Language Technologies Institute

Carnegie Mellon University

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 (10/3 and 10/5)
  - First project report (Sunday 10/8)
- Second project assignment
  - Midterm presentations (11/7 and 11/9)
  - Midterm report (Sunday 11/12)
- Final project assignment
  - Final presentation (12/4 & 12/5)
  - Final report (Sunday 12/10)

Proposal Presentation (10/3/2017 and 10/5/2017)

- 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 (similar to pre-proposal)
  - 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 10/6 (on Gradescope)

Project Proposal Report – Due on 10/8/17

- 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 – Due on 10/8/17

- 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 – Due on 10/8/17

- Part 3
  - **Language Modality Exploration:**
    - Explore neural language models on your dataset (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., AlexNet or 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
- Sequence modeling tasks
- Recurrent neural networks
- Backpropagation through time
- Gates recurrent neural networks
  - Long Short-Term Memory (LSTM) model

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

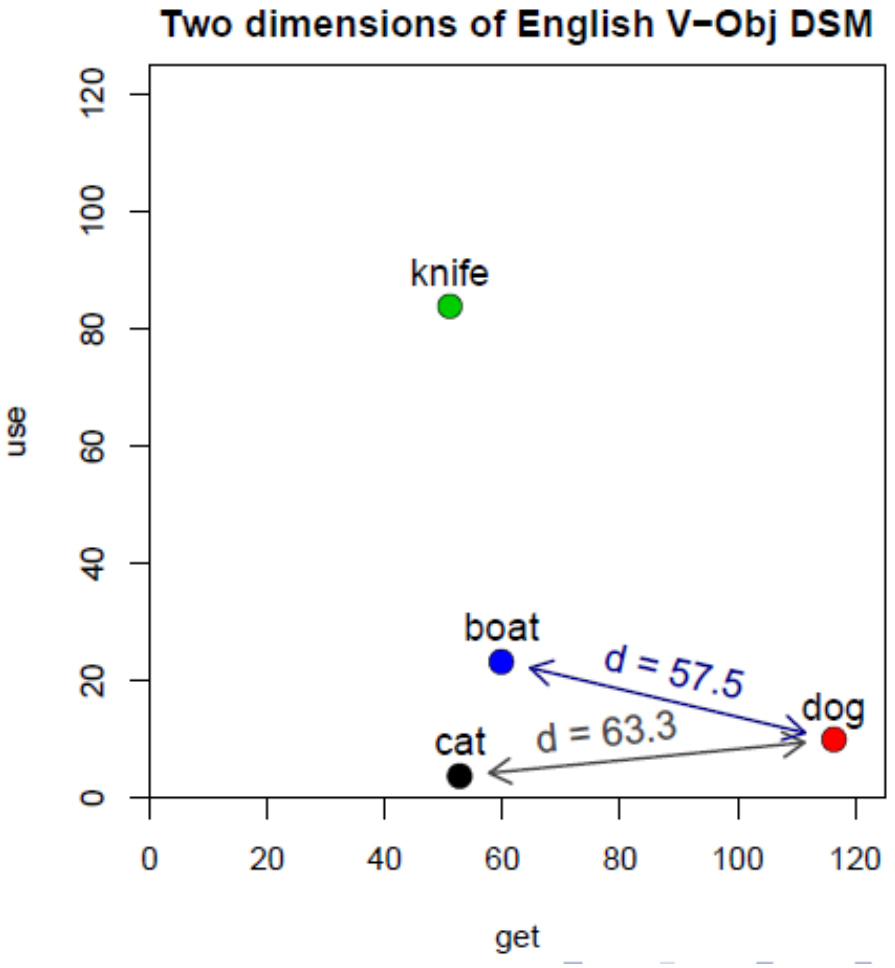
- row vector $\mathbf{x}_{dog}$ describes usage of word *dog* in the corpus
- can be seen as coordinates of point in *n*-dimensional Euclidean space R^n

	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**

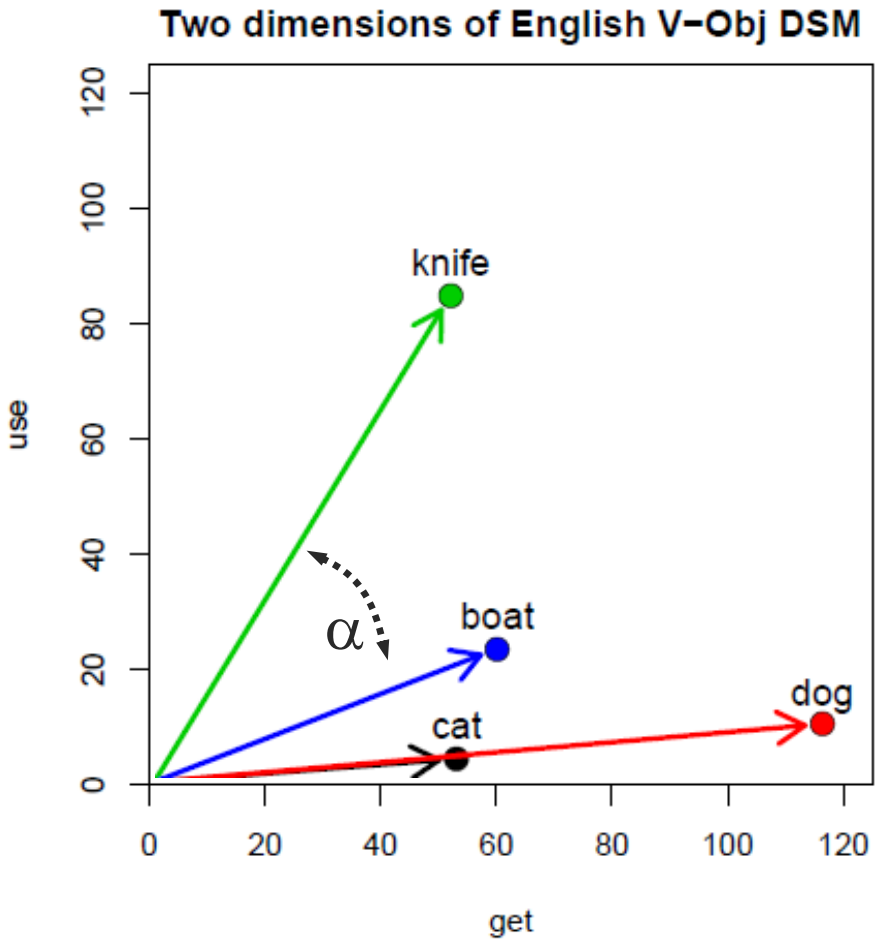
Distance and similarity

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

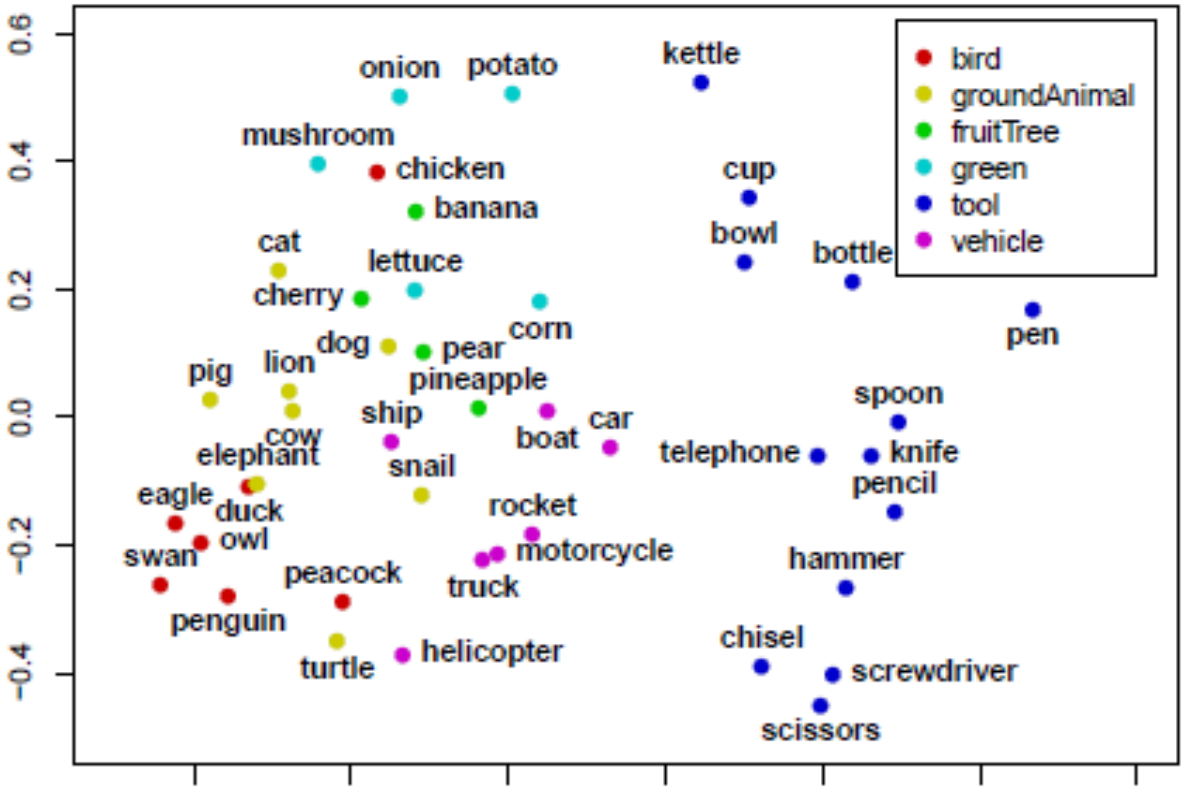


Angle and similarity

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



Semantic maps



Learning Neural Representations of Words

How to learn (word) features/representations?

