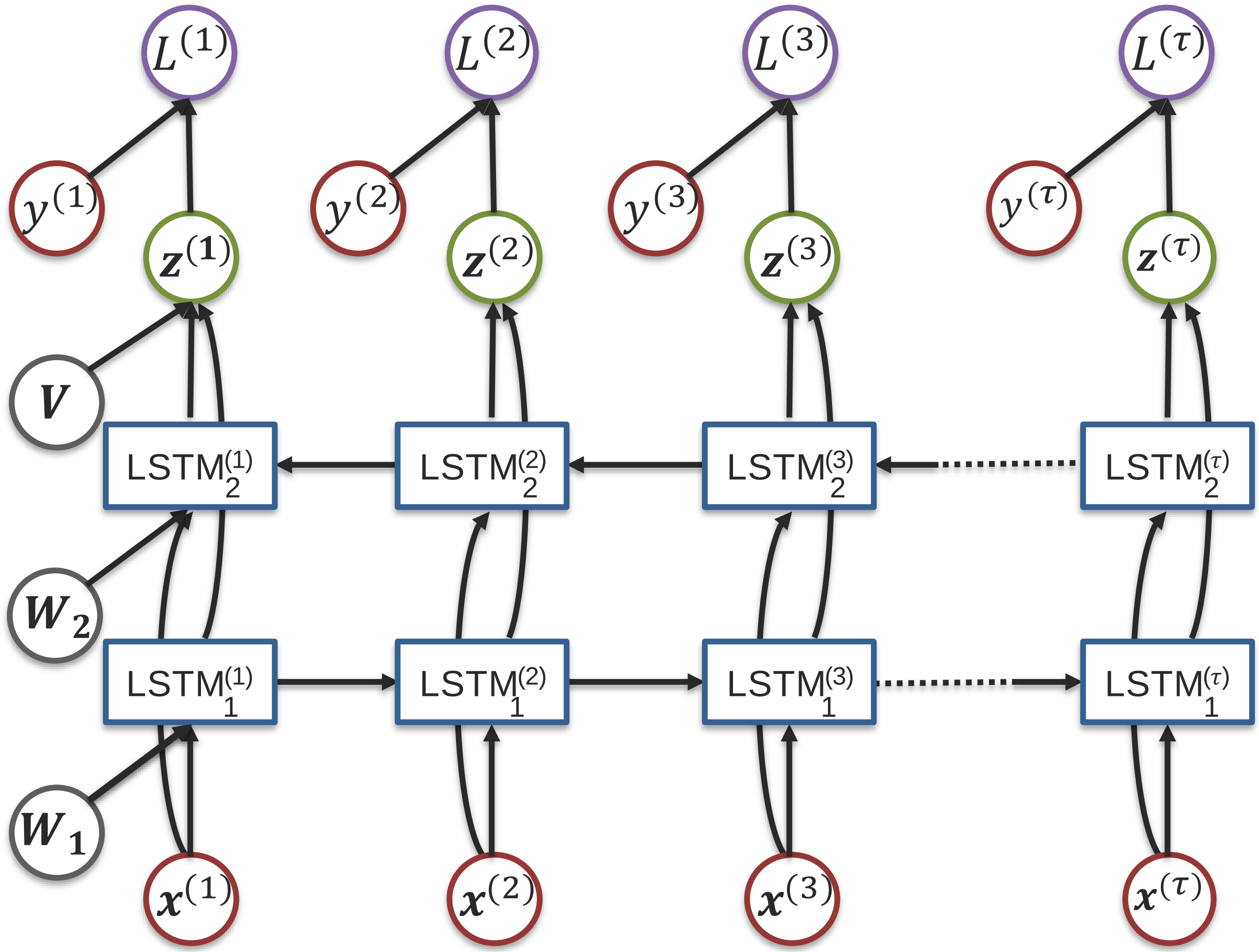


Recurrent Neural Network using LSTM Units



Gradient can still be computer using backpropagation!

Bi-directional LSTM Network



Deep LSTM Network