Advanced Multimodal Machine Learning

Lecture 4.1: Recurrent Networks

Louis-Philippe Morency

* Original version co-developed with Tadas Baltrusaitis

Administrative Stuff

Upcoming Schedule

- First project assignment:
  - Proposal presentation (10/2 and 10/4)
  - First project report (Sunday 10/7)
- Second project assignment
  - Midterm presentations (11/6 and 11/8)
  - Midterm report (Sunday 11/11)
- Final project assignment
  - Final presentation (12/4 & 12/6)
  - Final report (Sunday 12/9)

Proposal Presentation (10/2 and 10/4)

- 5 minutes (about 5-10 slides)
- All team members should be involved in the presentation
- Will receive feedback from instructors and other students
  - 1-2 minutes between presentations reserved for written feedback
- Main presentation points
  - General research problem and motivation
  - Dataset and input modalities
  - Multimodal challenges and prior work
- You need to submit a copy of your slides (PDF or PPT)
  - Deadline: Friday 10/5 (on Gradescope)

Project Proposal Report

- Part 1 (updated version of your pre-proposal)
  - **Research problem:**
    - Describe and motivate the research problem
    - Define in generic terms the main computational challenges
  - **Dataset and Input Modalities:**
    - Describe the dataset(s) you are planning to use for this project.
    - Describe the input modalities and annotations available in this dataset.

Project Proposal Report

- Part 2
  - **Related Work:**
    - Include 12-15 paper citations which give an overview of the prior work
    - Present in more details the 3-4 research papers most related to your work
  - **Research Challenges and Hypotheses:**
    - Describe your specific challenges and/or research hypotheses
    - Highlight the novel aspect of your proposed research

Project Proposal Report

- Part 3
  - **Language Modality Exploration:**
    - Explore neural language models on your dataset (e.g., using Keras)
    - Train at least two different language models (e.g., using SimpleRNN, GRU or LSTM) on your dataset and compare their perplexity.
    - Include qualitative examples of successes and failure cases.
  - **Visual Modality Exploration:**
    - Explore pre-trained Convolutional Neural Networks (CNNs) on your dataset
    - Load a pre-existing CNN model trained for object recognition (e.g., VGG-Net) and process your test images.
    - Extract features at different network layers in the network and visualize them (using t-sne visualization) with overlaid class labels with different colors.

Lecture Objectives

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

Representing Words: 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.

11

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

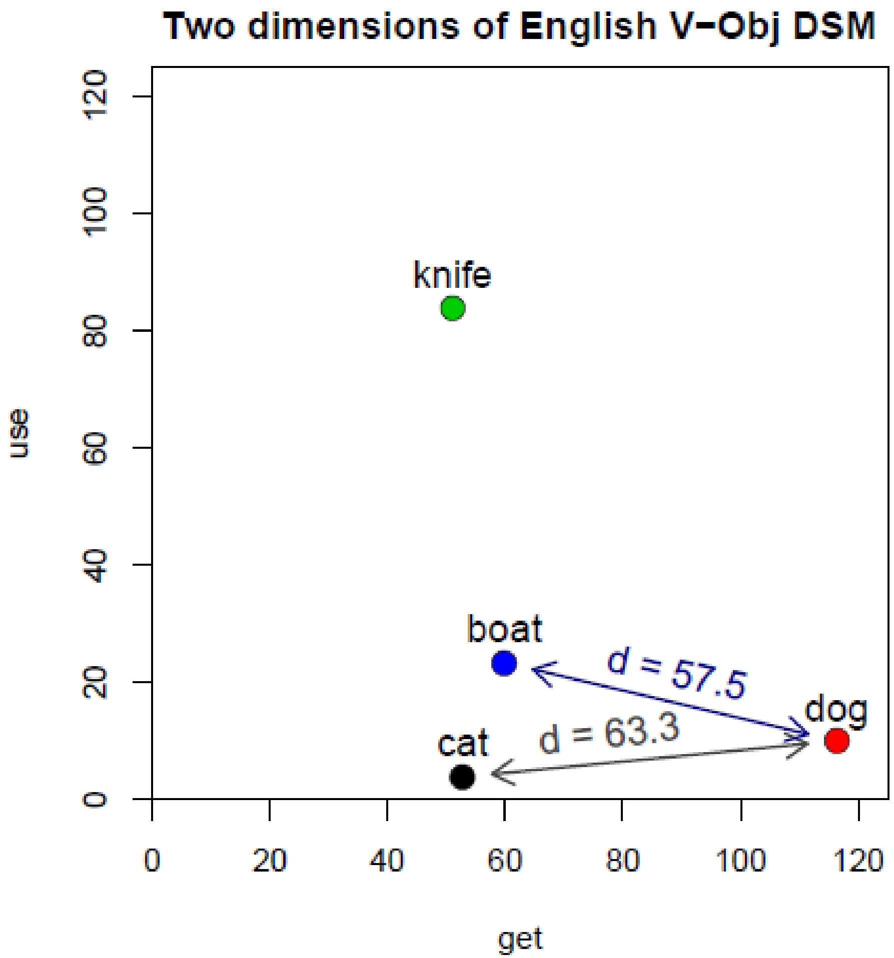
- can be seen as coordinates of point in *n*-dimensional Euclidean space R^n

	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**

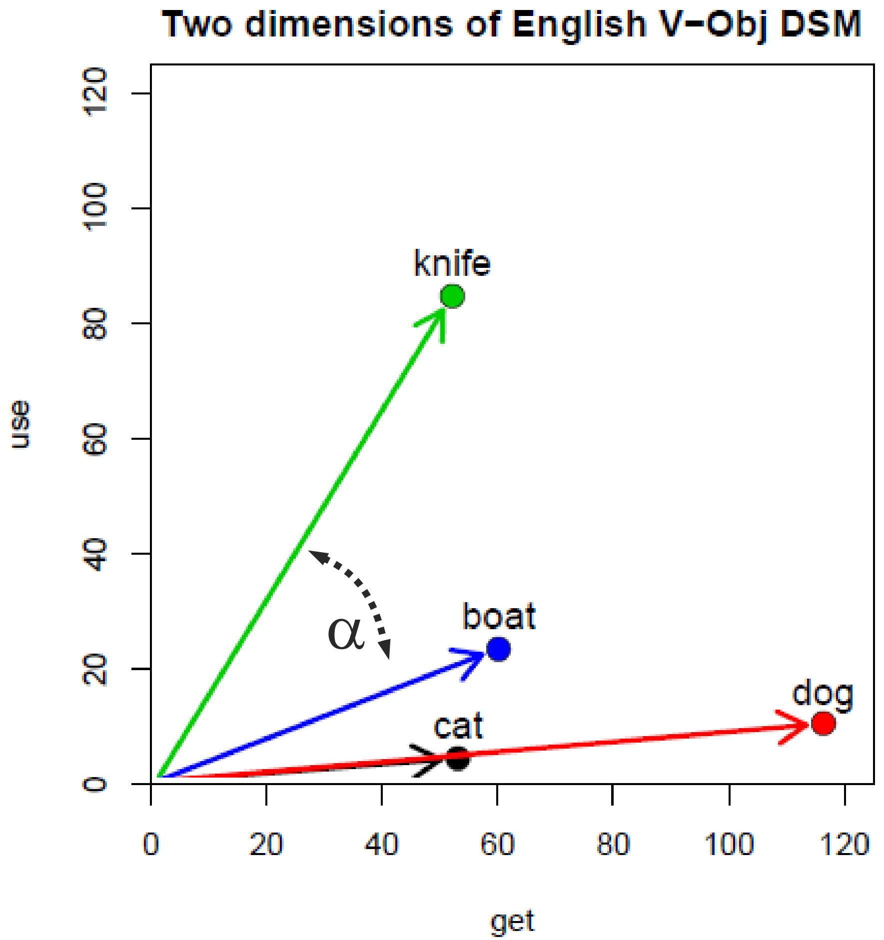
Distance and similarity

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

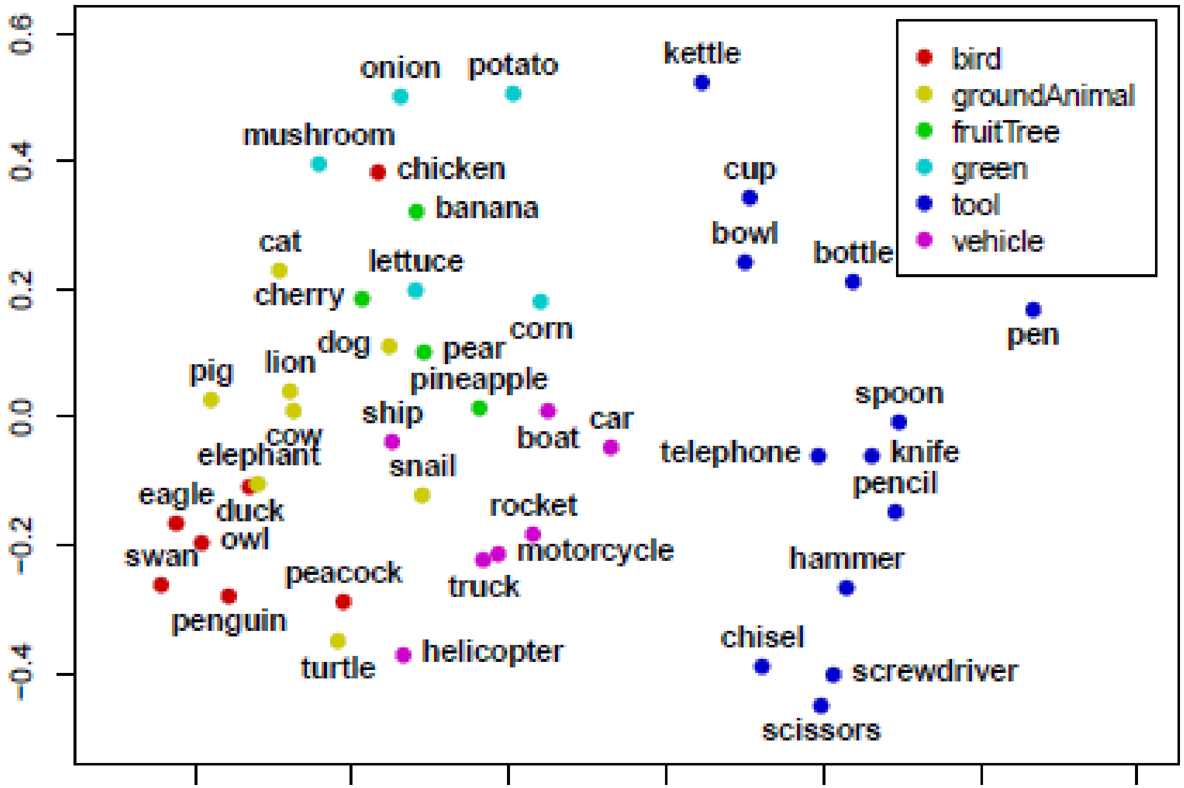


Angle and similarity

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



Semantic maps



Learning Neural Word Representations

How to learn neural word representations?

Distribution hypothesis: Approximate the word meaning by its surrounding words

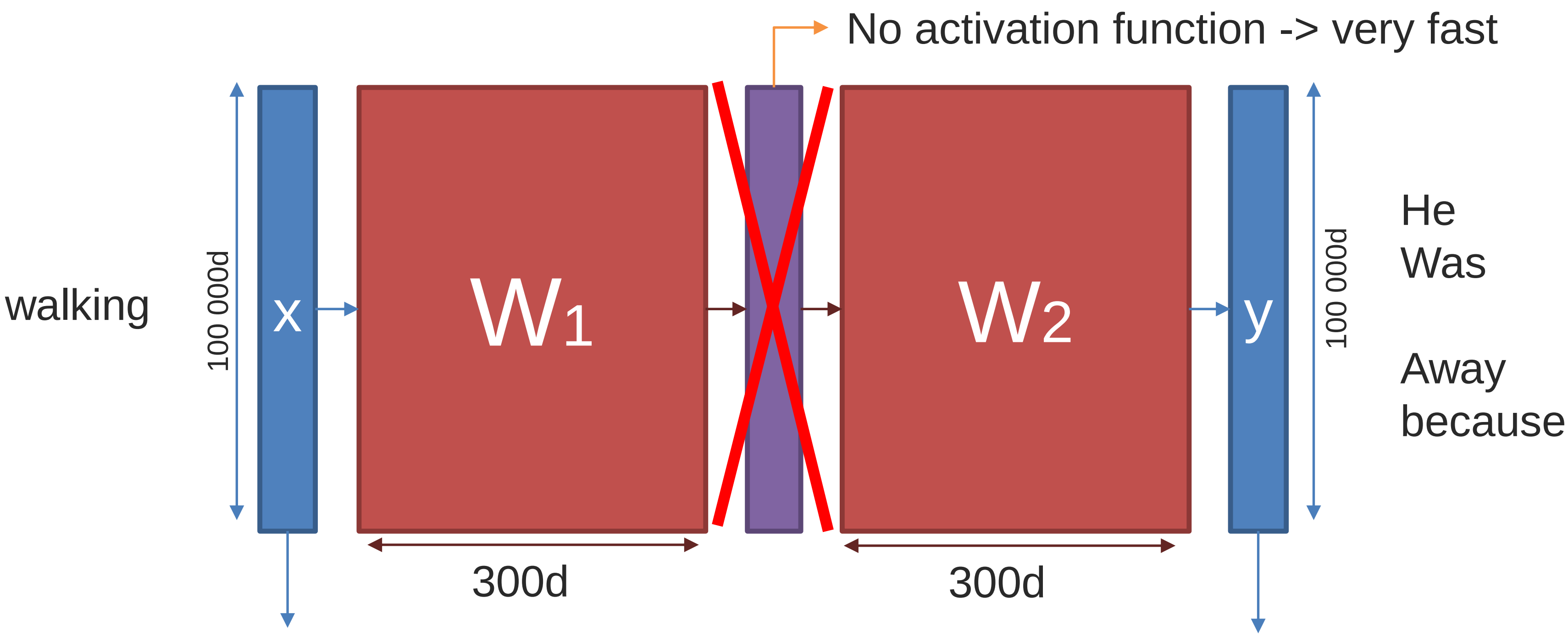
Words used in a similar context will lie close together

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 neural word representations?