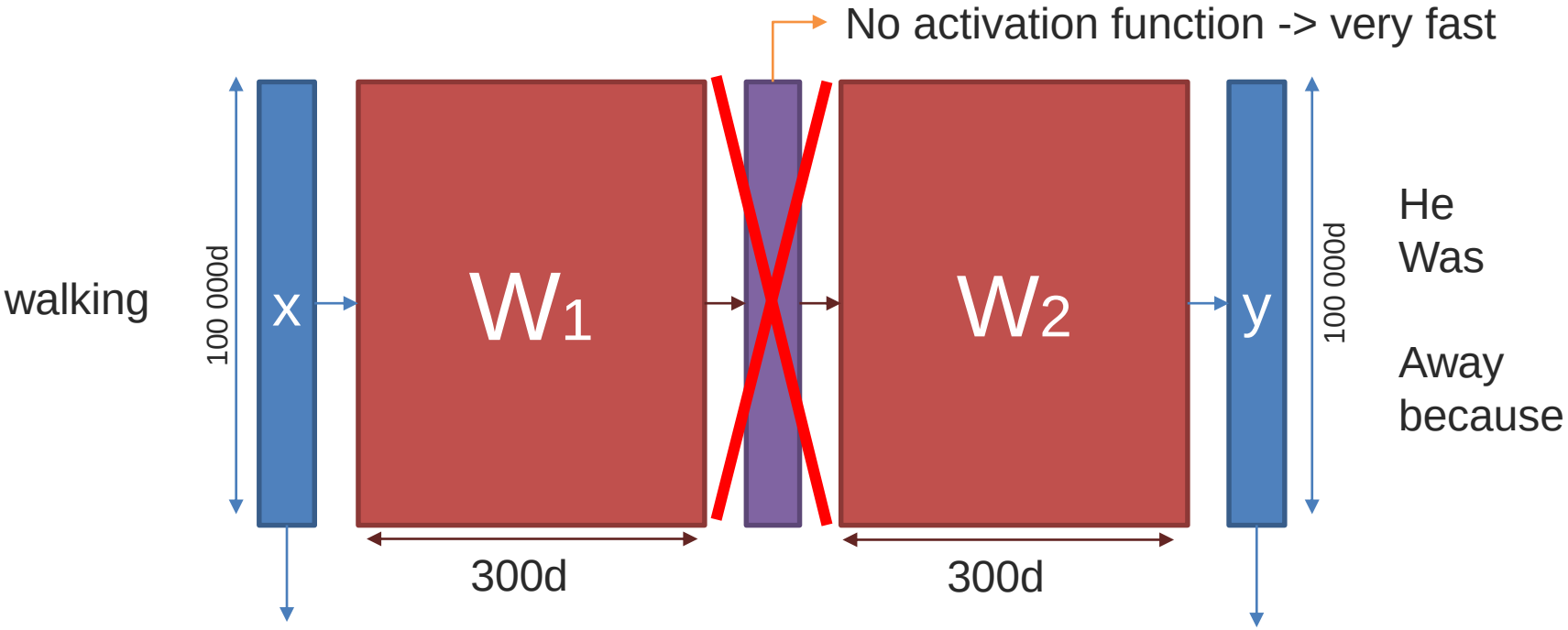
Distribution hypothesis: Approximate the word meaning by its surrounding words

Words used in a similar context will lie close together

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:

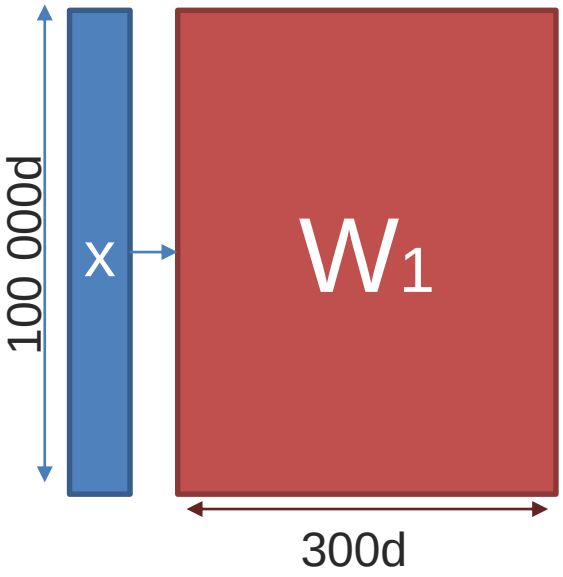
Classic NLP: 100 000 dimensional vector

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

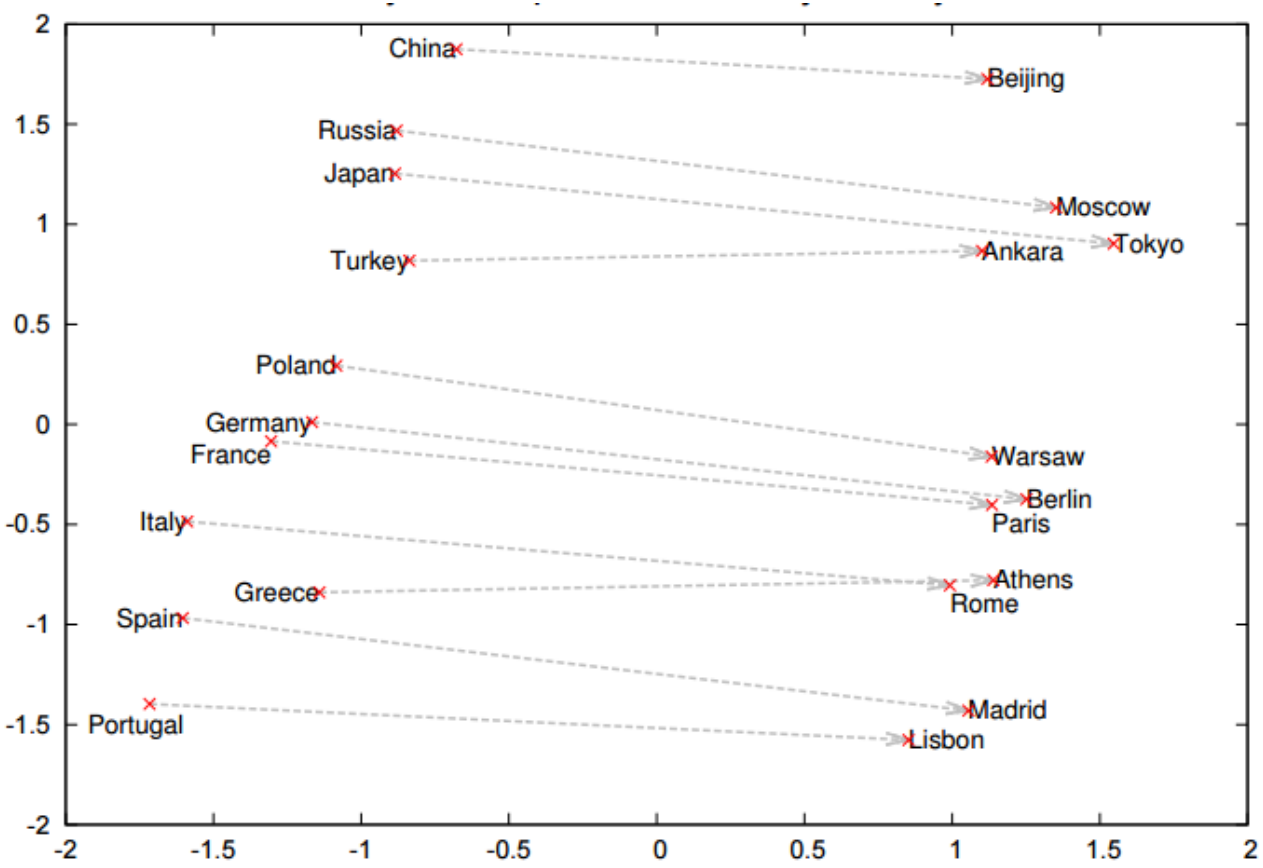
Vector space models of words

- 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:

Vec(king) – vec(man) + vec(woman) ≈ 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 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

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

- 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

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

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

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

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

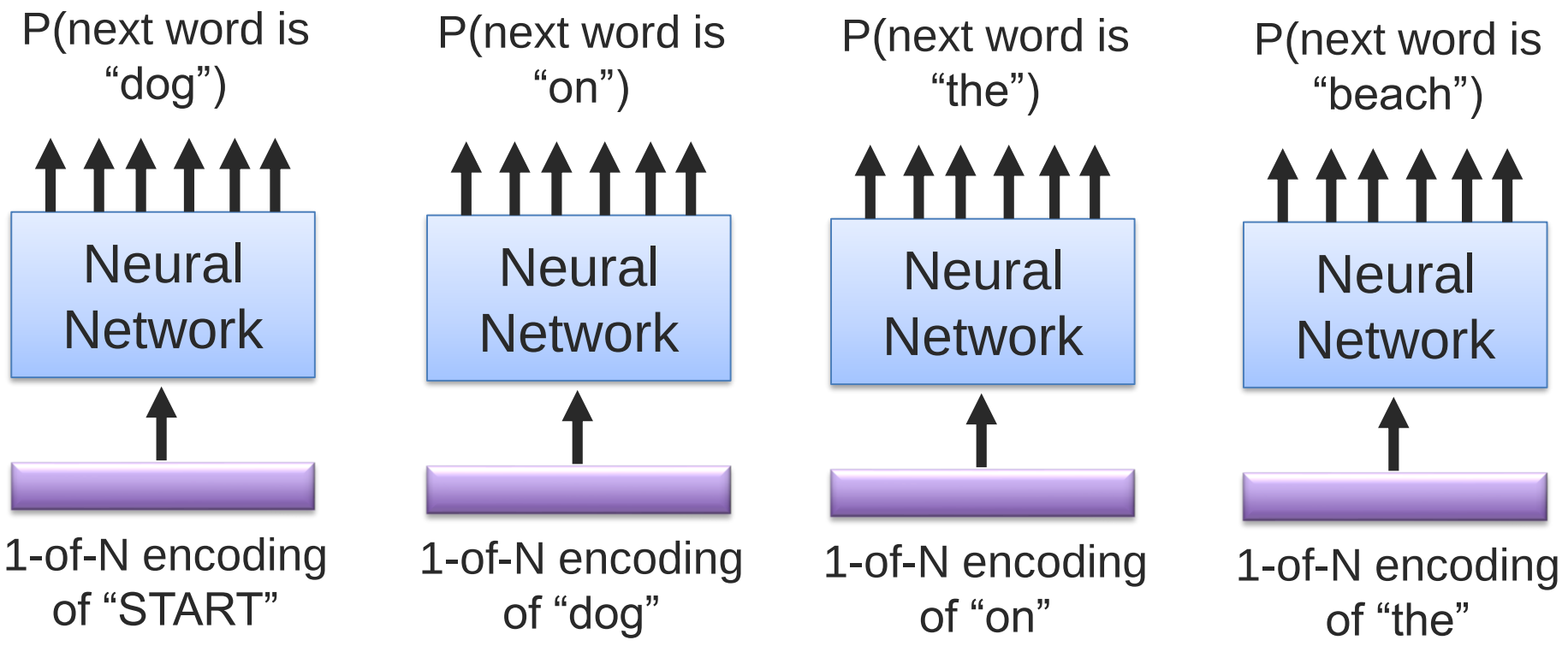
$$\arg\max_{wordsequence} P(acoustics \mid wordsequence) \times P(wordsequence)$$

