SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Established in collaboration with MIT

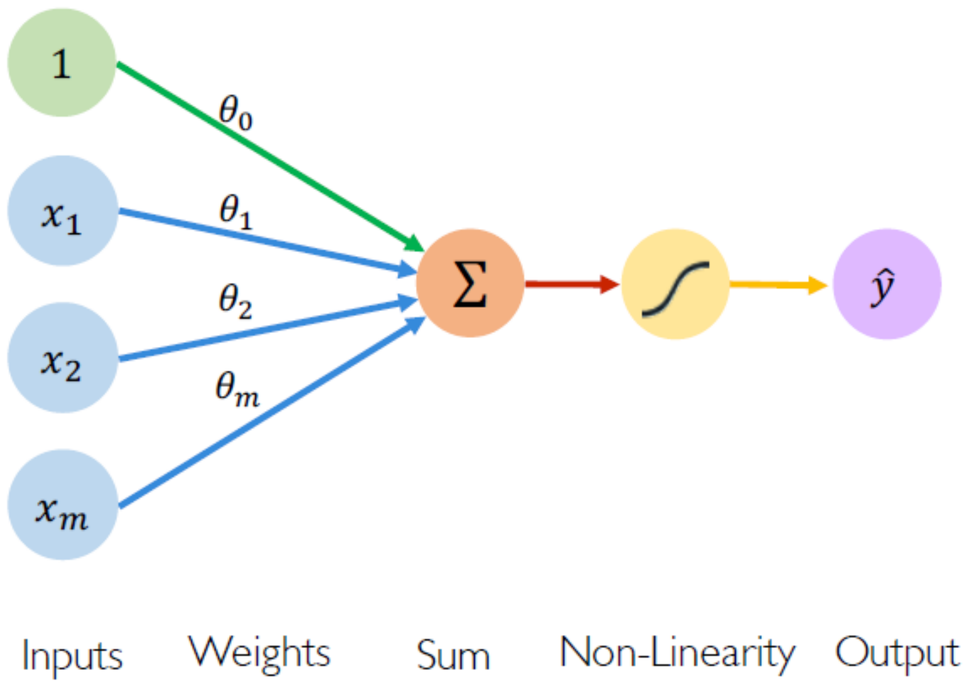
Recurrent Neural Networks (Part I)

SOUJANYA PORIA

Objectives

- Understand the limitations of feed-forward neural networks
- Understand how a Recurrent Neural Network (RNN) works
- Able to list the types of RNNs in terms of input/outputs

Recap on Perceptron



Output

Linear combination of inputs

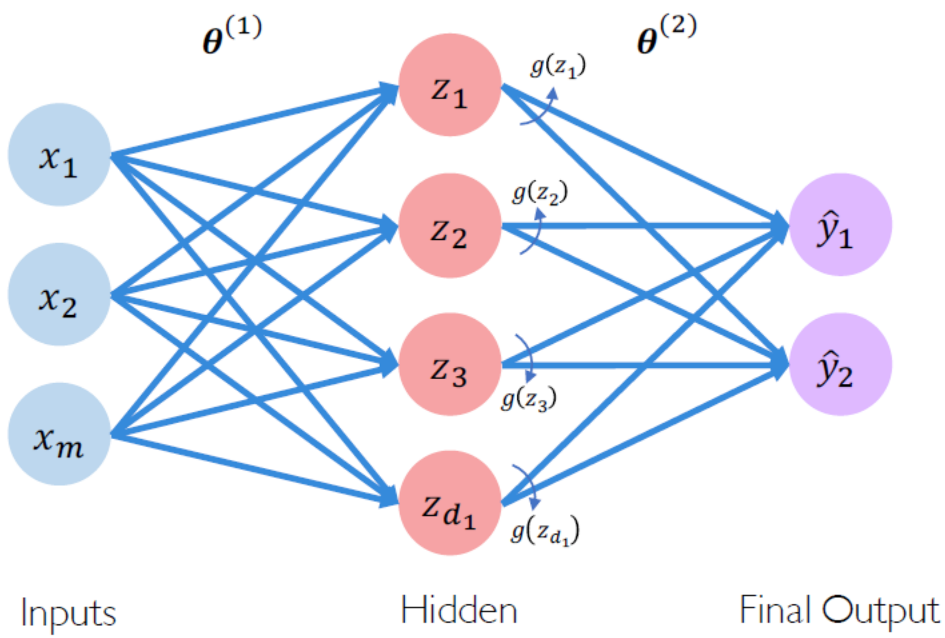
$$\hat{y} = g\left(\theta_0 + \sum_{i=1}^{m} x_i \, \theta_i\right)$$