[0; 0; 0; 0;….; 0; 0; 1; 0;…; 0; 0]

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

[0; 1; 0; 0;….; 0; 0; 0; 0;…; 0; 0]
[0; 0; 0; 1;….; 0; 0; 0; 0;…; 0; 0]
[0; 0; 0; 0;….; 1; 0; 0; 0;…; 0; 0]
[0; 0; 0; 0;….; 0; 0; 0; 0;…; 0; 1]

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

How to use these word representations

If we would have a vocabulary of 100 000 words:

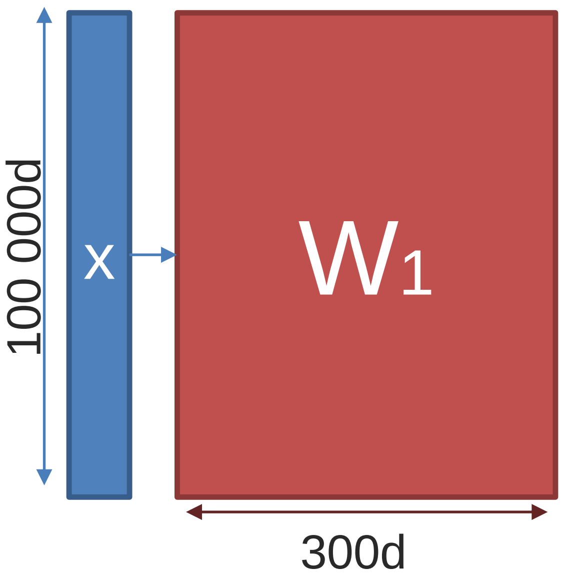
Classic NLP: 100 000 dimensional vector

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

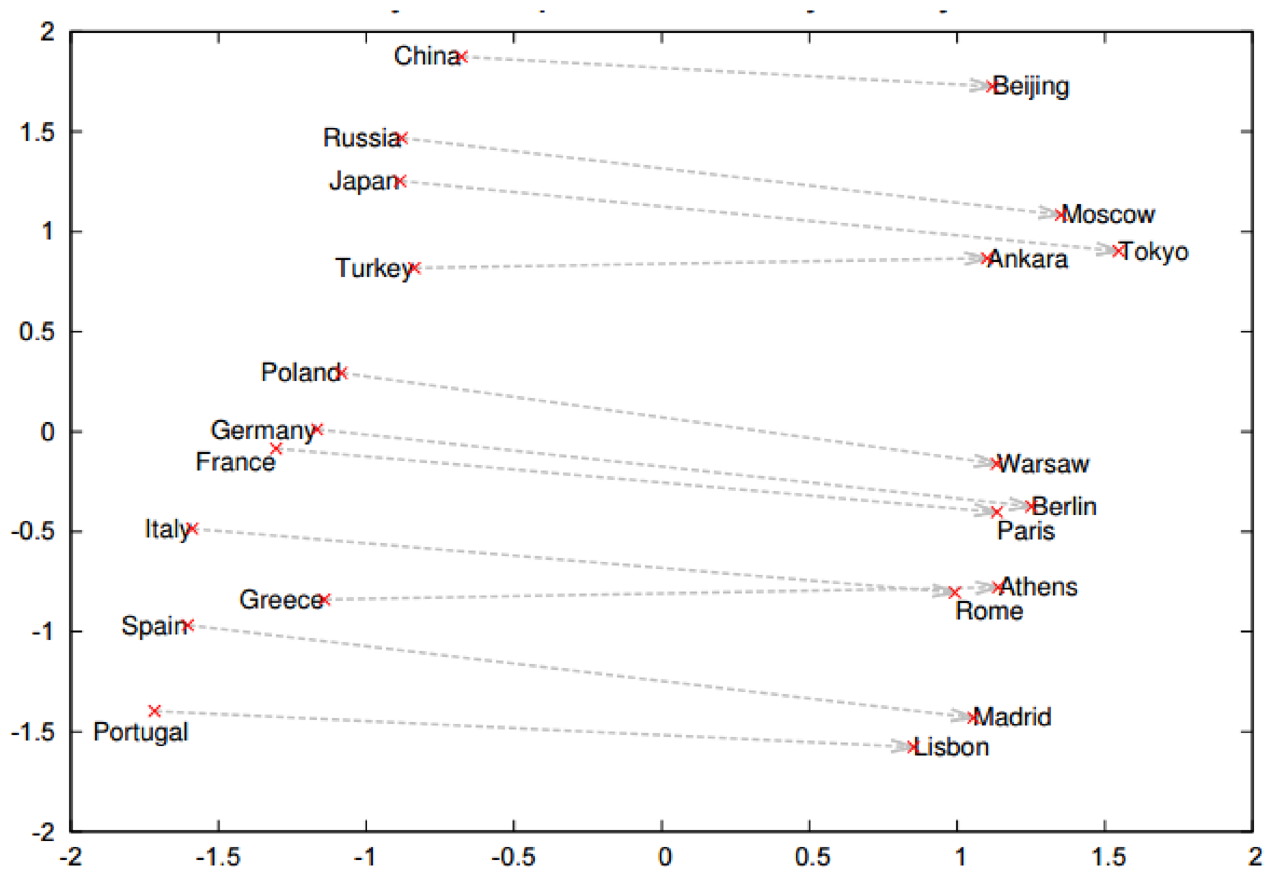
Vector space models of words

- 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:

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

Why linear algebra is working?

Vector space models of words: semantic relationships



Trained on the Google news corpus with over 300 billion words

Language Sequence Modeling Tasks

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 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 Representation

Learning

Sequence representation

[0,1; 0,0004; 0;….; 0,01; 0.09; 0,05]

Ideal for anyone with an interest in disguises

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

Application: Speech Recognition

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

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

$$\arg\max_{wordsequence} P(acoustics \mid wordsequence) \times P(wordsequence)$$

Language model

Application: Language Generation

Embedding

[0,1;
0,0004;
….;
0.09;
0,05]

Generation

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

Example: Image captioning

[0,1;
0,0004;
….;
0.09;
0,05]

The man at bat readies to swing at the pitch while the umpire looks on.

N-Gram Language Model Formulations

- Word sequences

$$w_1^n = w_1 \ldots w_n$$

- Chain rule of probability

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

- Bigram approximation

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

- N-gram approximation

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

Evaluating Language Model: Perplexity

The best language model is one that best predicts an unseen test set

- Gives the highest P(sentence)

Perplexity is the inverse probability of the test set, normalized by the number of words:

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Chain rule:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$$

For bigrams:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$$

Challenges in 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

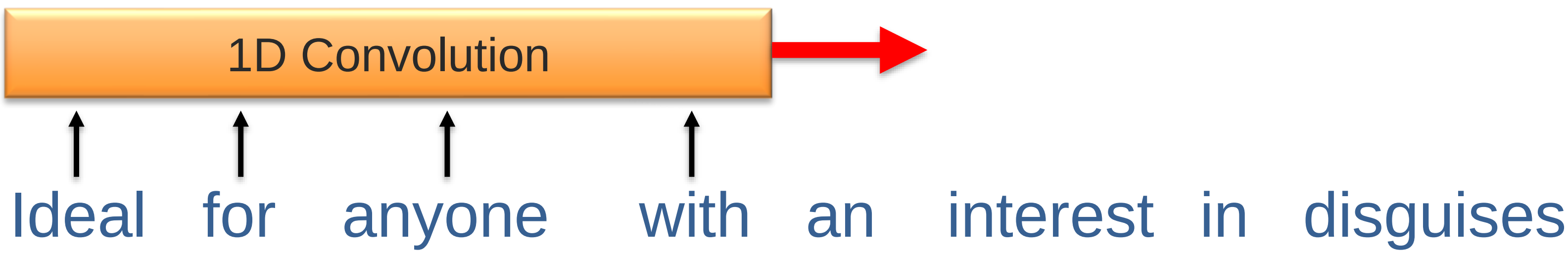
Model

- 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



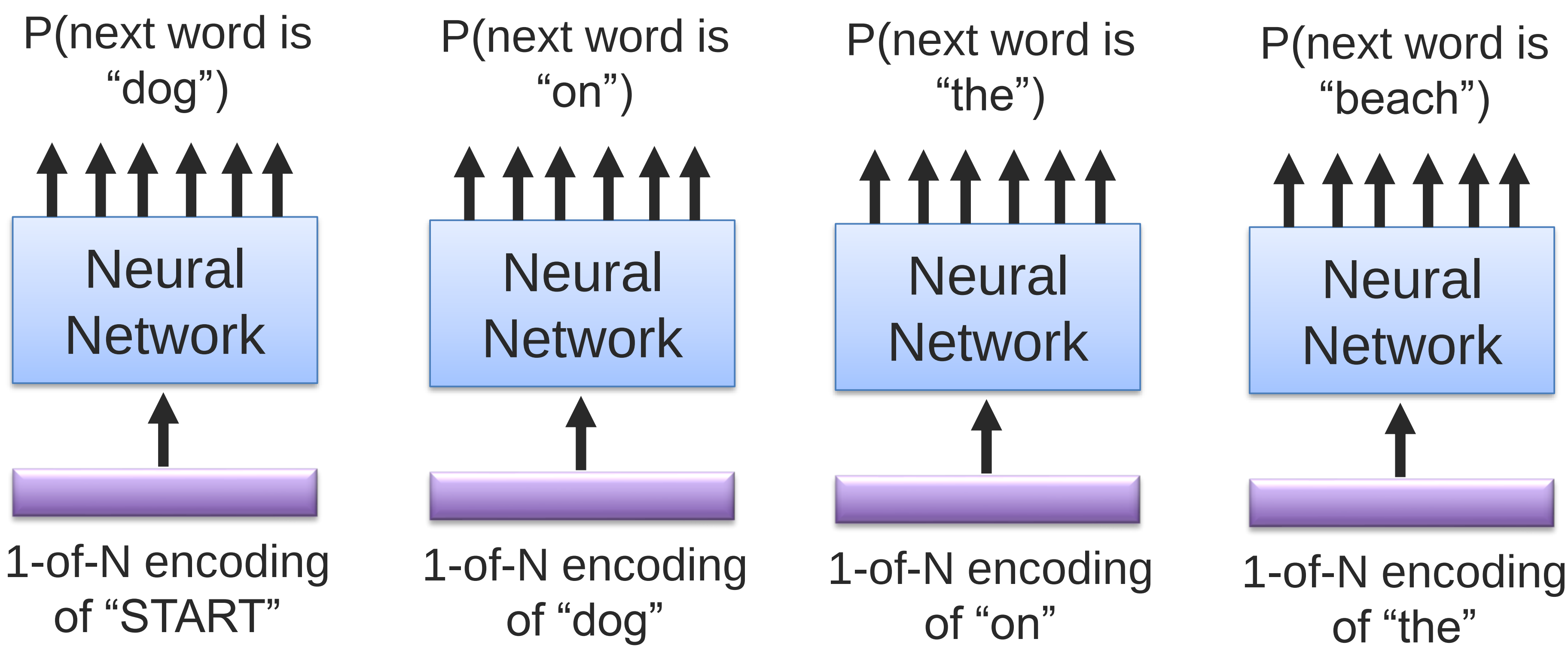
Main Challenges:

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

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.

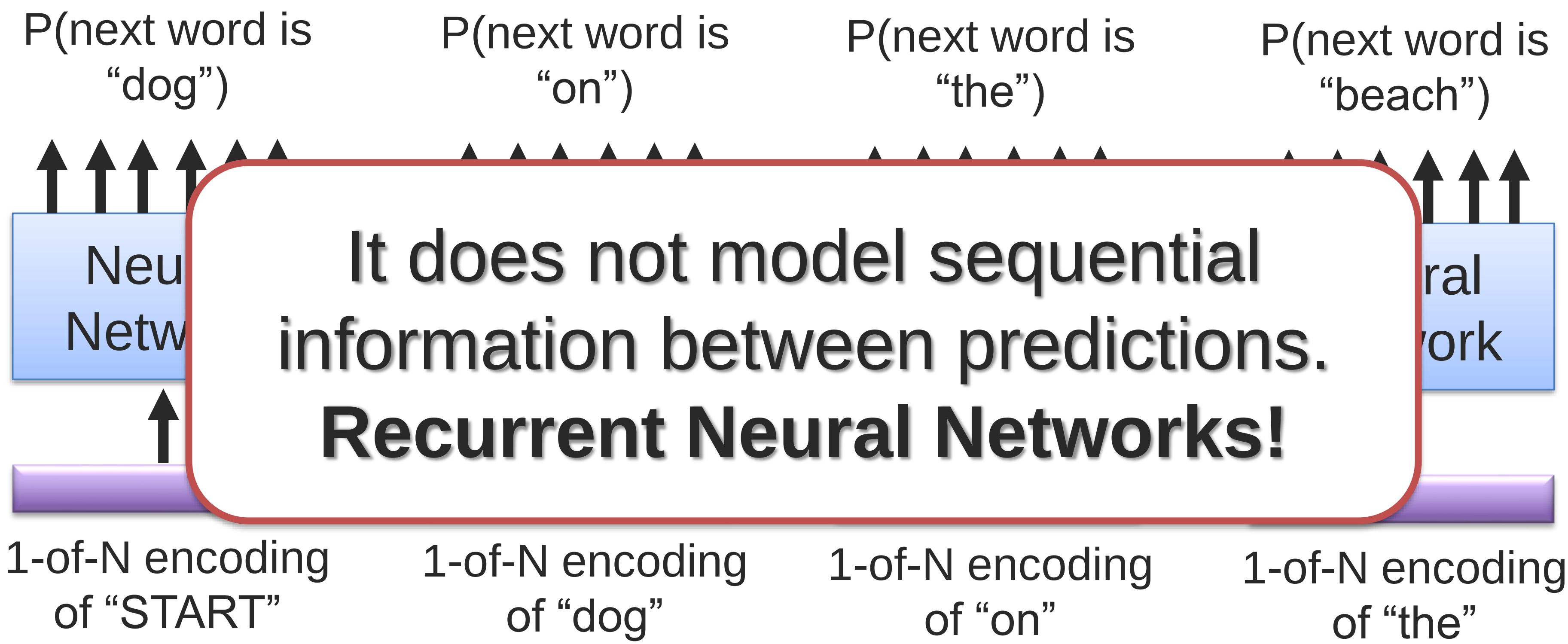


Neural-based Unigram Language Model (LM)

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.