Language model

Neural-based Unigram Language Model (LM)

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

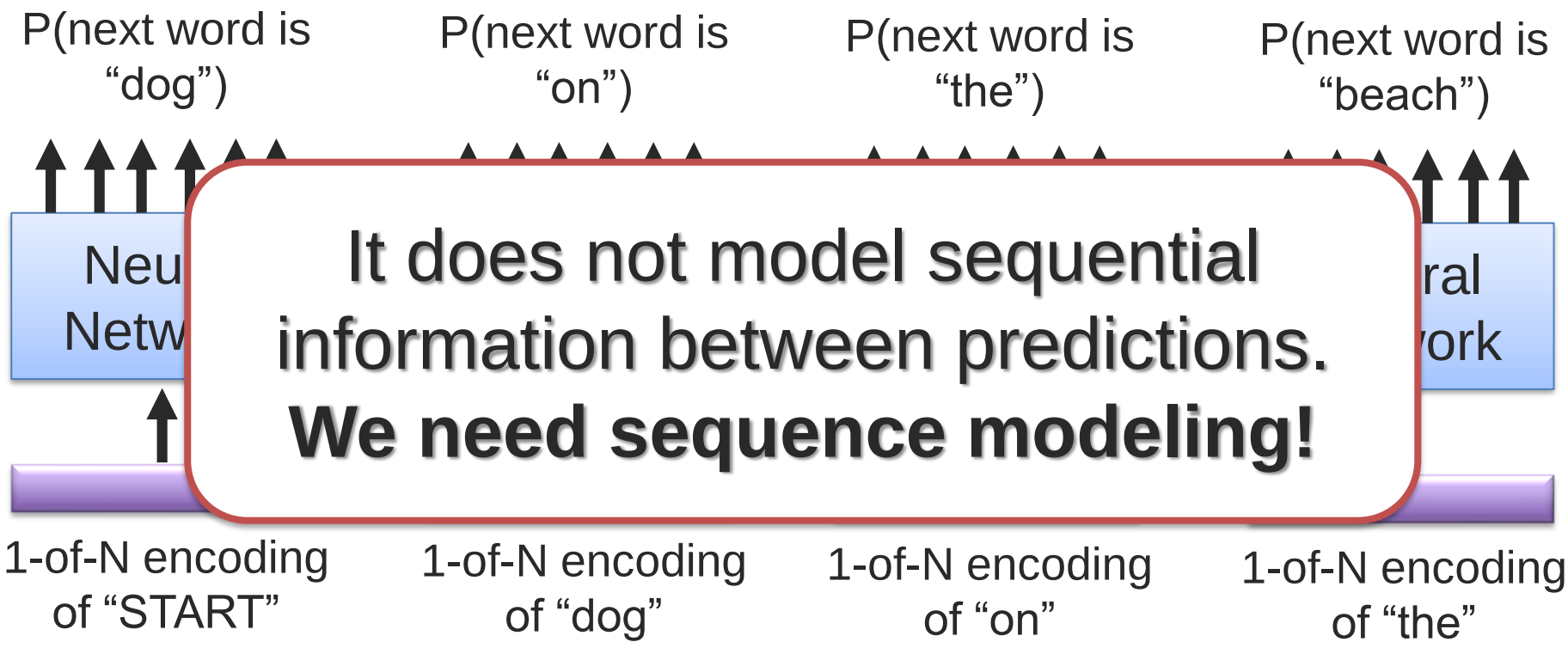
P(b|a): not from count, but the NN that can predict the next word.



Neural-based Unigram Language Model (LM)

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

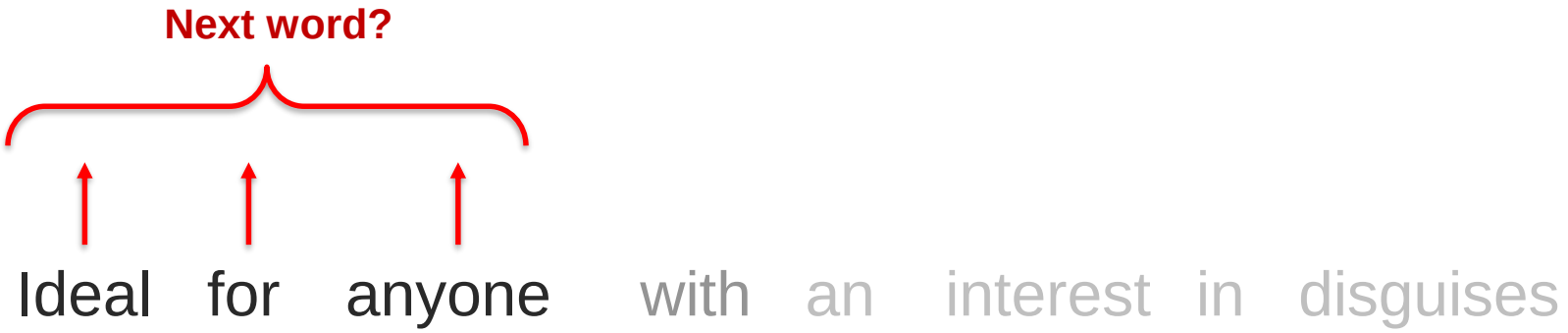
P(b|a): not from count, but the NN that can predict the next word.



Sequence Modeling Tasks

Sequence Modeling: Language Model

Prediction → Language Model



Sequence Modeling: Sequence Prediction

Prediction →

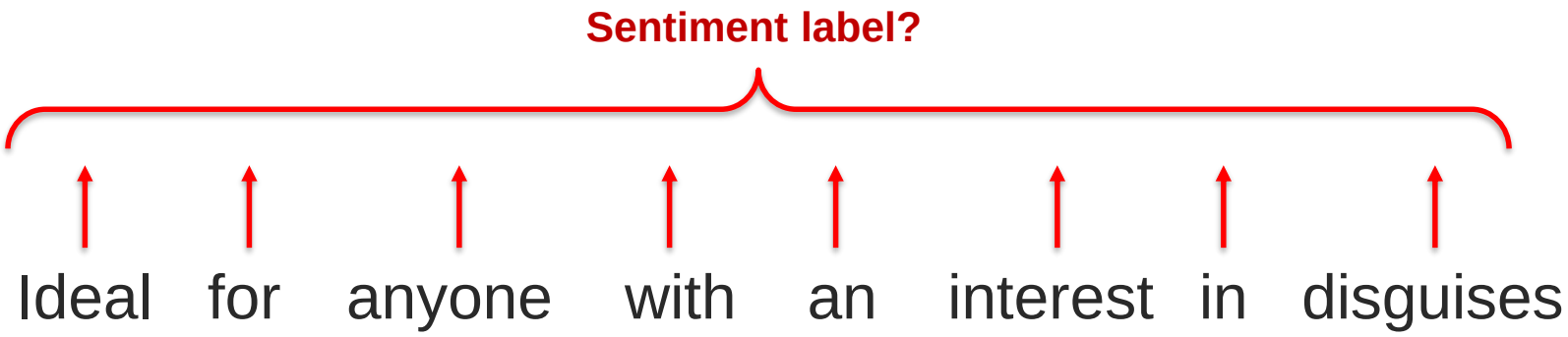
Part-of-speech ?
(noun, verb,…)

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

Sequence Modeling: Sequence Label Prediction

Prediction →

Sentiment ?
(positive or negative)



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

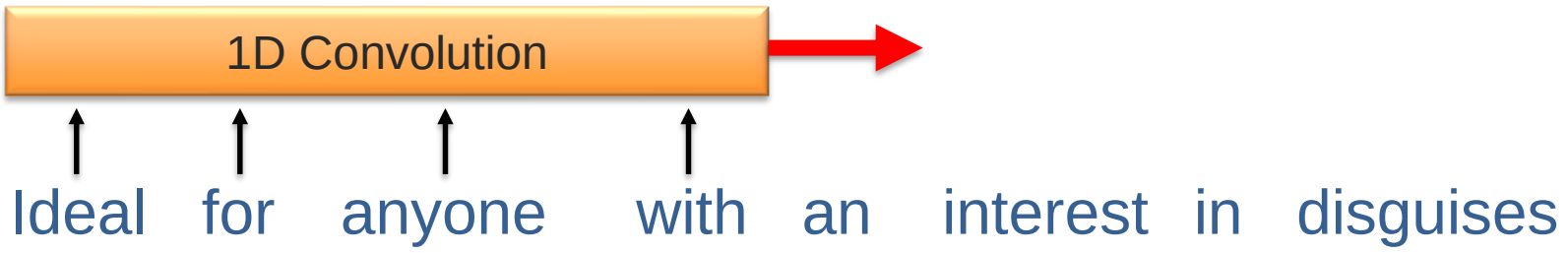
Sequence Modeling

- 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:

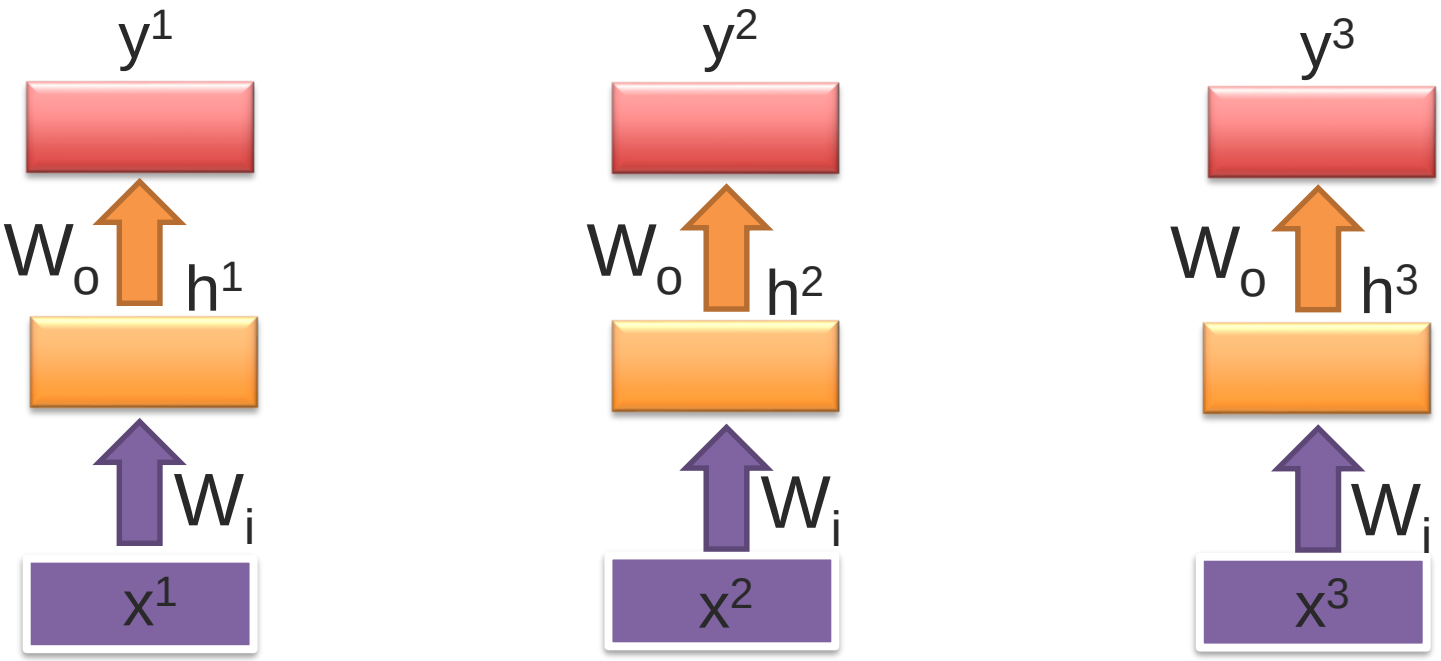
- Sequences of variable lengths (e.g., sentences)
- Keep the number of parameters at a minimum
- Take advantage of possible redundancy

Recurrent Neural Networks

Sequence Prediction (or Unigram Language Model)

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

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



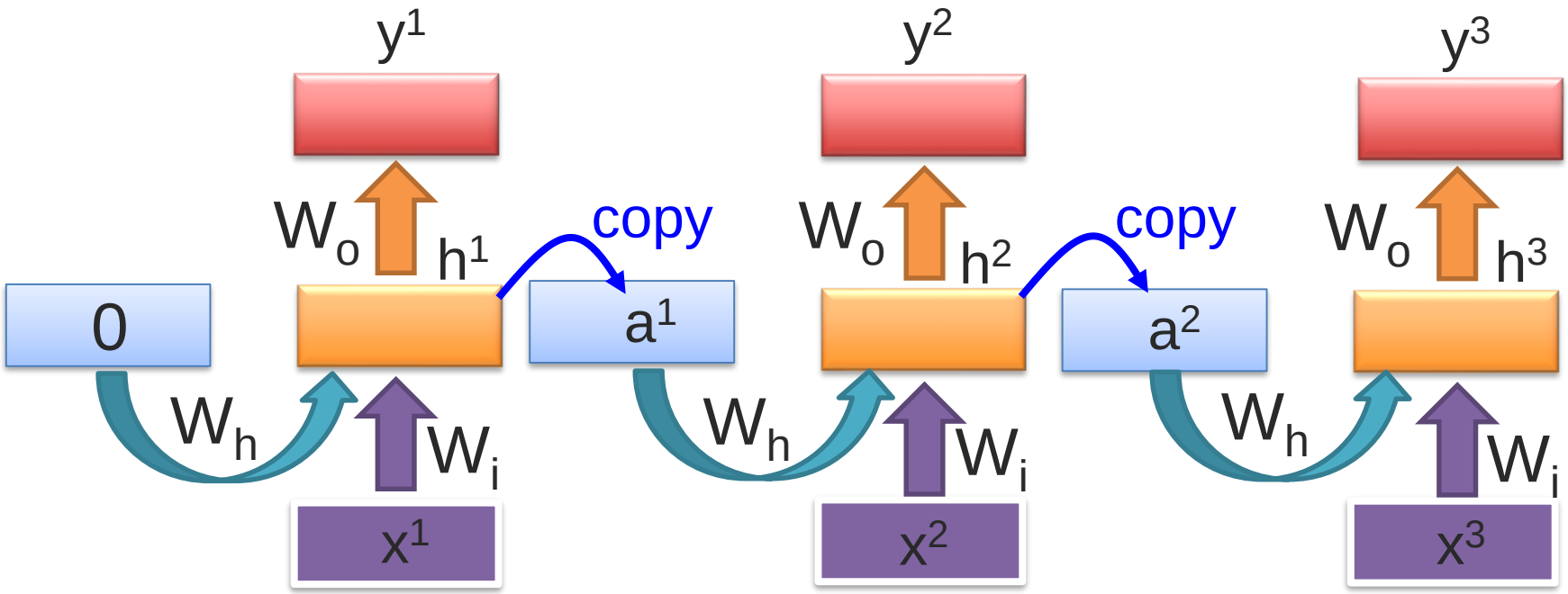
How can we include temporal dynamics?

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)

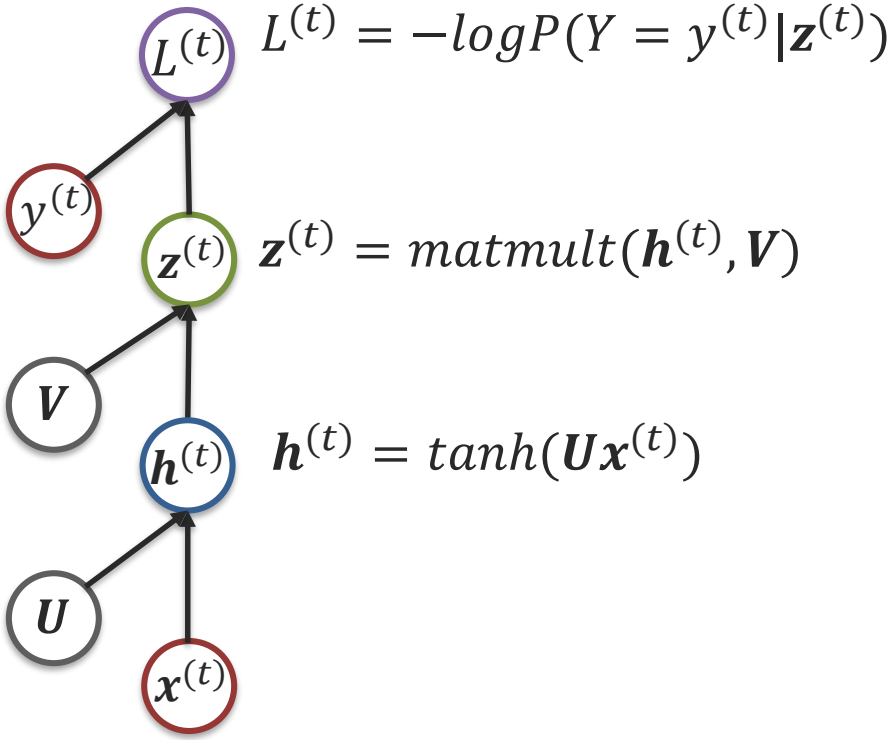


The same model parameters are used again and again.

Can be trained using backpropagation

Recurrent Neural Network

Feedforward Neural Network



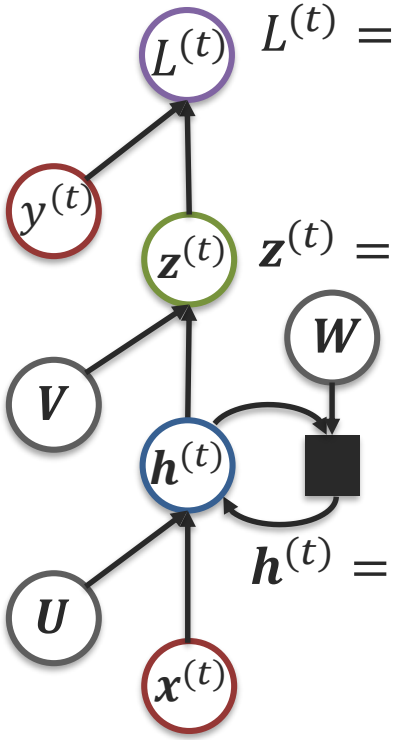
$$L^{(t)} = -logP(Y = y^{(t)}|\boldsymbol{z}^{(t)})$$