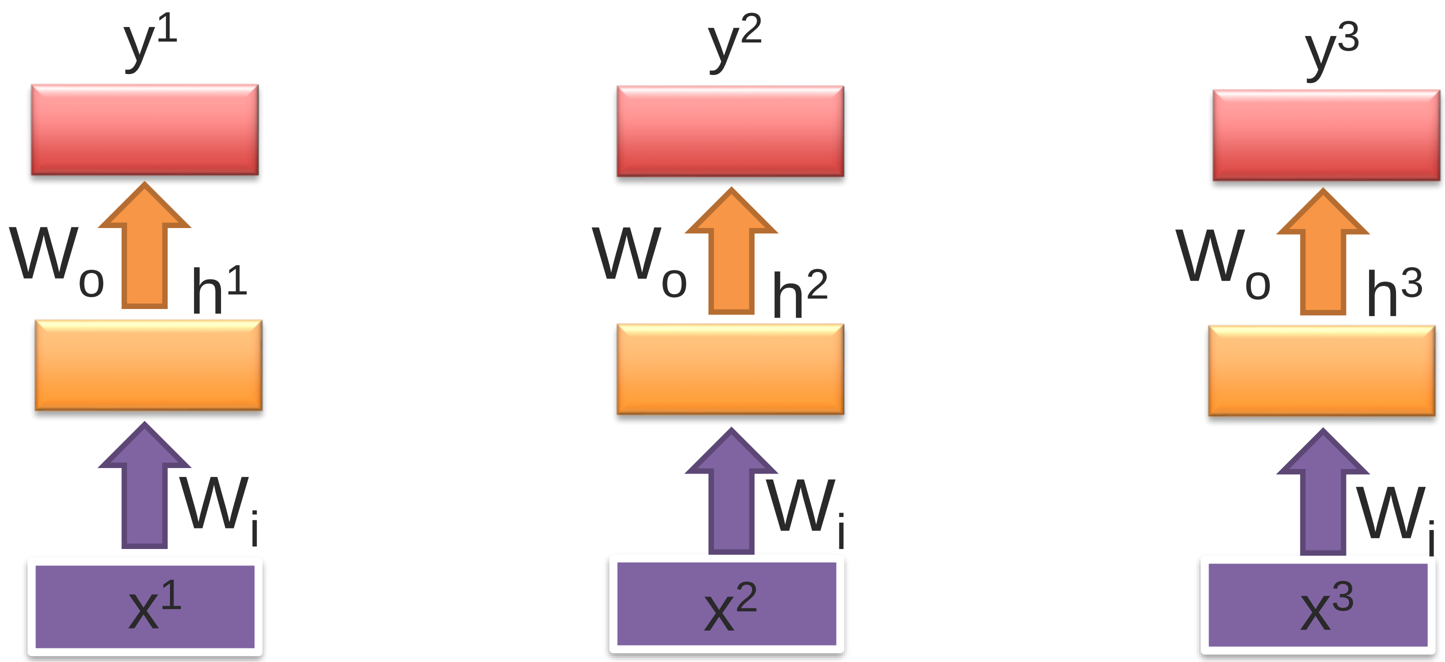
Recurrent Neural Networks!

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)



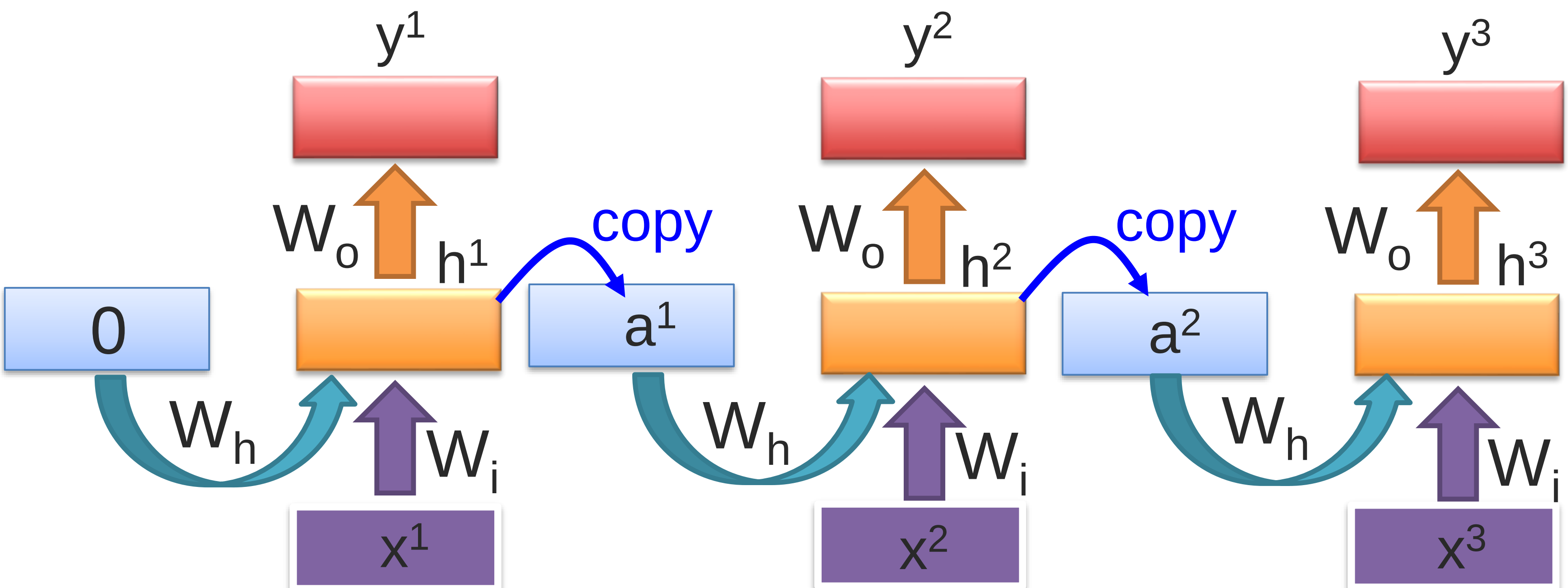
How can we include temporal dynamics?

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)

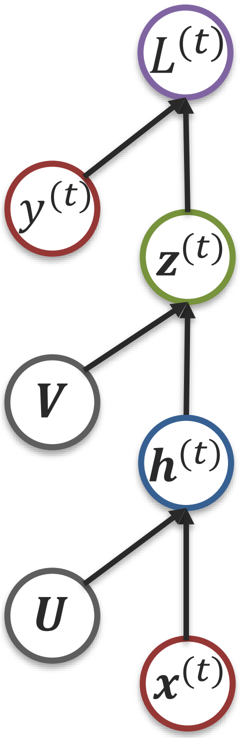


The same model parameters are used again and again.

Can be trained using backpropagation

Recurrent Neural Network

Feedforward Neural Network



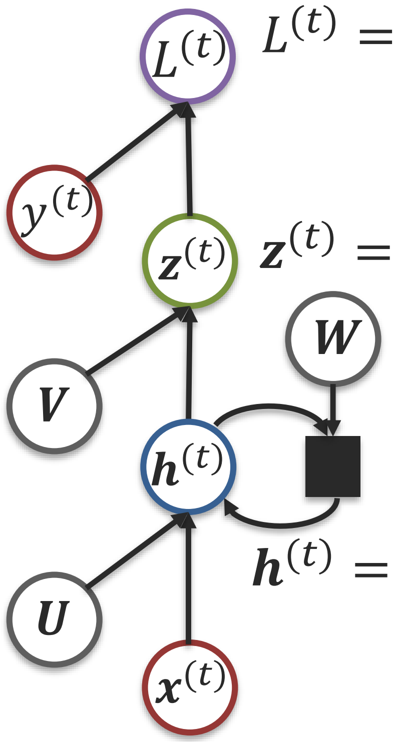
$$L^{(t)} = -logP(Y = y^{(t)} | \boldsymbol{z}^{(t)})$$

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

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

Recurrent Neural Networks

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



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

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

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

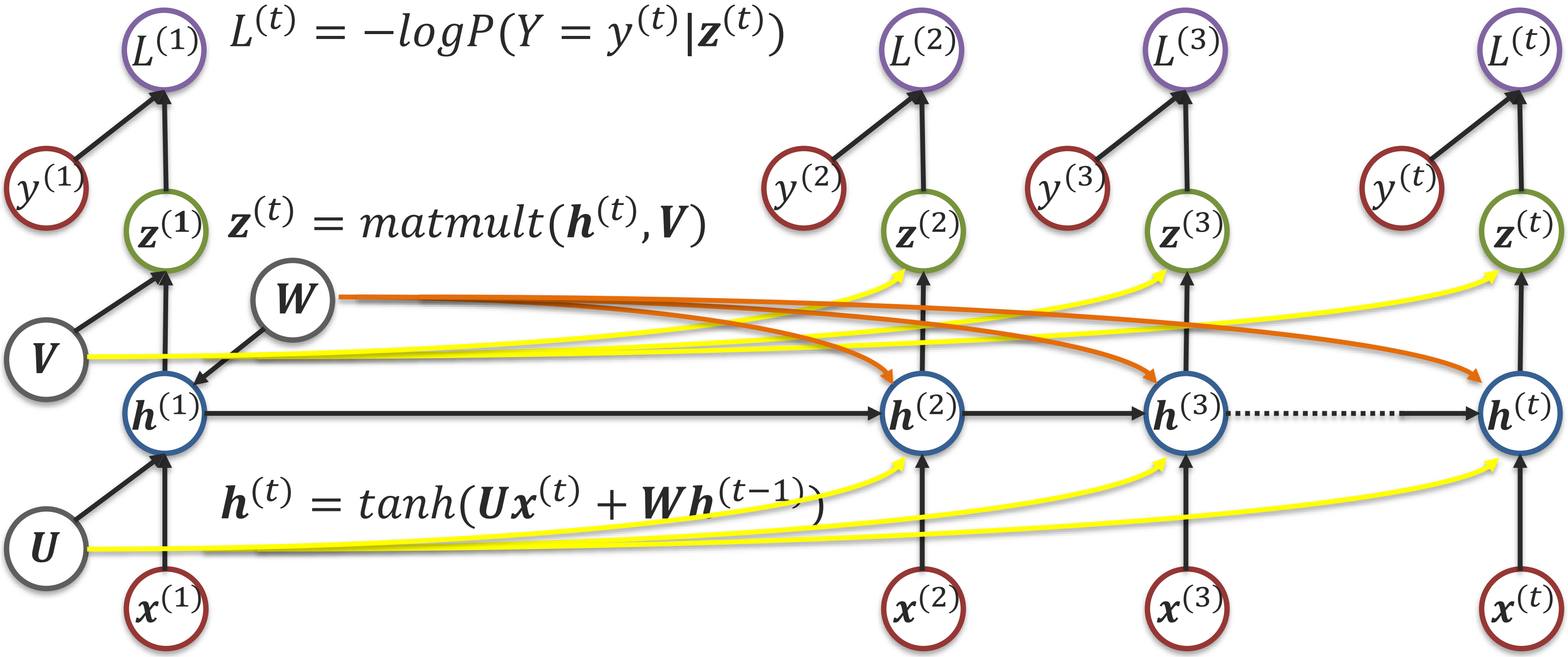
Recurrent Neural Networks - Unrolling

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

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

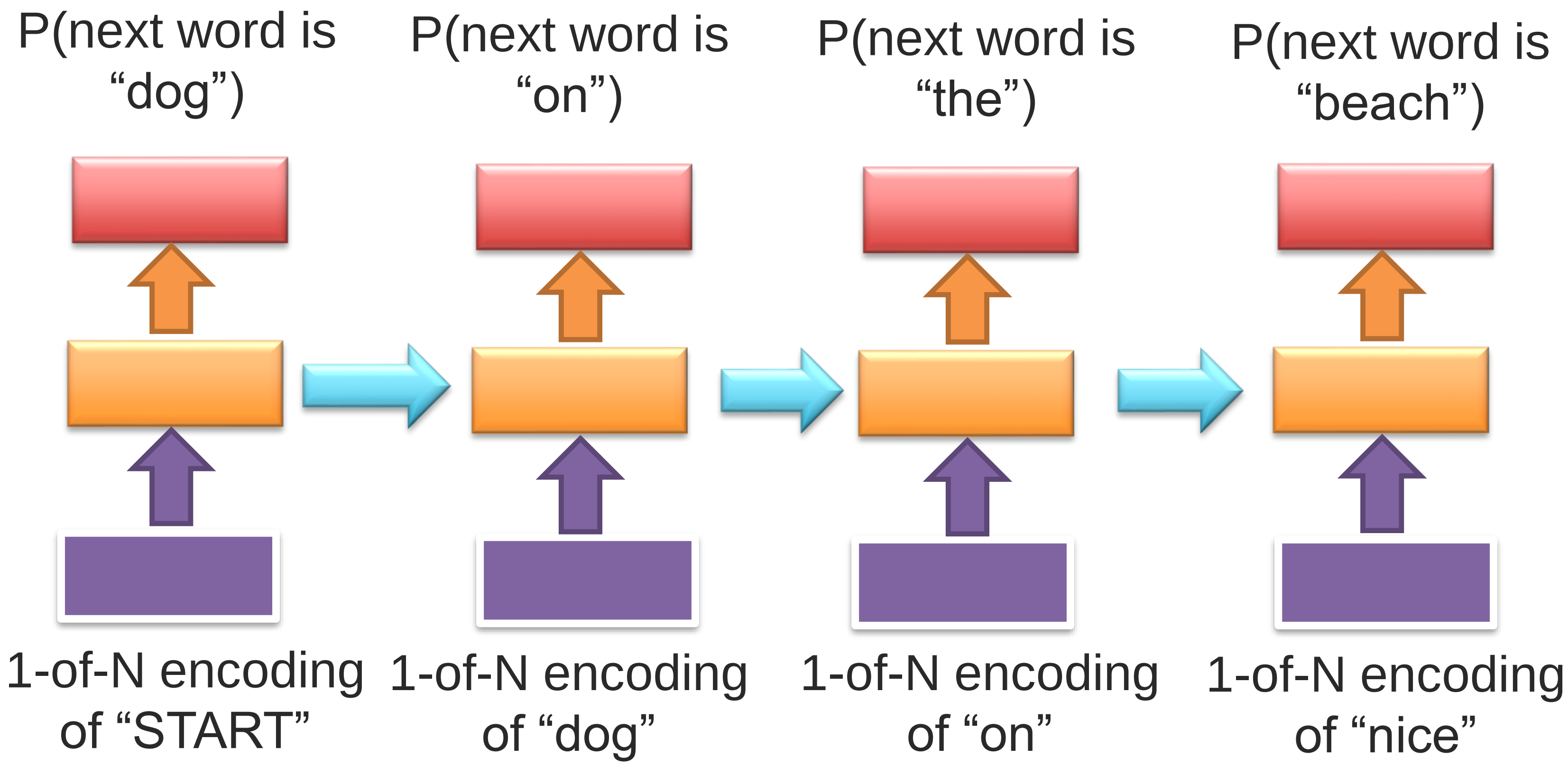
$$\boldsymbol{z}^{(t)} = matmult(\boldsymbol{h}^{(t)}, \boldsymbol{V})$$

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



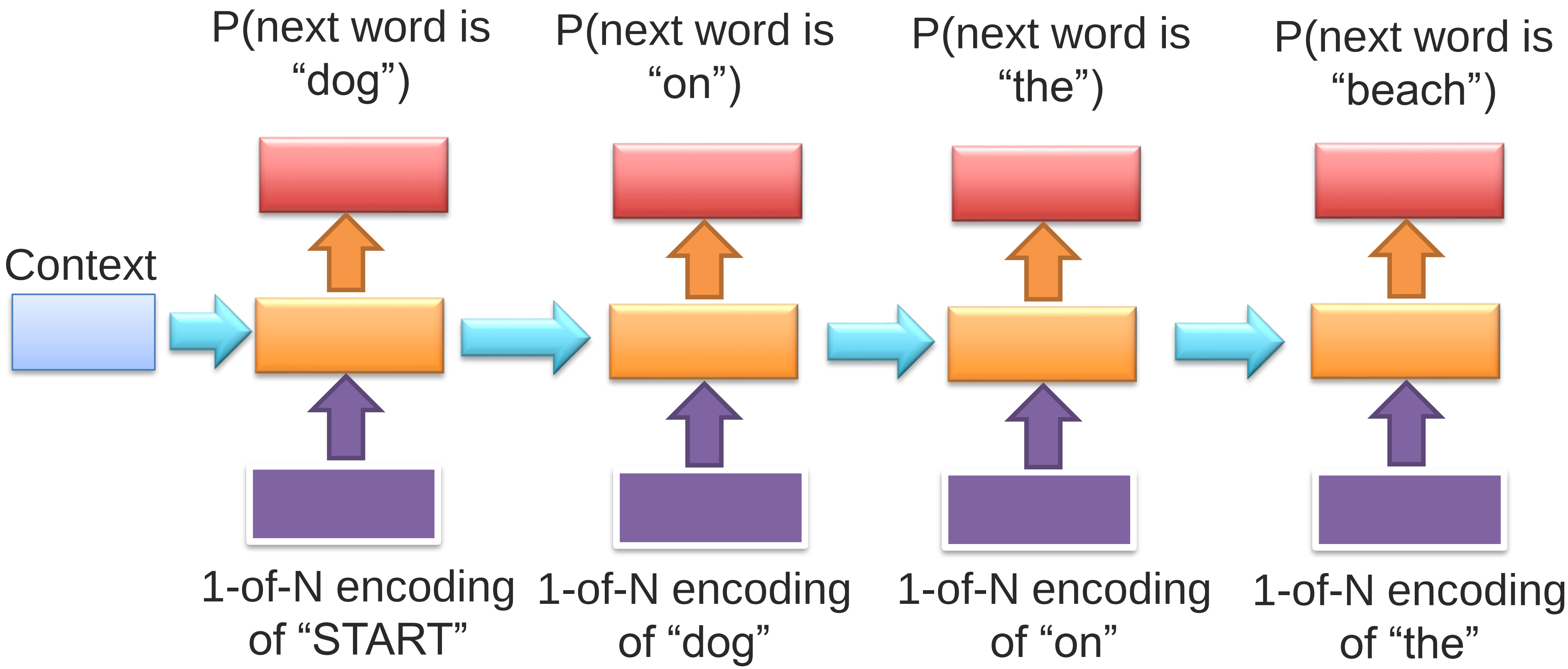
Same model parameters are used for all time parts.

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

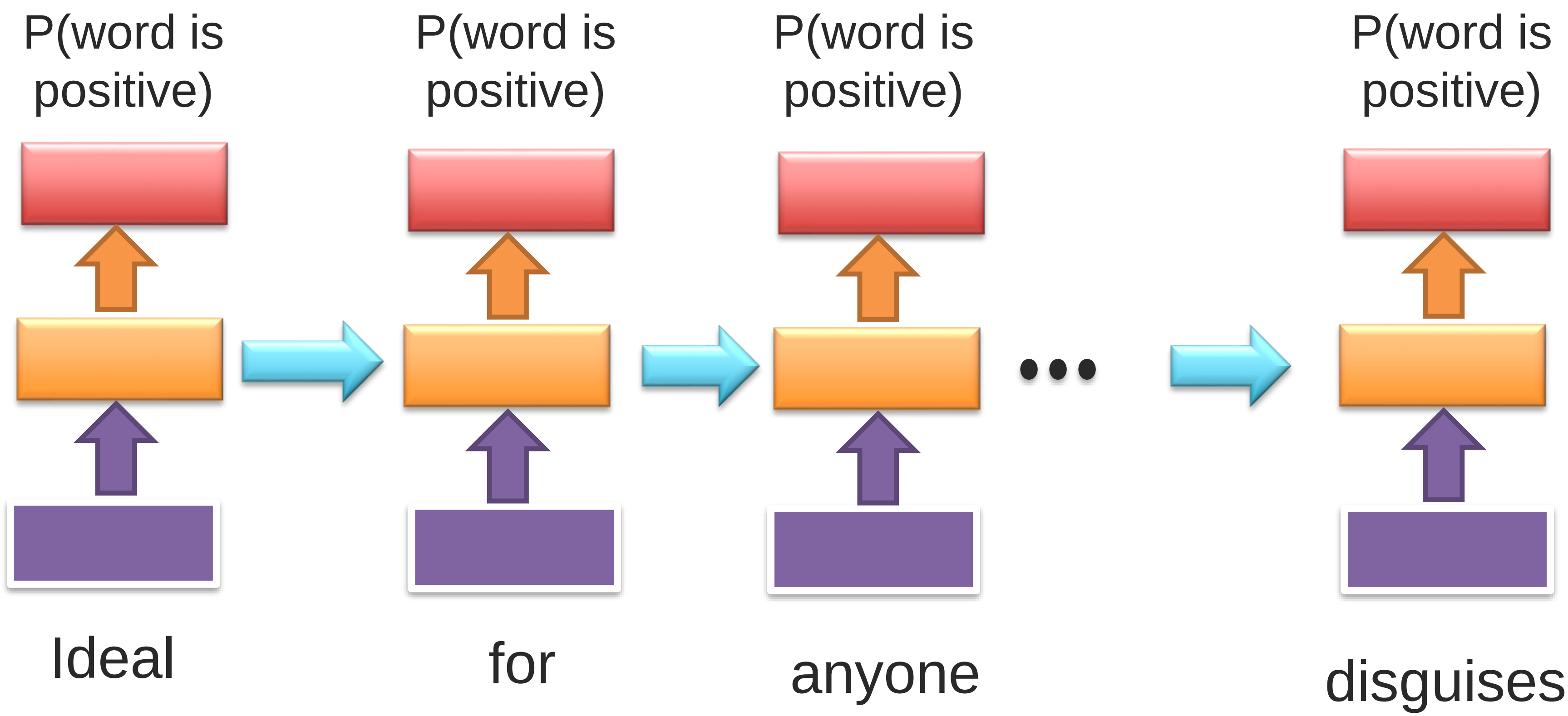
Prediction

Sentiment ?
(positive or negative)

Sentiment label?

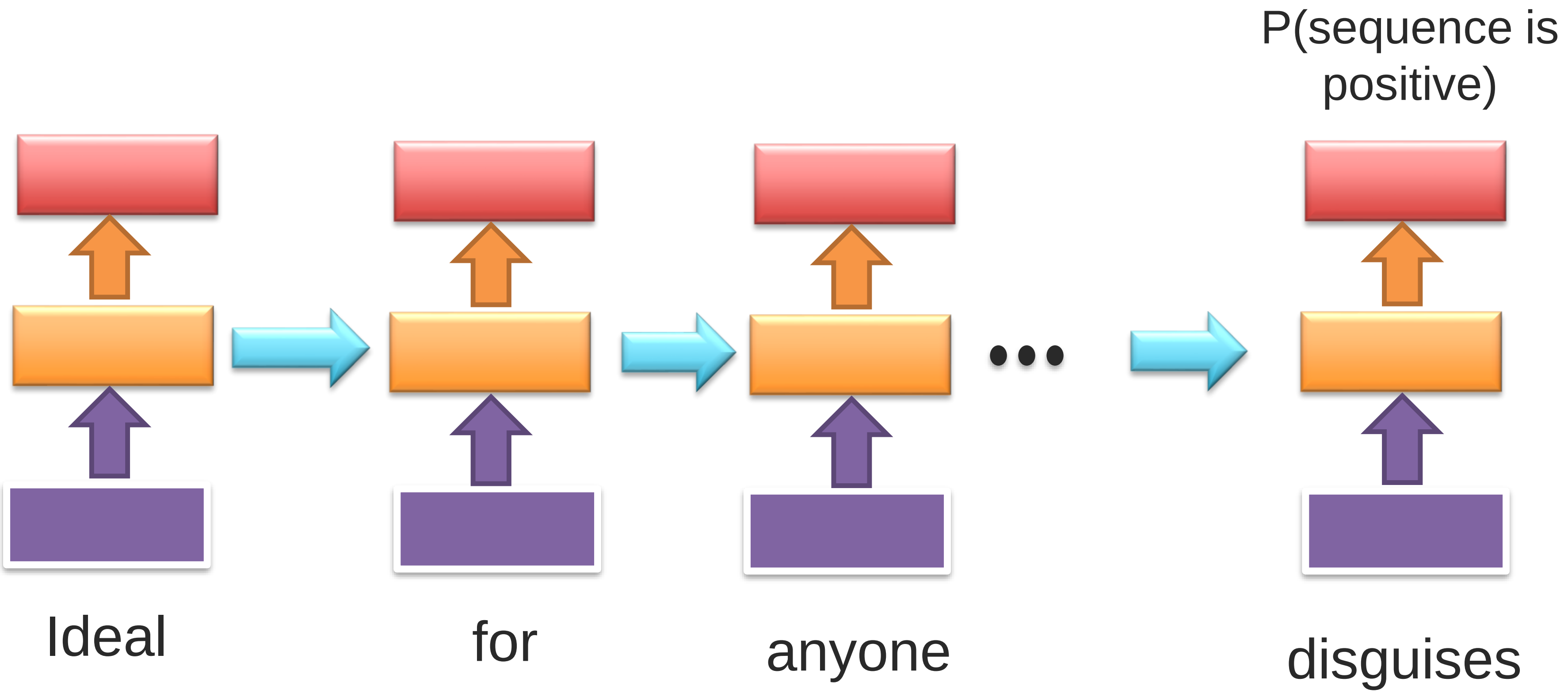
Ideal for anyone with an interest in disguises

RNN for Sequence Prediction



$$L = \frac{1}{N}\sum_t L^{(t)} = \frac{1}{N}\sum_t -logP(Y = y^{(t)}|\mathbf{z}^{(t)})$$