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

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

Recurrent Neural Networks

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



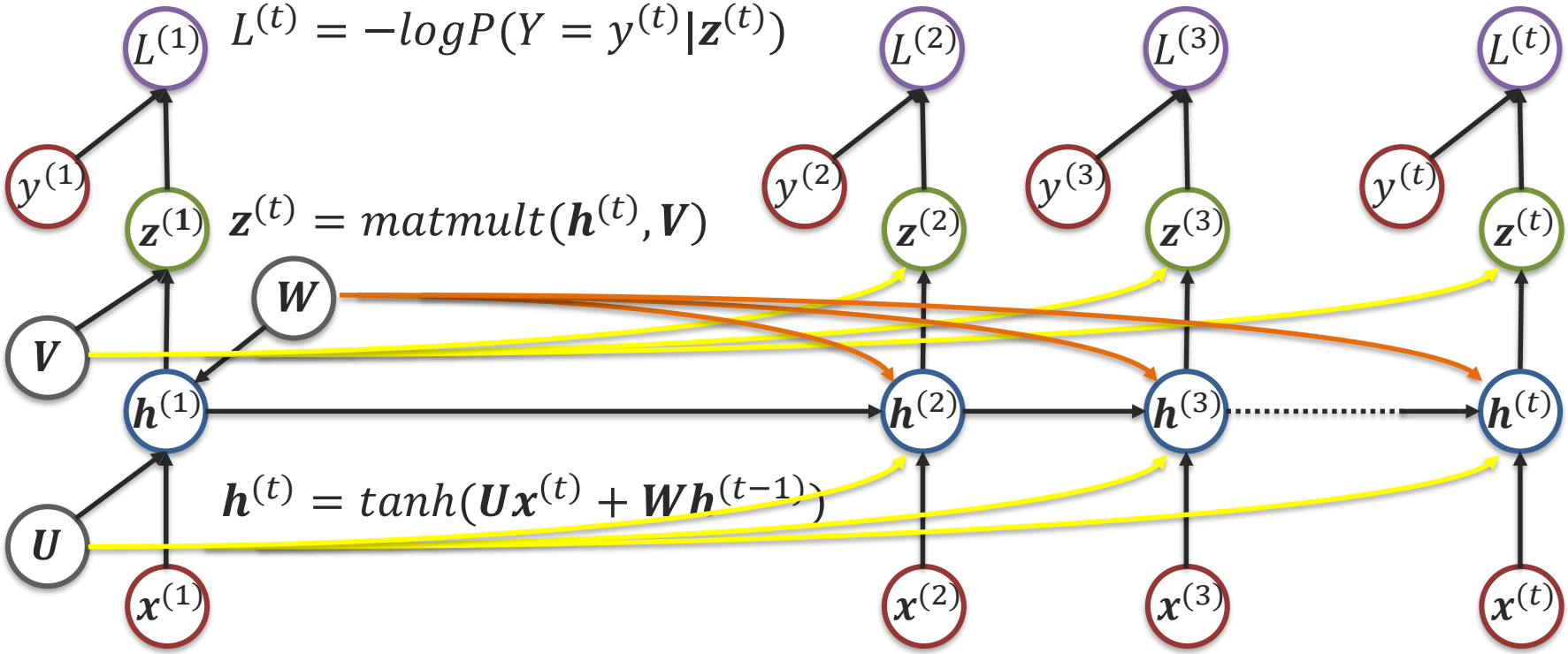
$$L^{(t)} = -logP(Y = y^{(t)}|\boldsymbol{z}^{(t)})$$

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

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

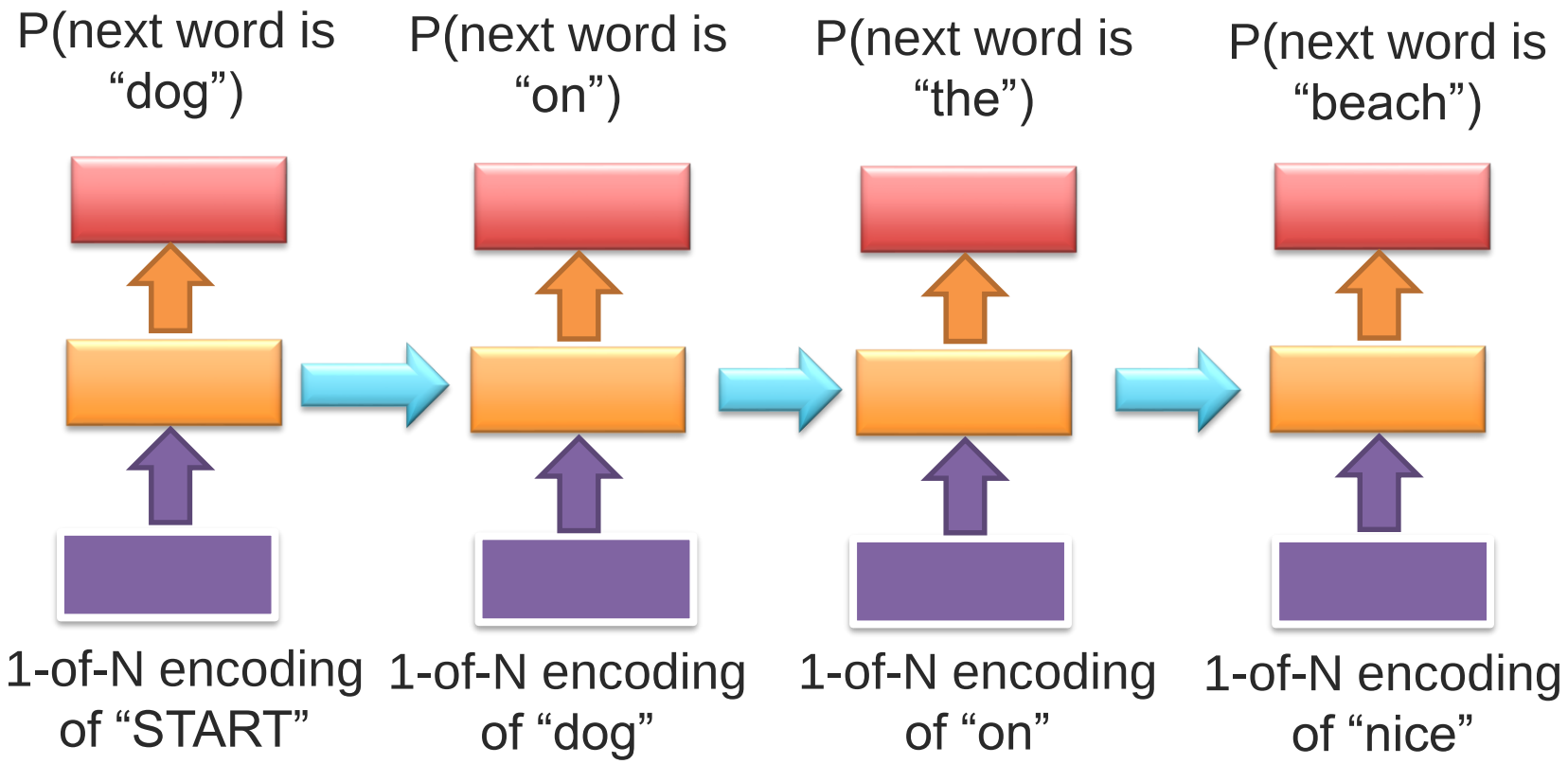
Recurrent Neural Networks - Unrolling

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



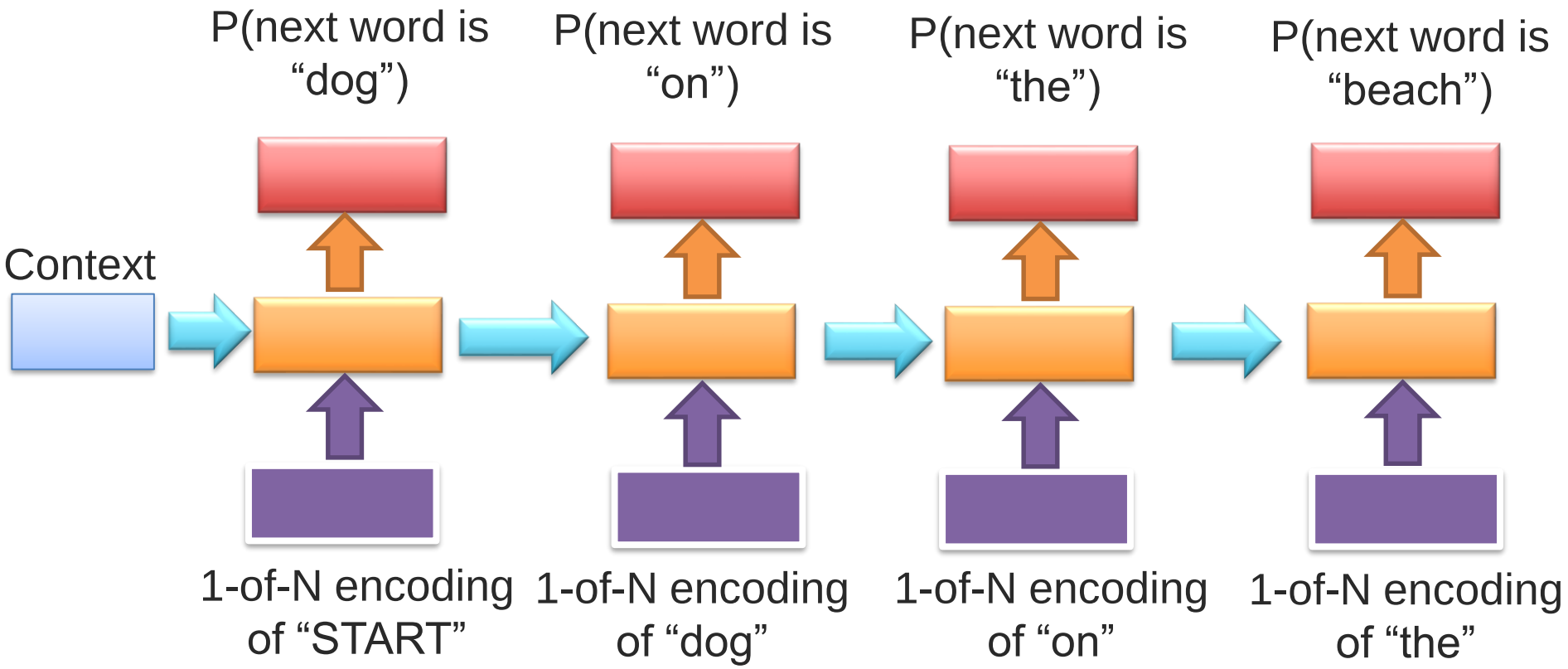
Same model parameters are used for all time parts.

RNN-based Language Model



- Models long-term information

RNN-based Sentence Generation (Decoder)

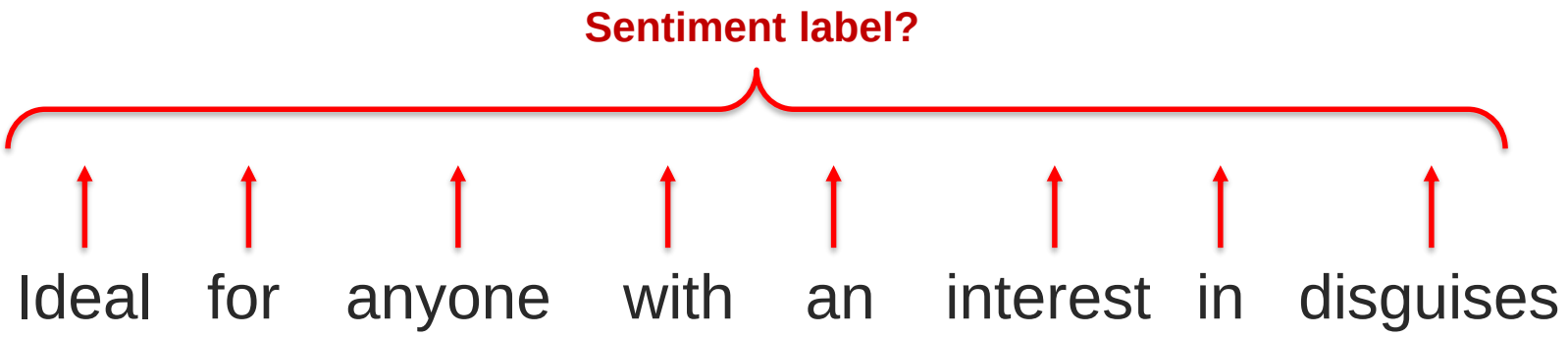


- Models long-term information

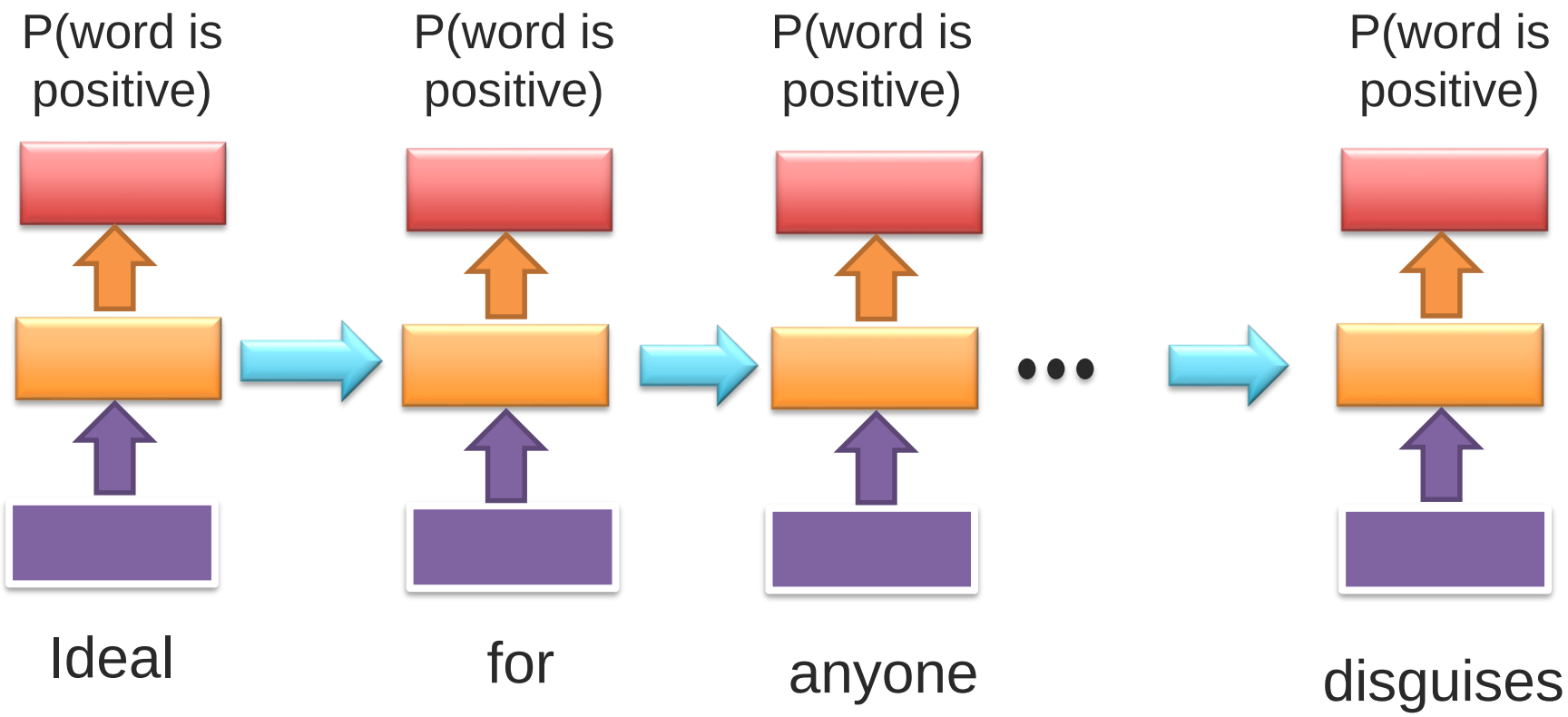
Sequence Modeling: Sequence Prediction

Prediction →

Sentiment ?
(positive or negative)