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

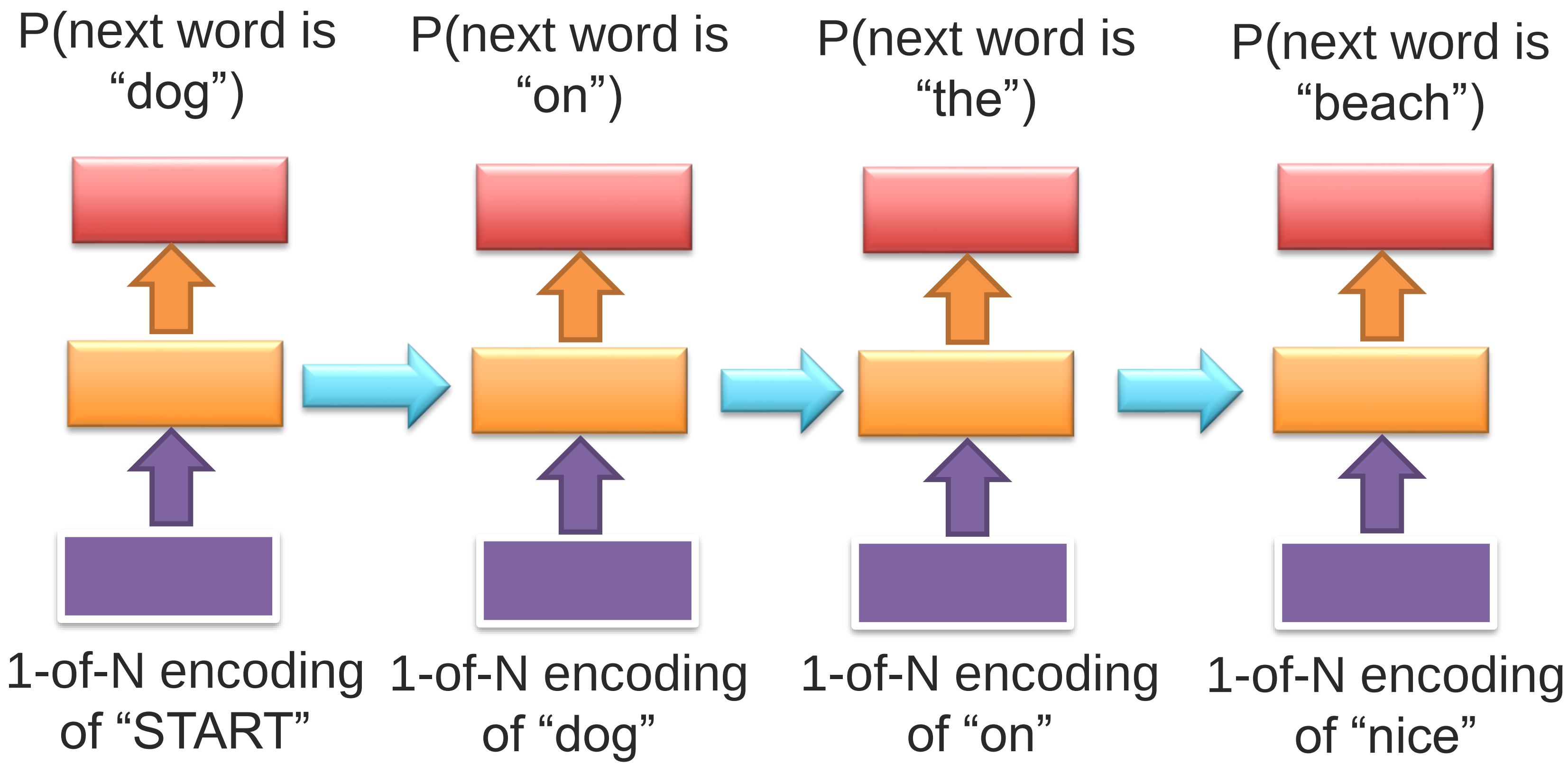
Learning

Sequence representation

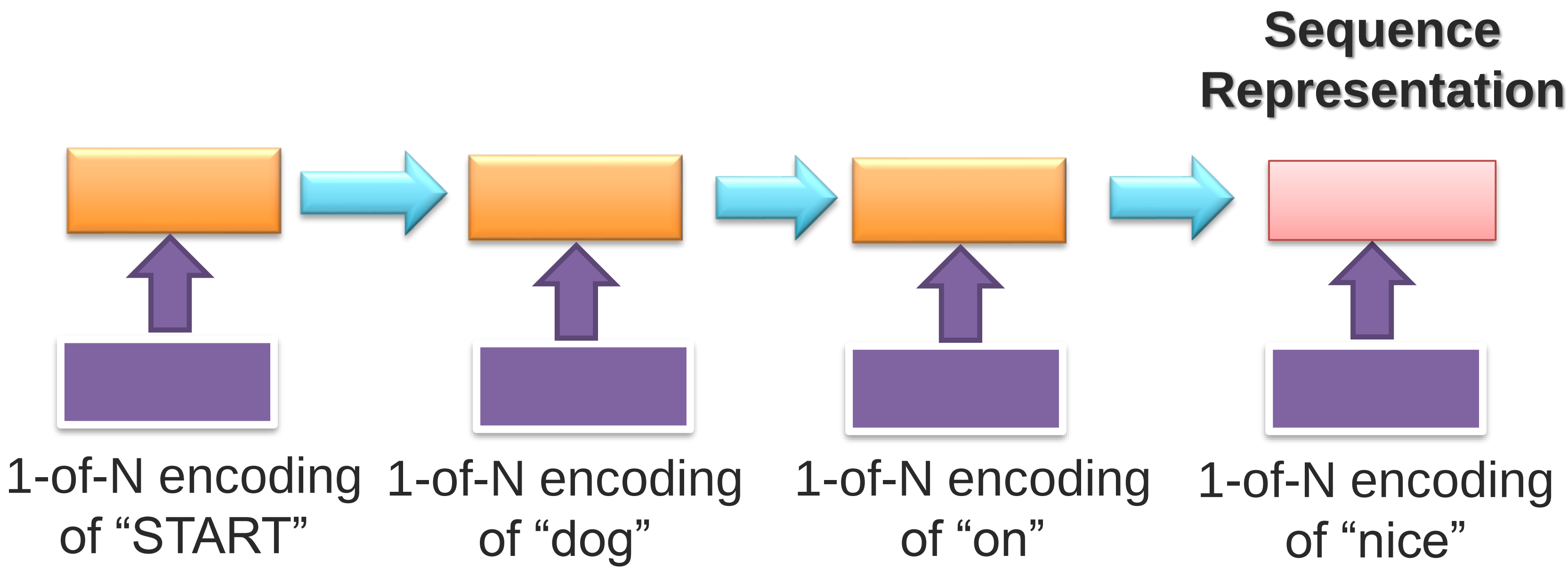
[0,1; 0,0004; 0;….; 0,01; 0.09; 0,05]

Ideal for anyone with an interest in disguises

RNN for Sequence Representation

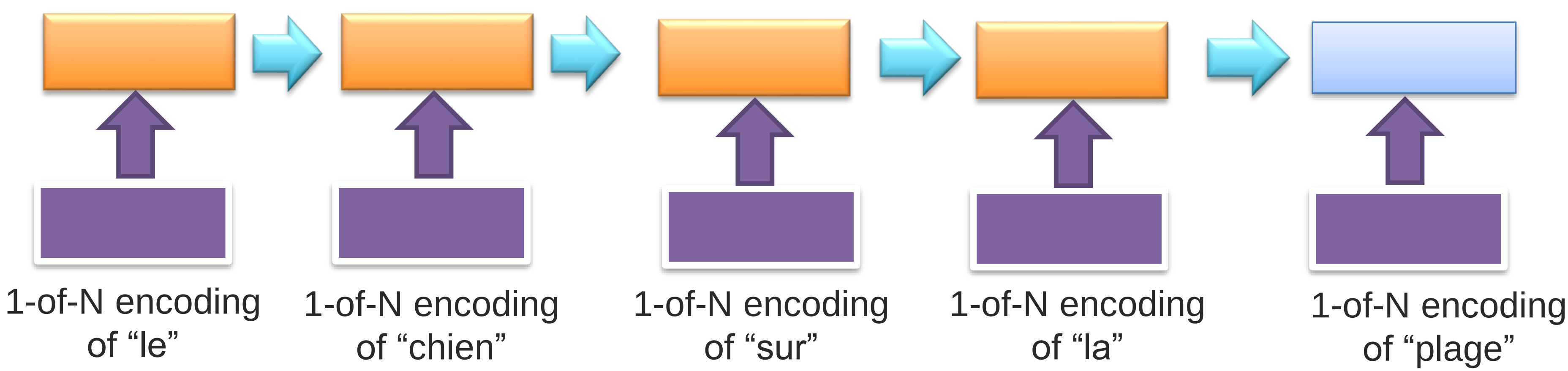


RNN for Sequence Representation (Encoder)

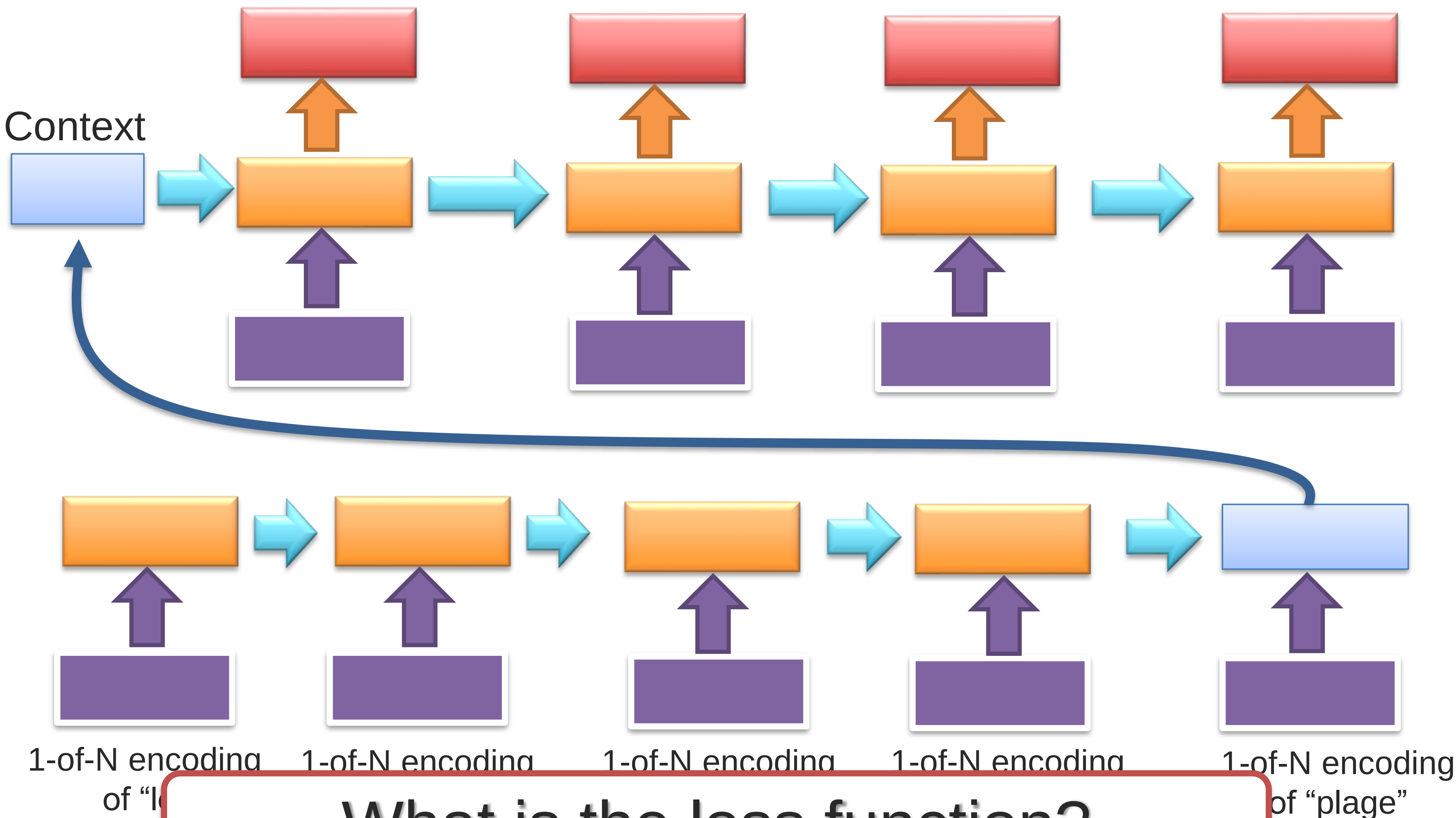


RNN-based for Machine Translation

Le chien sur la plage → The dog on the beach



Encoder-Decoder Architecture



What is the loss function?

Related 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:

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

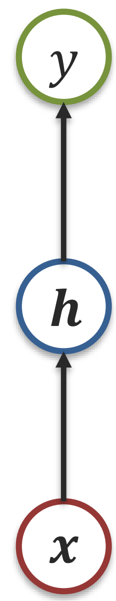
Gradient

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

"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

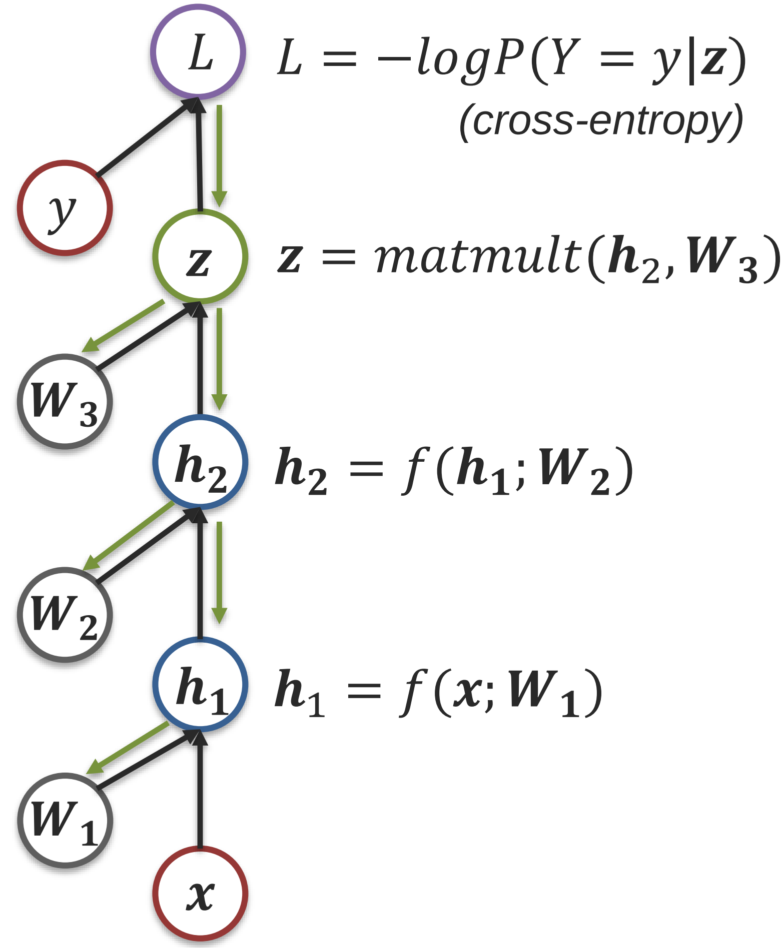
Forward pass

- 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



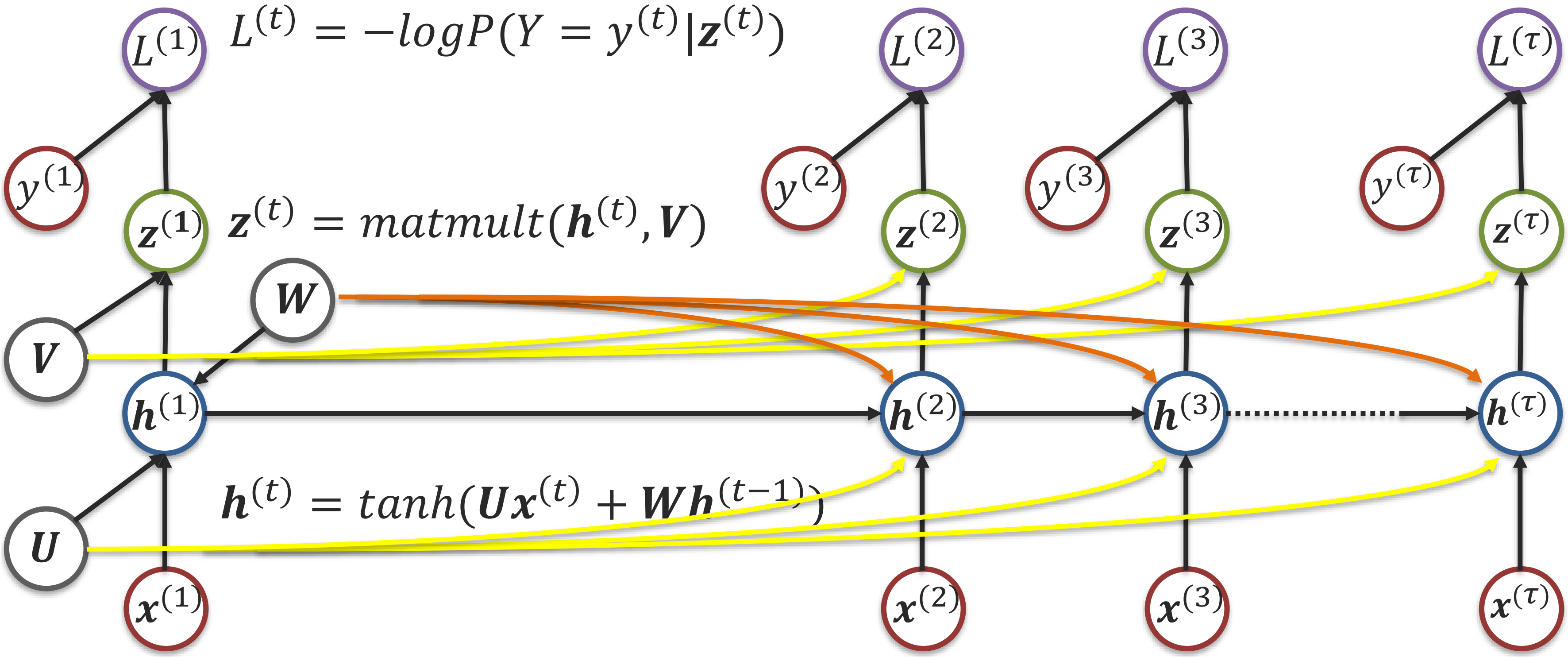
Recurrent Neural Networks

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

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

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

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



Backpropagation Through Time

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

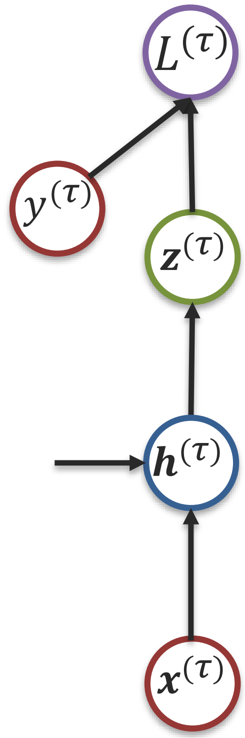
Gradient = "backprop" gradient
x "local" Jacobian

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

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

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

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



Backpropagation Through Time

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

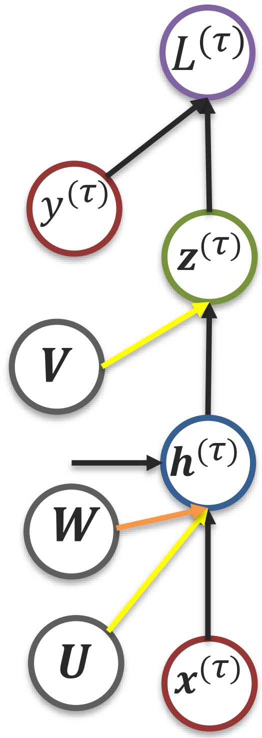
Gradient = "backprop" gradient

x "local" Jacobian

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

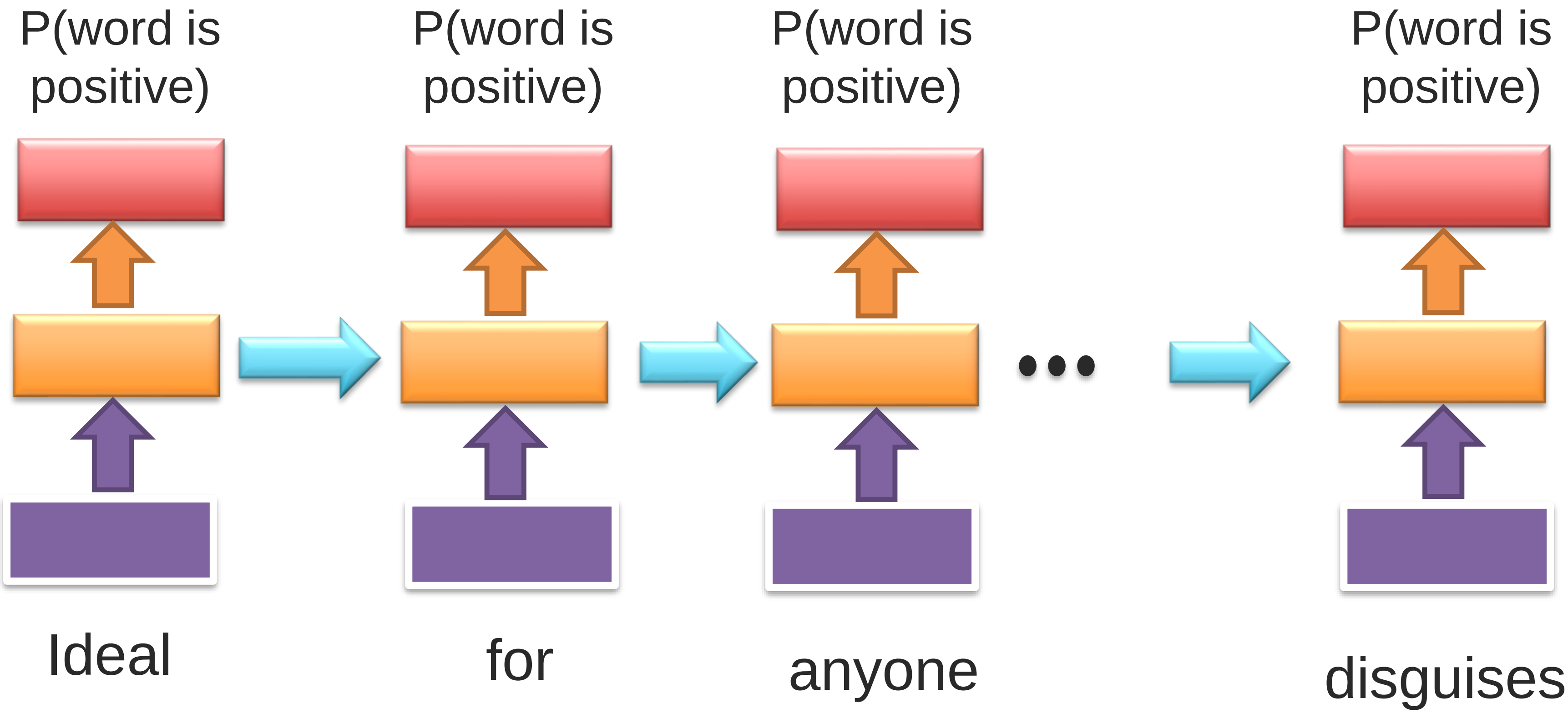
$\boldsymbol{W}$ $$\nabla_{\boldsymbol{W}} L = \sum_t (\nabla_{\boldsymbol{h}^{(t)}} L) \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{W}}$$