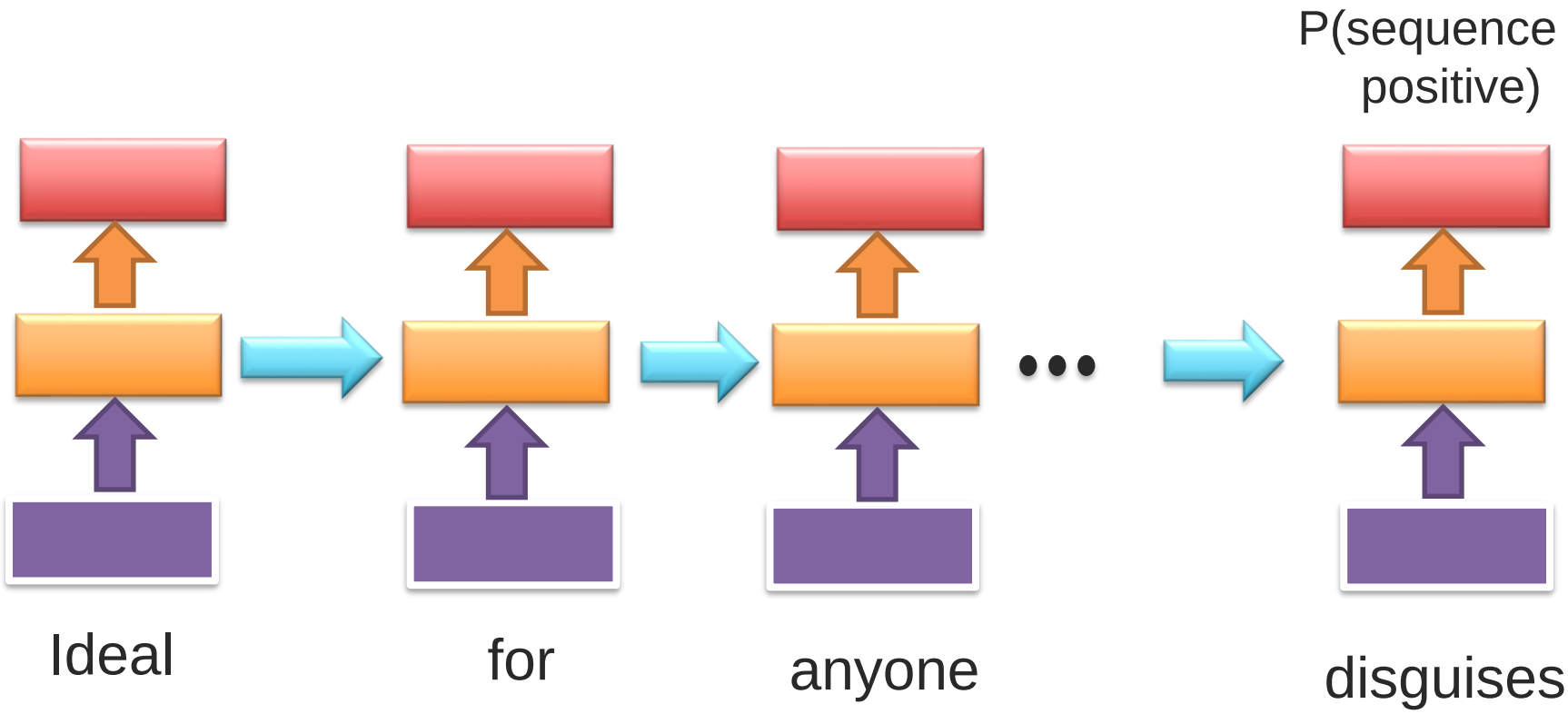


RNN for Sequence Prediction



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

RNN for Sequence Prediction



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

Sequence Modeling: Sequence Representation

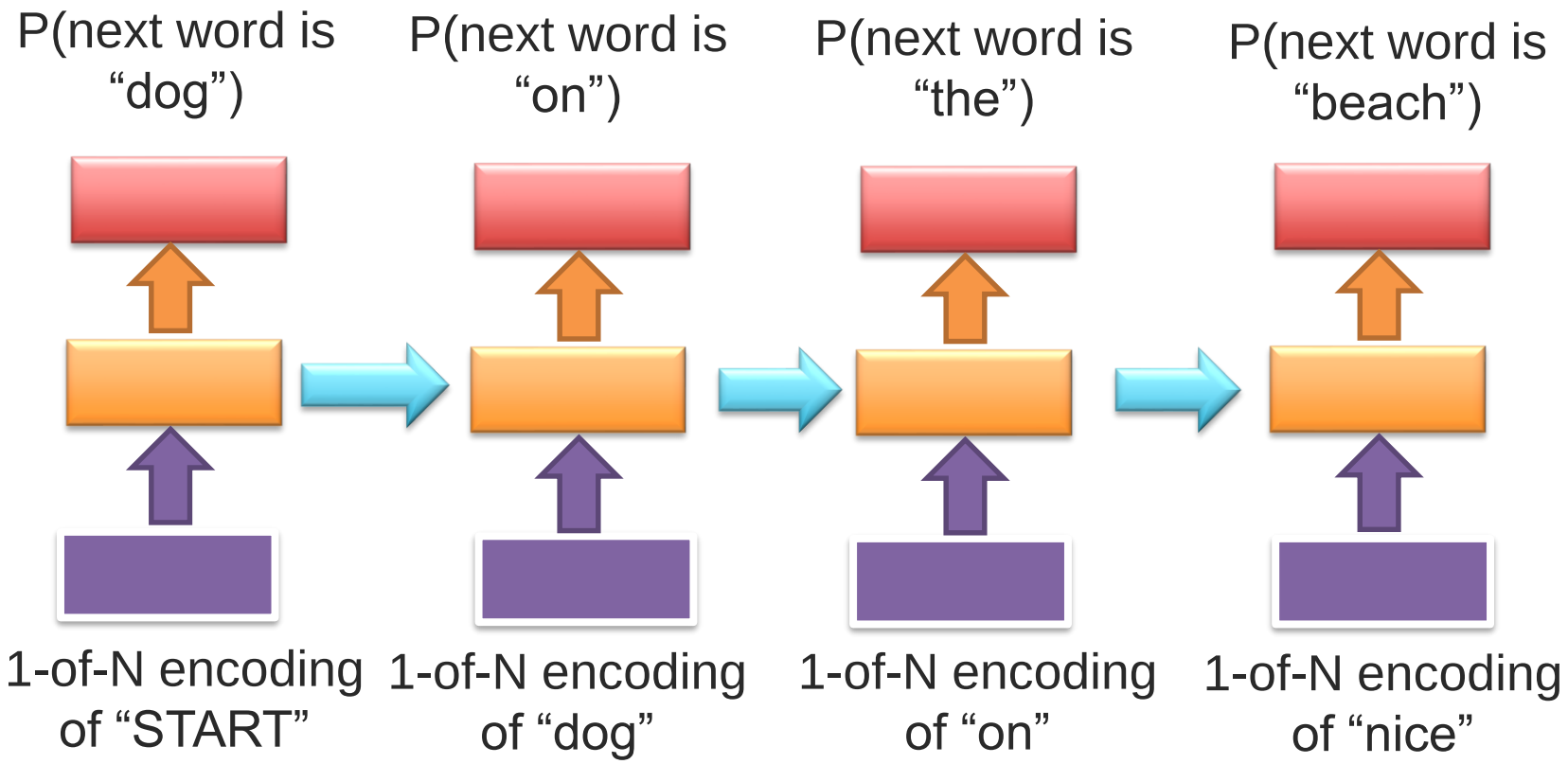
Learning →

Sequence representation

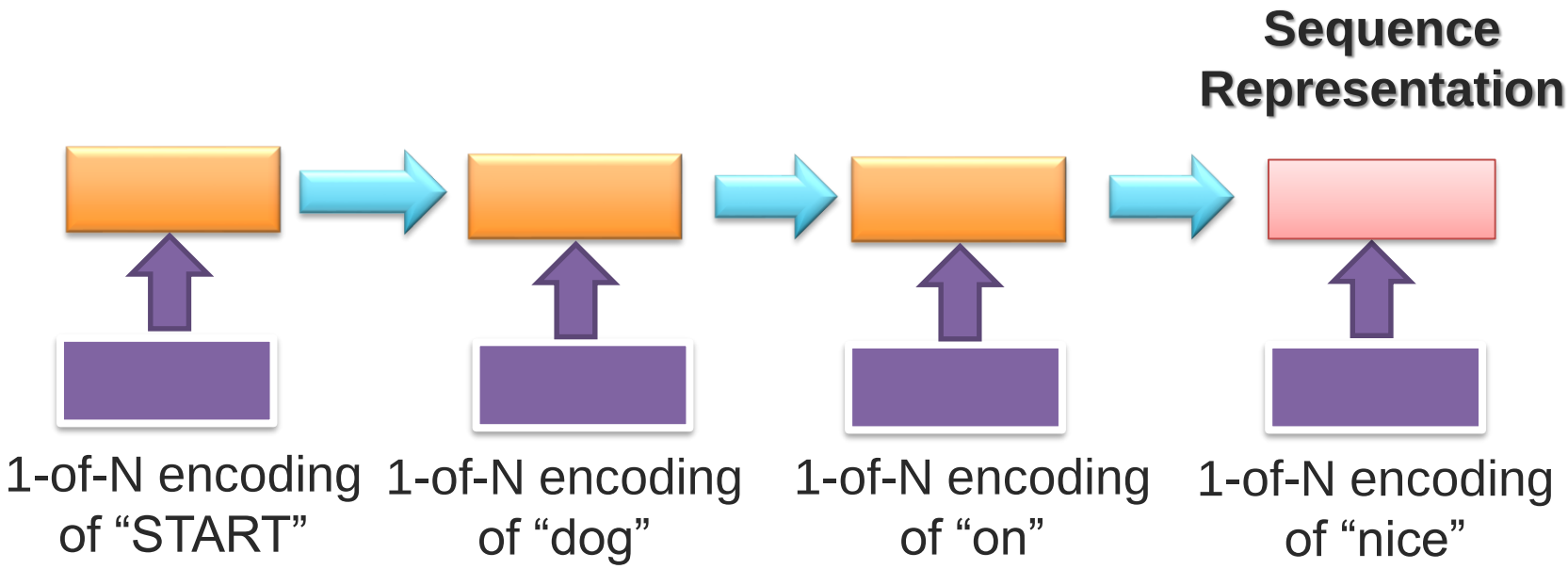
[0,1; 0,0004; 0;….; 0,01; 0.09; 0,05]

Ideal for anyone with an interest in disguises

RNN for Sequence Representation

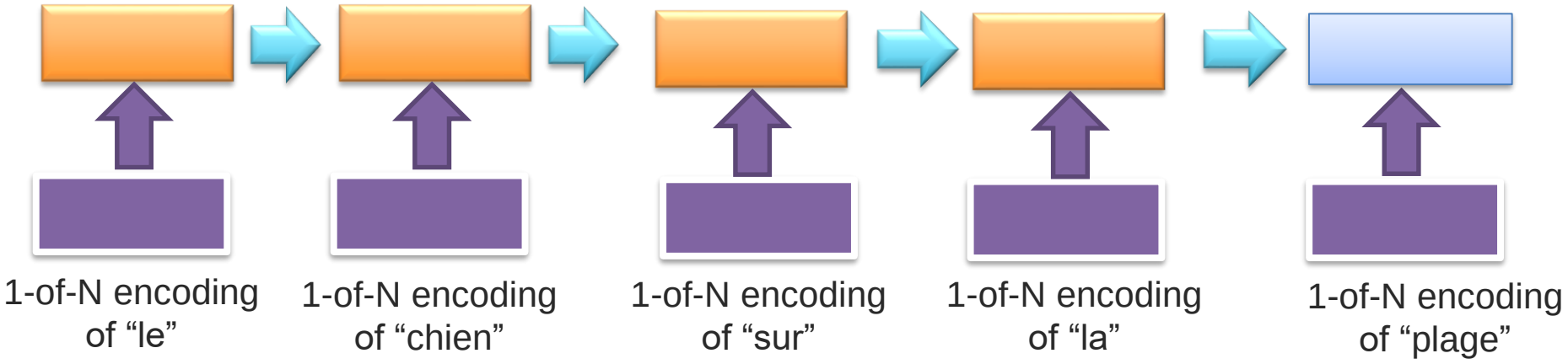


RNN for Sequence Representation (Encoder)

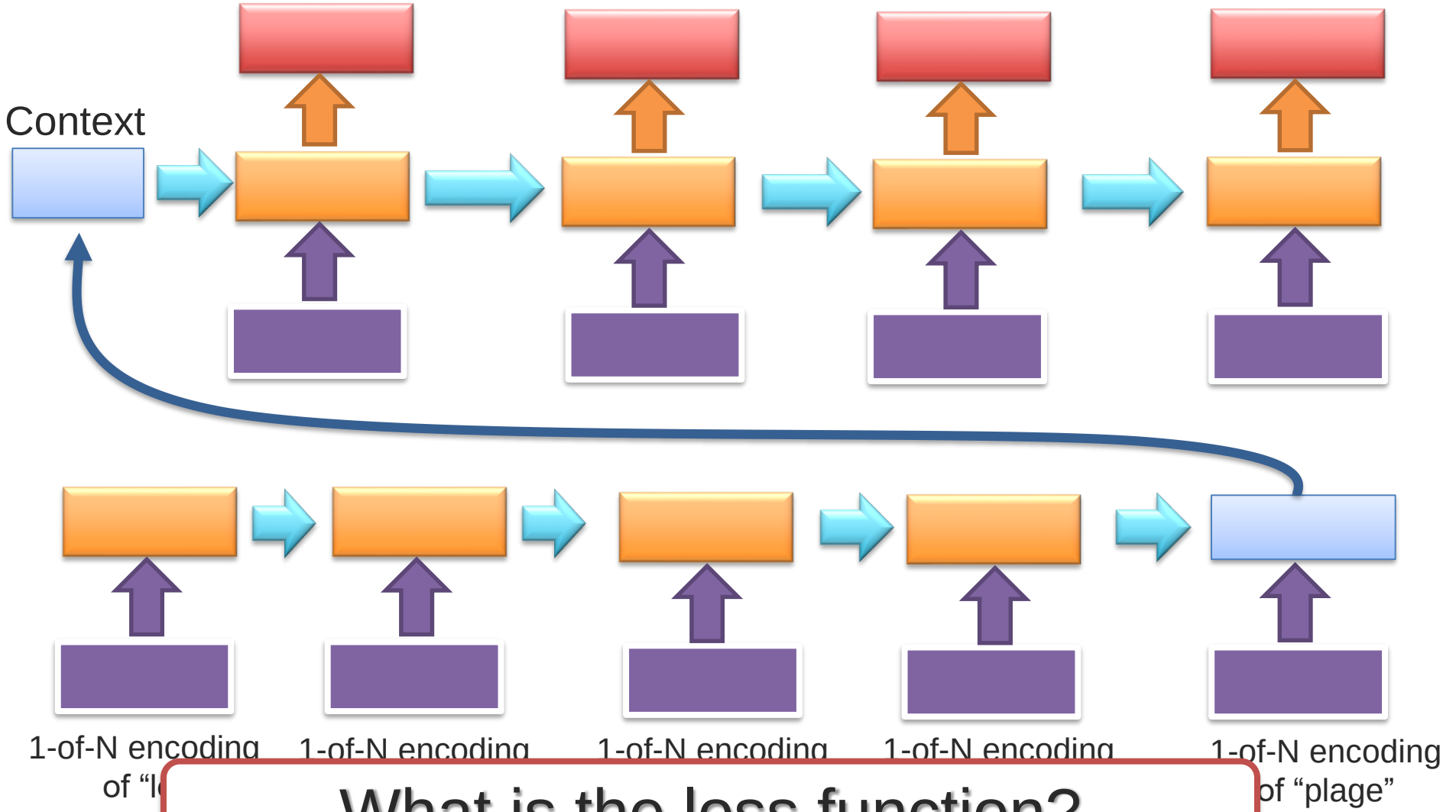


RNN-based for Machine Translation

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:

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

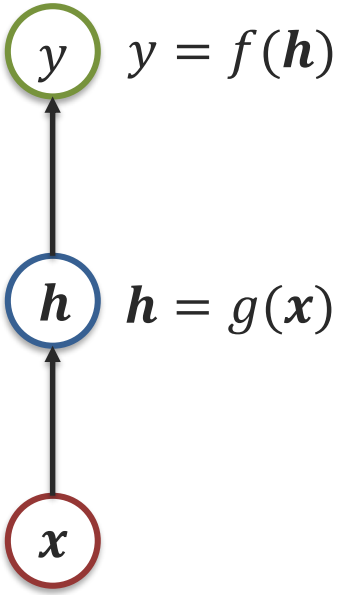
Gradient

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

"backprop" Gradient

"local" Jacobian

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



Backpropagation Algorithm

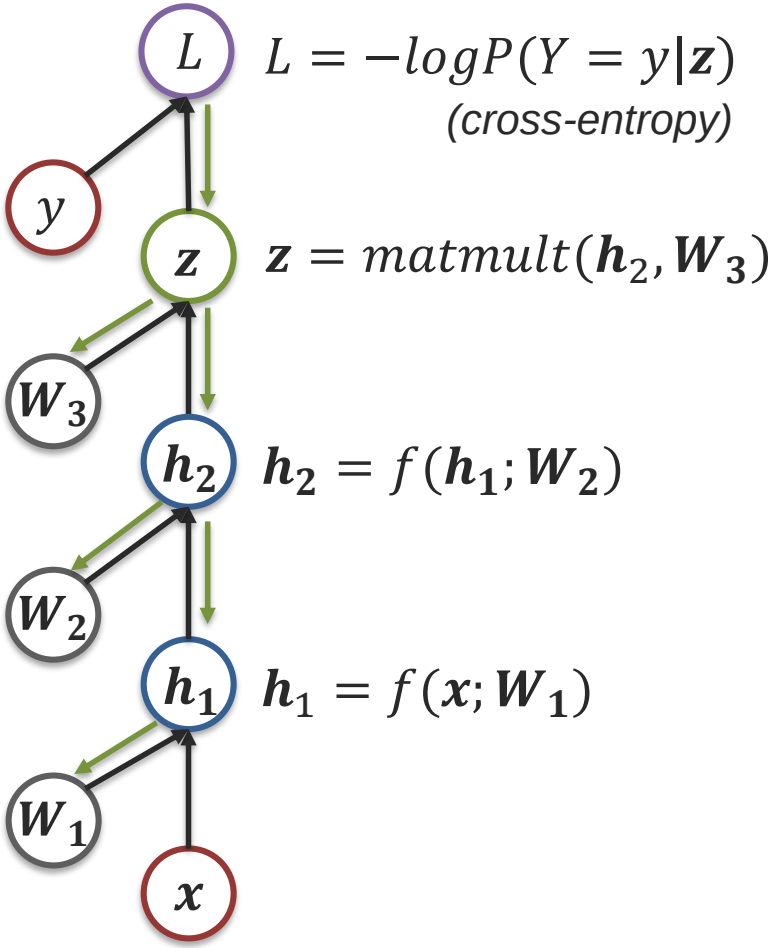
Forward pass

- Following the graph topology, compute value of each unit

Backpropagation pass

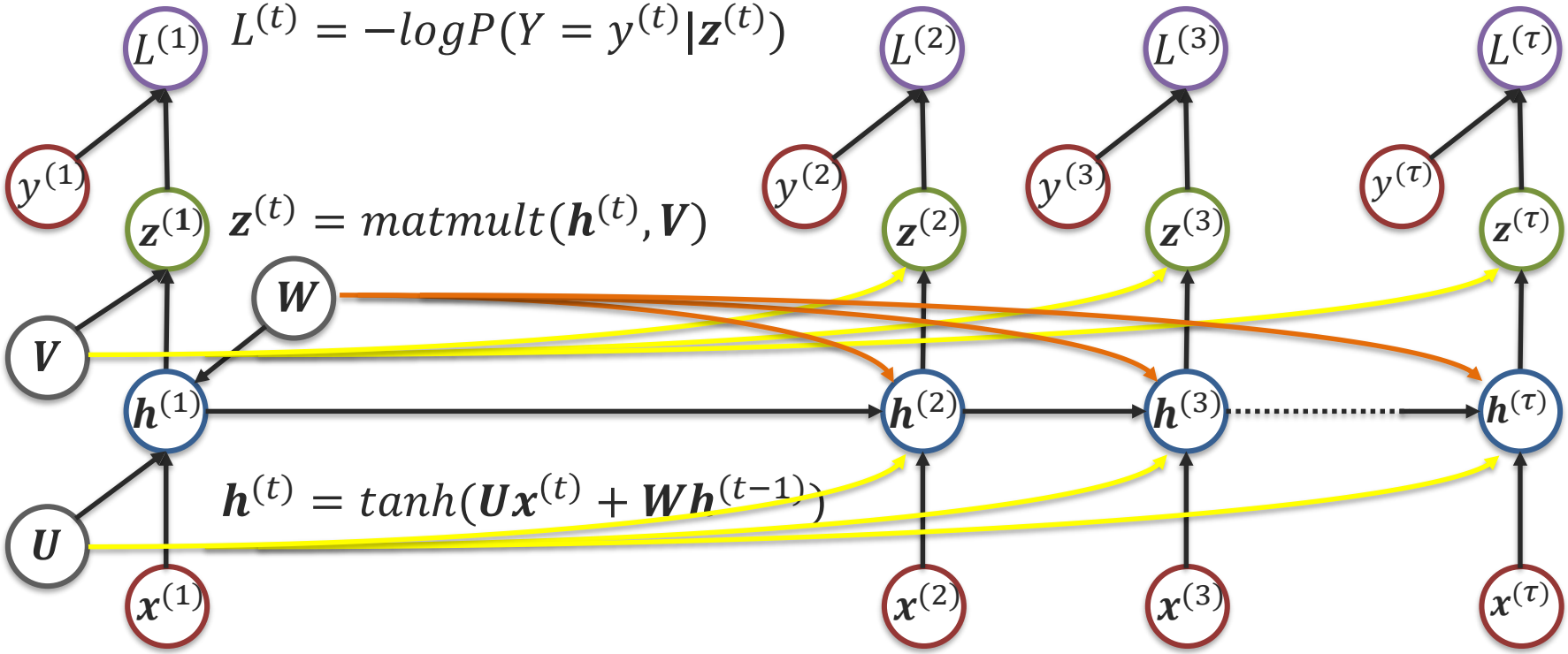
- Initialize output gradient = 1
- Compute "local" Jacobian matrix using values from forward pass
- Use the chain rule:

Gradient = "local" Jacobian x "backprop" gradient



Recurrent Neural Networks

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



Backpropagation Through Time

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

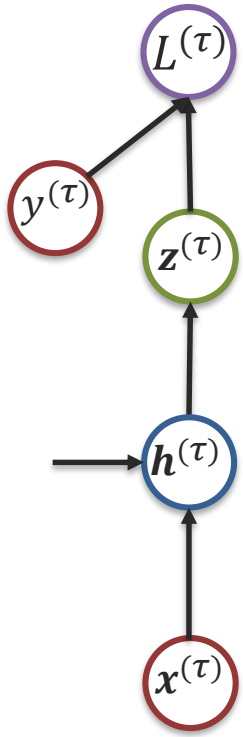
Gradient = "backprop" gradient x "local" Jacobian

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

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

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

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



Backpropagation Through Time

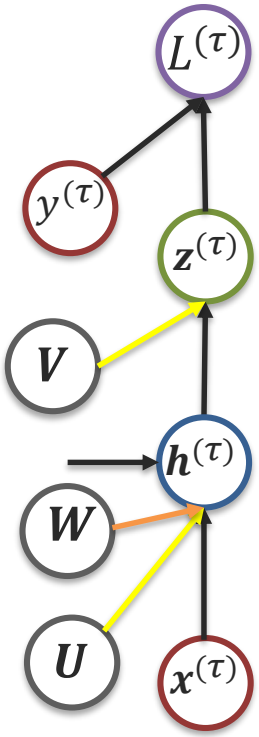
$$L = \sum_t L^{(t)} = -\sum_t logP(Y = y^{(t)}|\boldsymbol{z}^{(t)})$$

Gradient = "backprop" gradient x "local" Jacobian

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

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

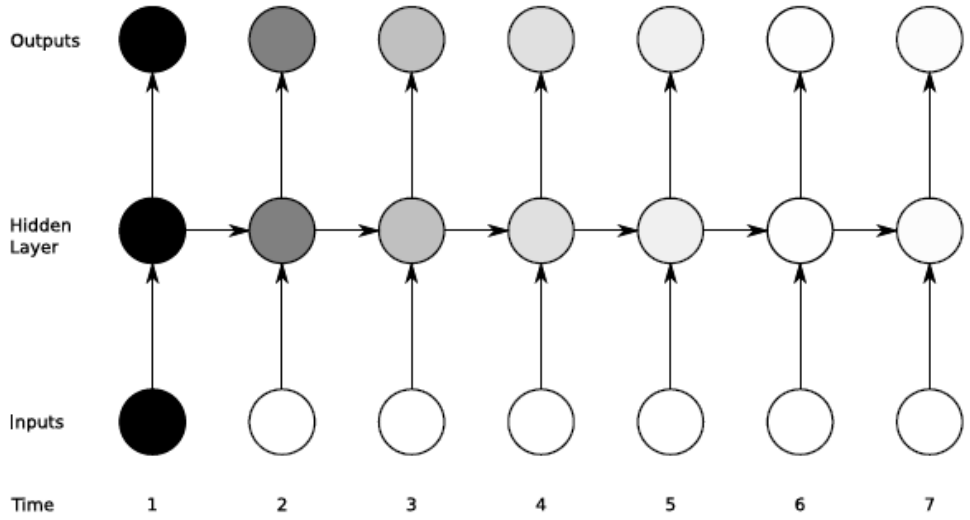
$$\boldsymbol{U} \quad \nabla_{\boldsymbol{U}} L = \sum_t \left(\nabla_{\boldsymbol{h}^{(t)}} L\right) \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{U}}$$



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.