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



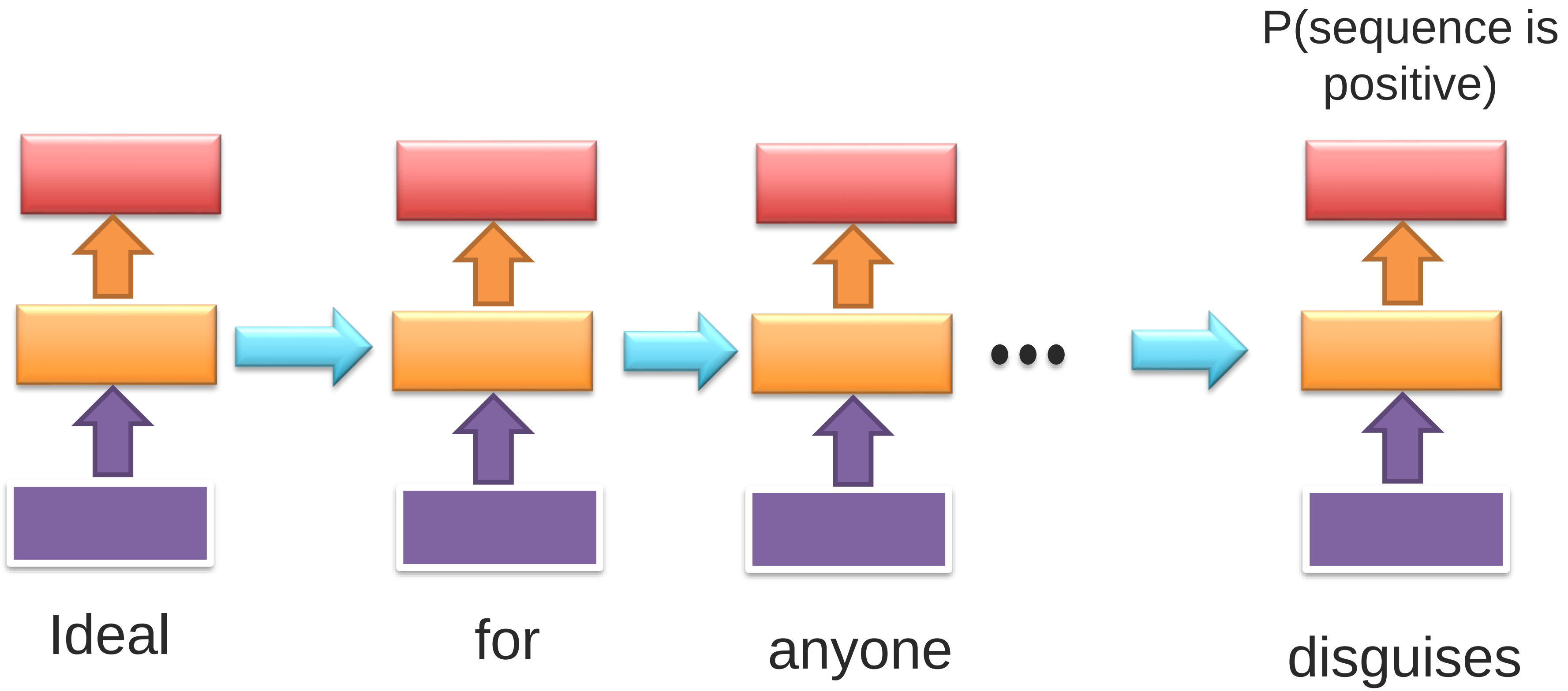
Gated Recurrent Neural Networks

RNN for Sequence Prediction



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

RNN for Sequence Prediction



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

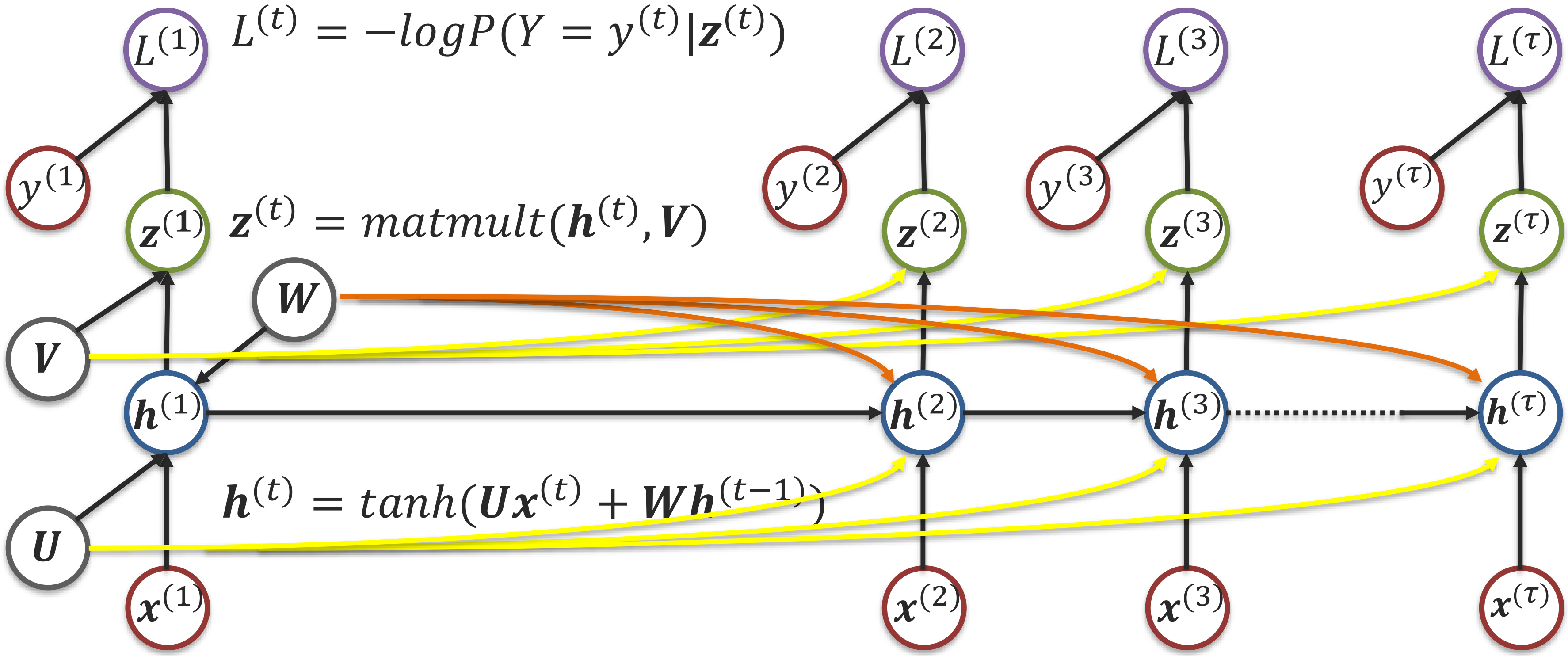
Recurrent Neural Networks

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

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

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

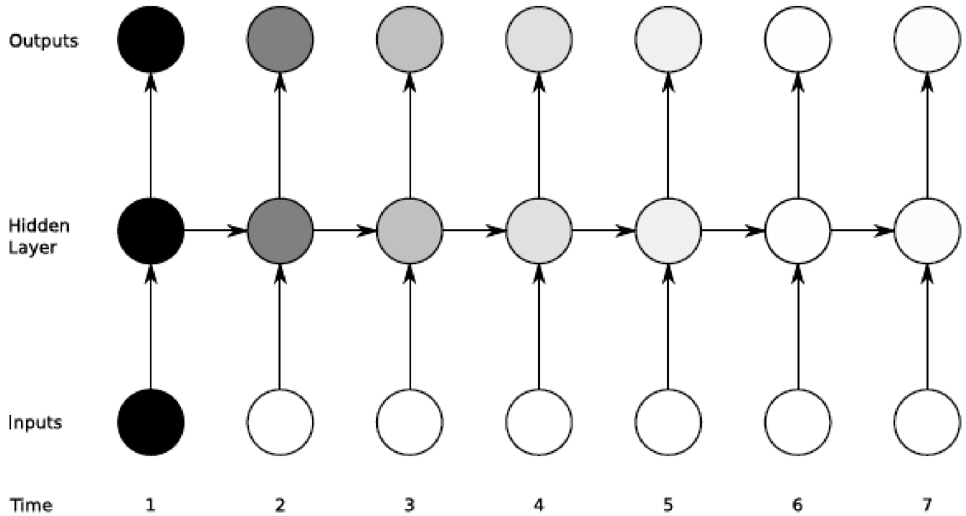
$$\boldsymbol{h}^{(t)} = tanh(\boldsymbol{U}\boldsymbol{x}^{(t)} + \boldsymbol{W}\boldsymbol{h}^{(t-1)})$$



Long-term Dependencies

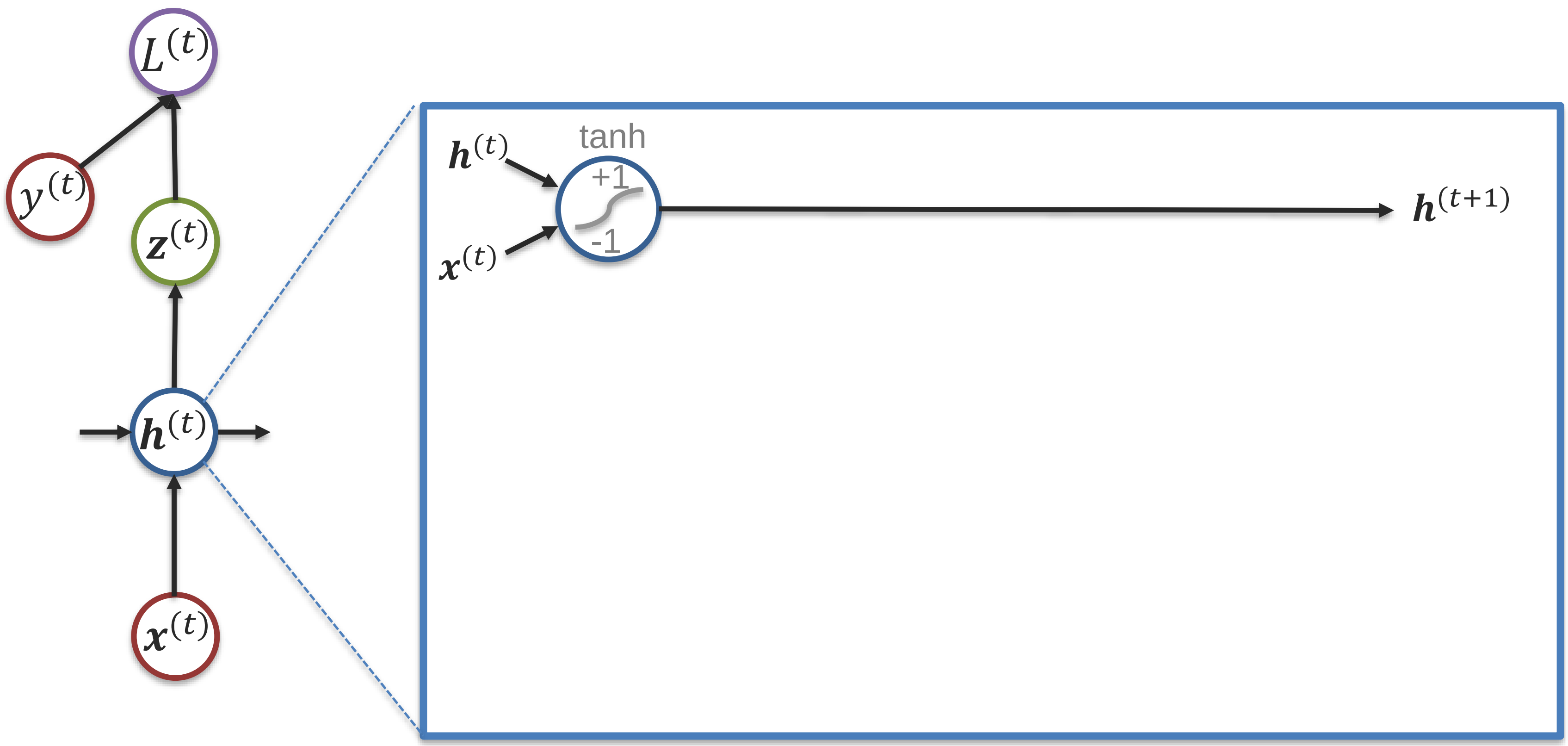
Vanishing gradient problem for RNNs:

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



- 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.

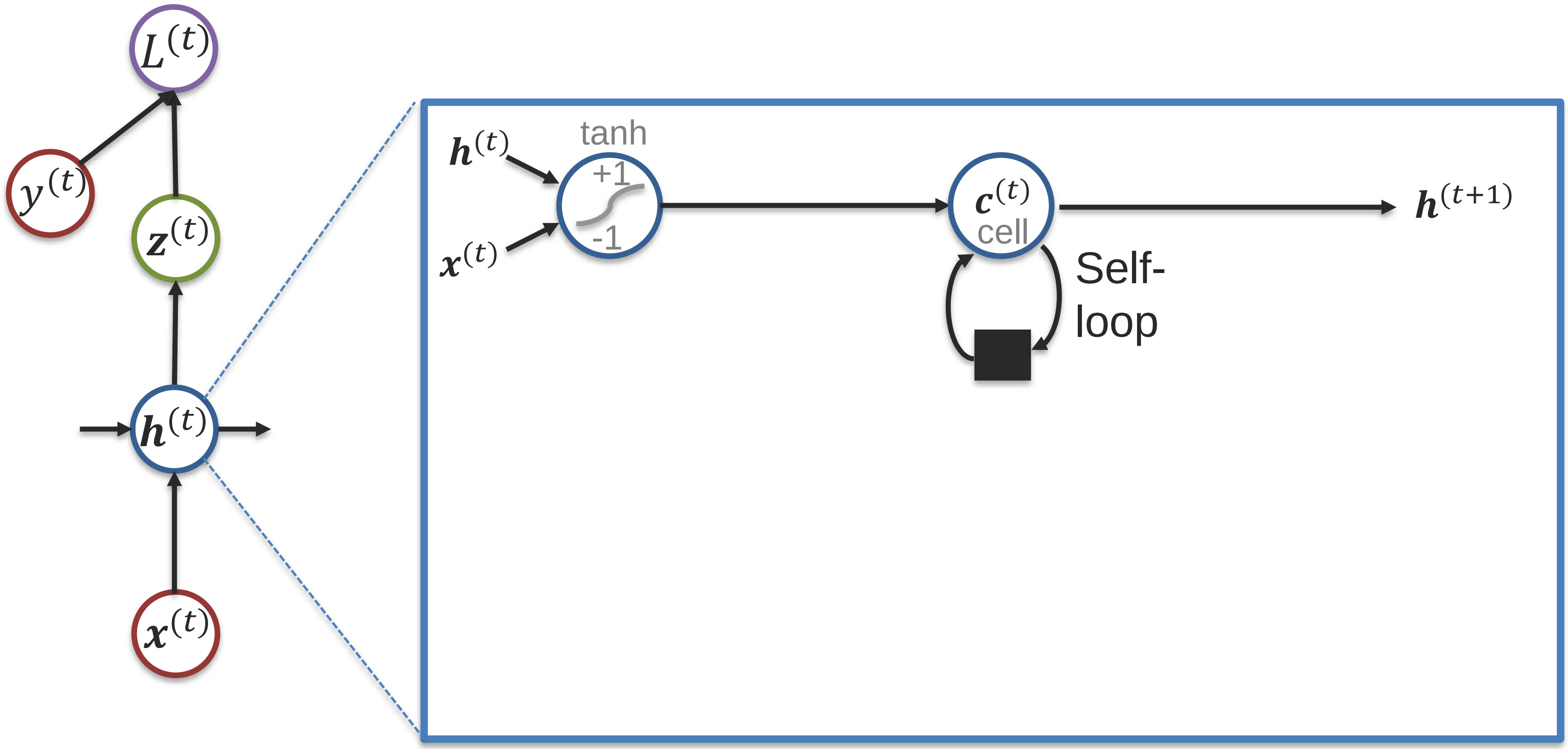
Recurrent Neural Networks



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

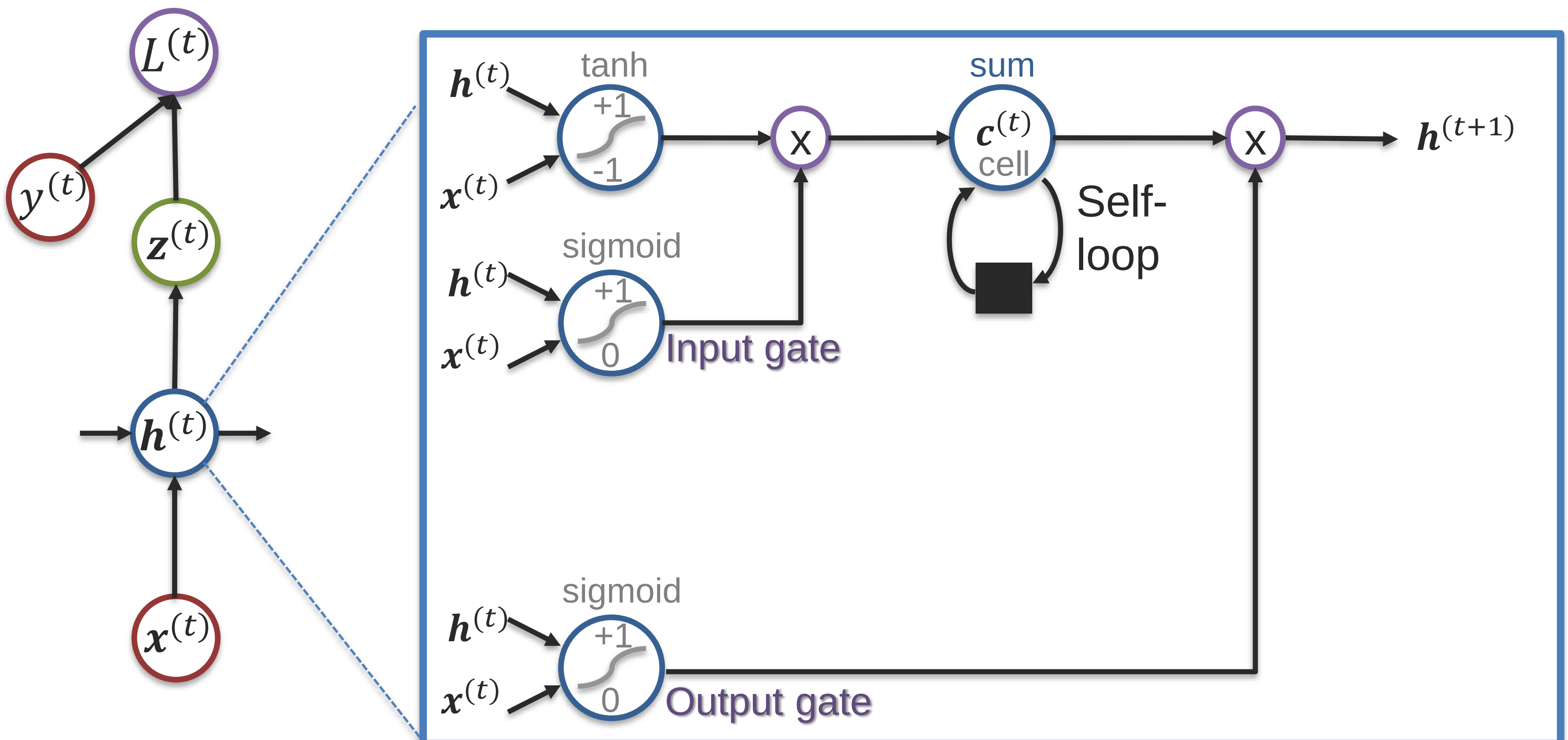
[Hochreiter and Schmidhuber, 1997]

Long Short-Term Memory (LSTM)



LSTM Ideas: (2) Input and Output Gates

[Hochreiter and Schmidhuber, 1997]

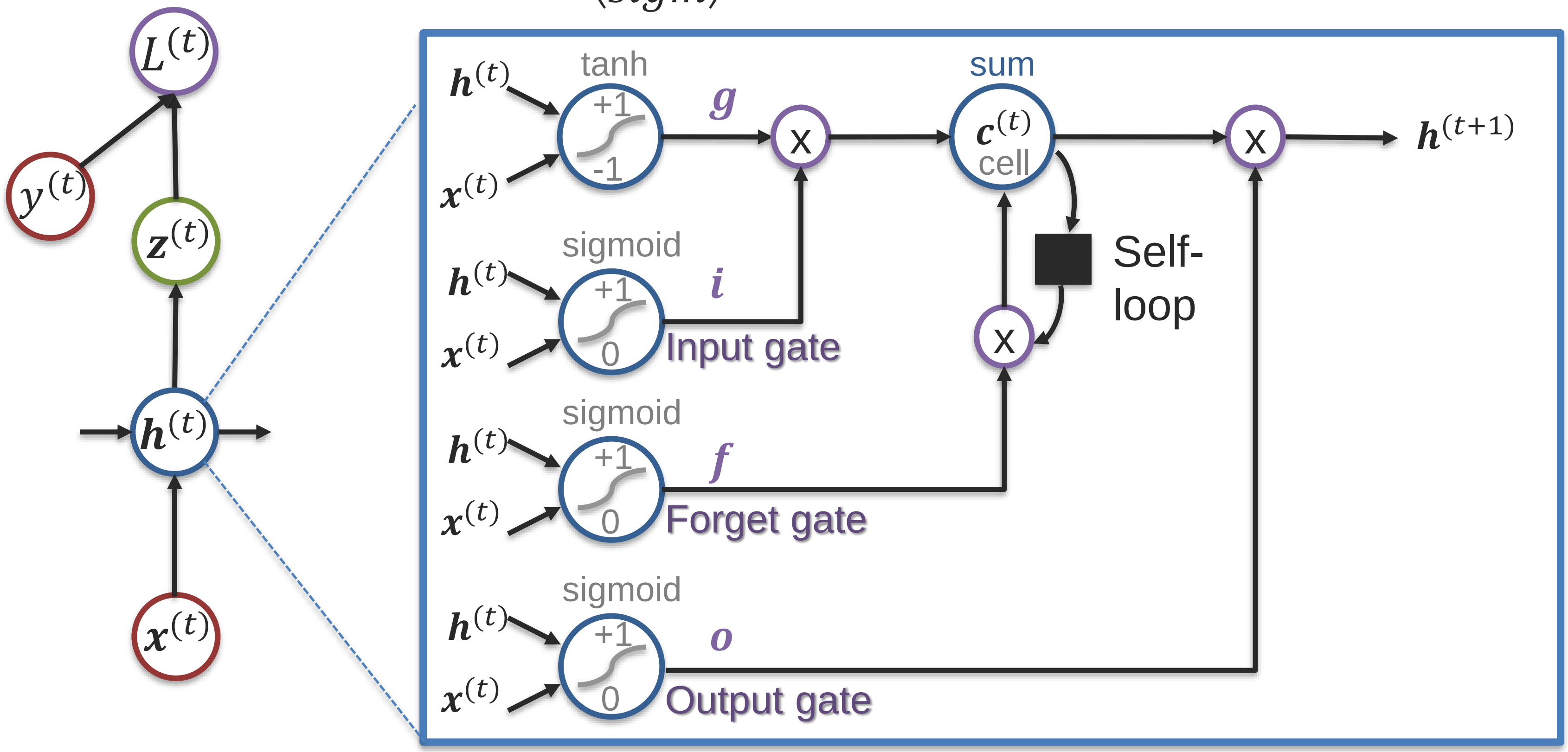


LSTM Ideas: (3) Forget Gate [Gers et al., 2000]

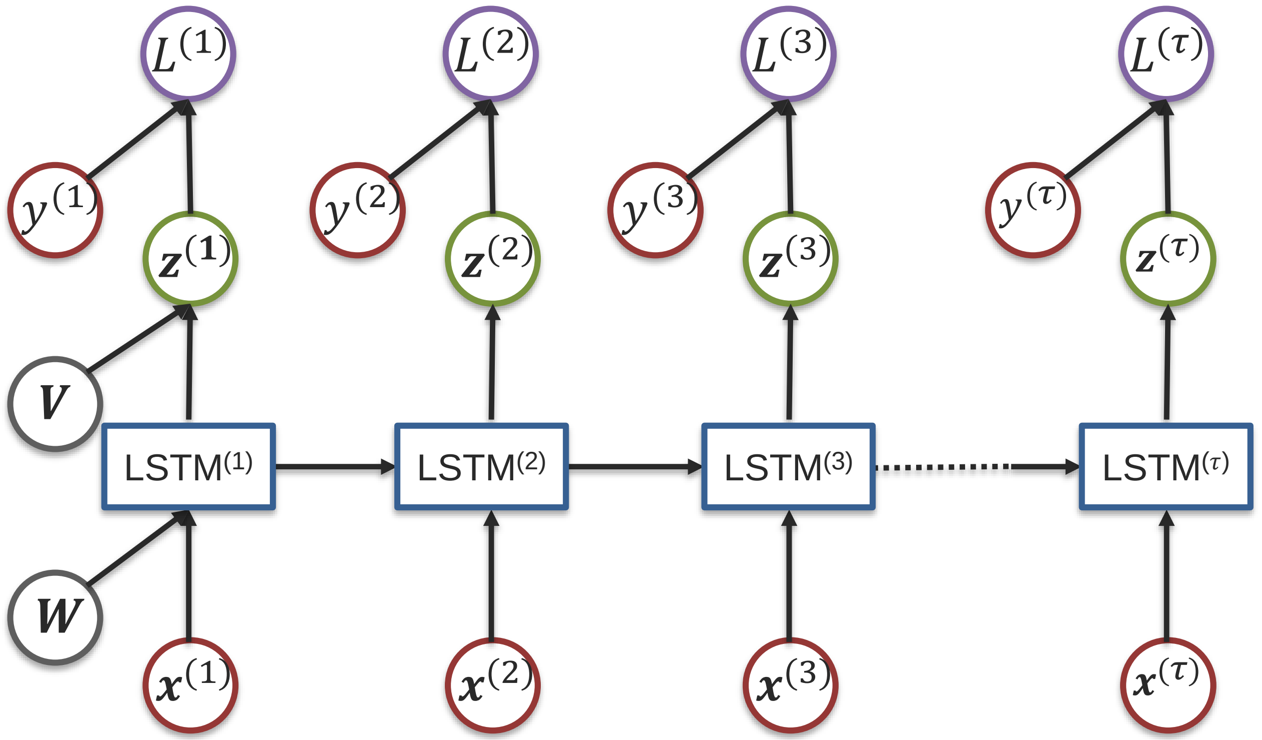
$$\begin{pmatrix} \boldsymbol{g} \\ \boldsymbol{i} \\ \boldsymbol{f} \\ \boldsymbol{o} \end{pmatrix} = \begin{pmatrix} tanh \\ sigm \\ sigm \\ sigm \end{pmatrix} \boldsymbol{W} \begin{pmatrix} \boldsymbol{h}^{(t)} \\ \boldsymbol{x}^{(t)} \end{pmatrix}$$

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

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

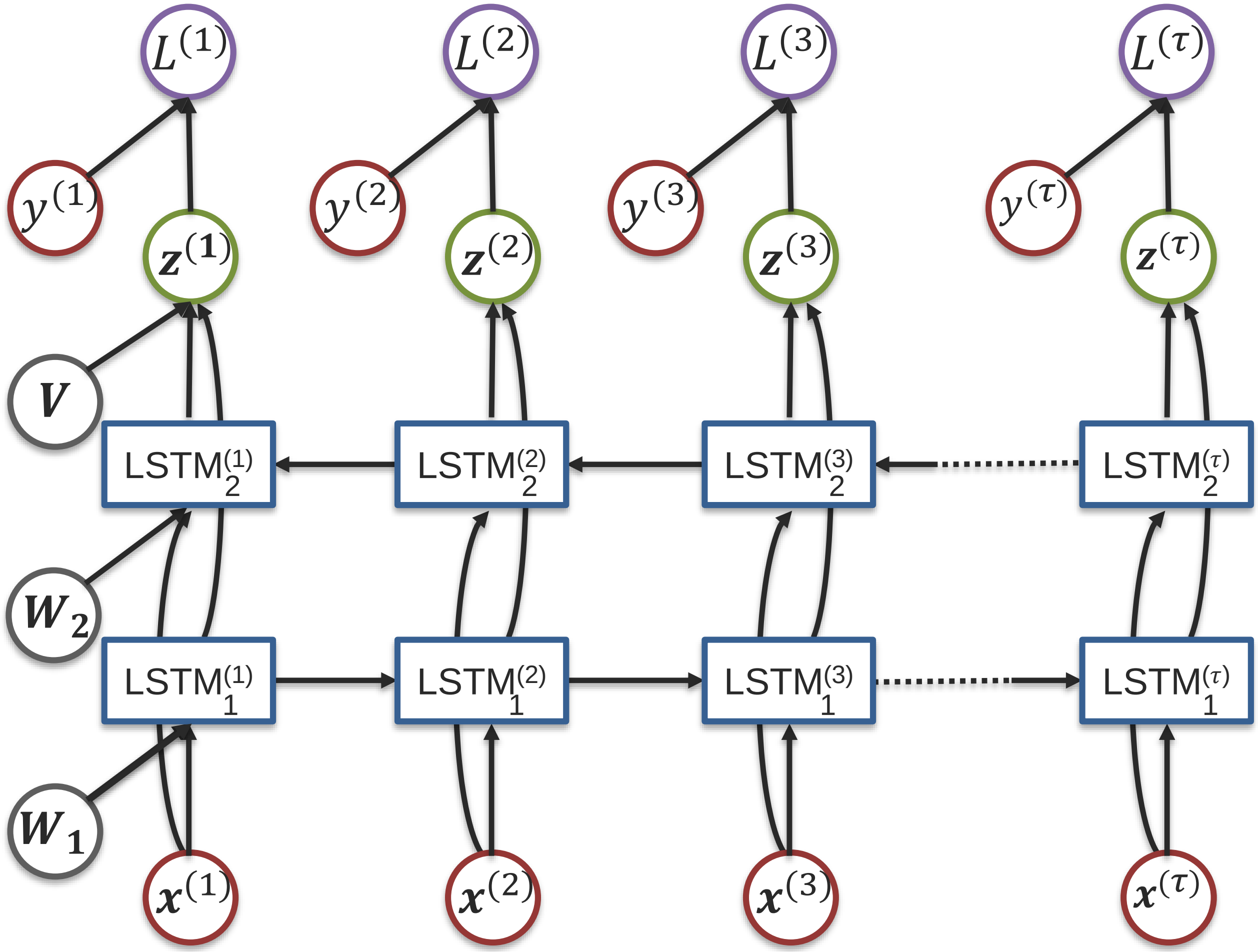


Recurrent Neural Network using LSTM Units



Gradient can still be computer using backpropagation!

Bi-directional LSTM Network



Deep LSTM Network