Non-linear activation function

Bias

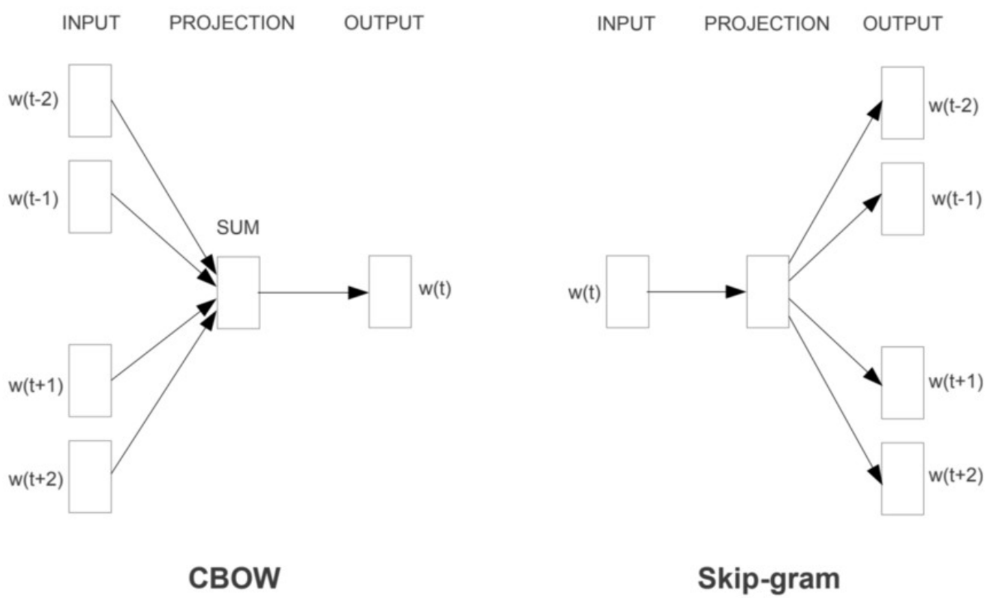
Recap on MLP

- Multiple perceptrons, aka a Multi-layer Perceptron (MLP)



Word2Vec

- Continuous Bag of Words (CBOW) and Skip-gram

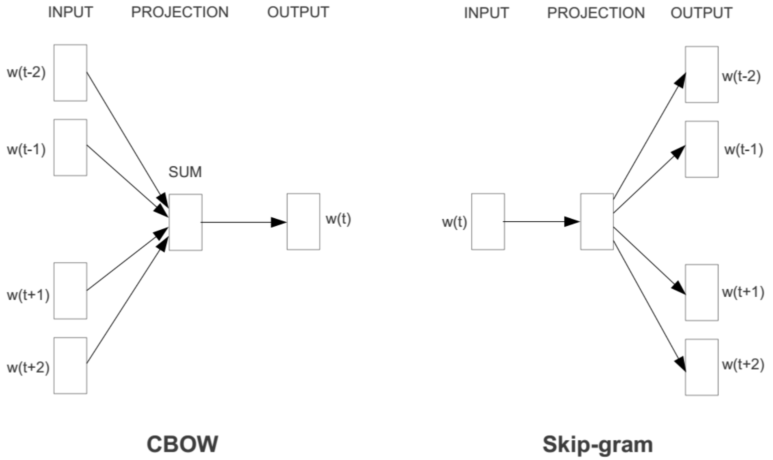


word2vec approach to represent the meaning of word

- Represent each word with a low-dimensional vector
- Word similarity = vector similarity
- Key idea: Predict surrounding words of every word
- Faster and can easily incorporate a new sentence/document or add a word to the vocabulary

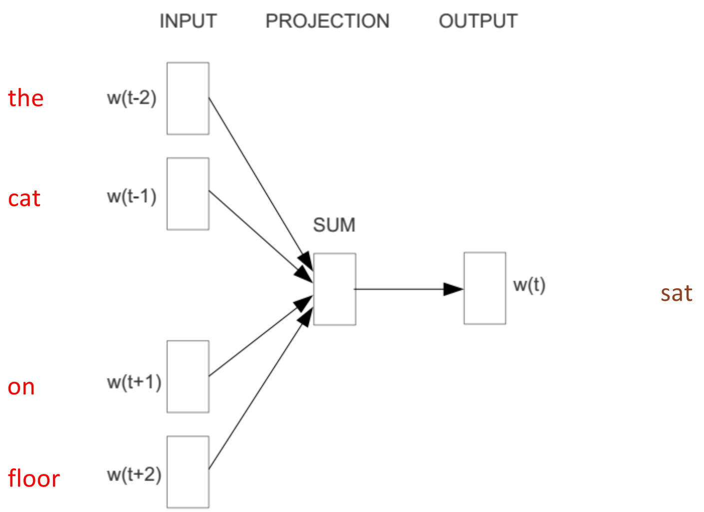
Represent the meaning of word – word2vec

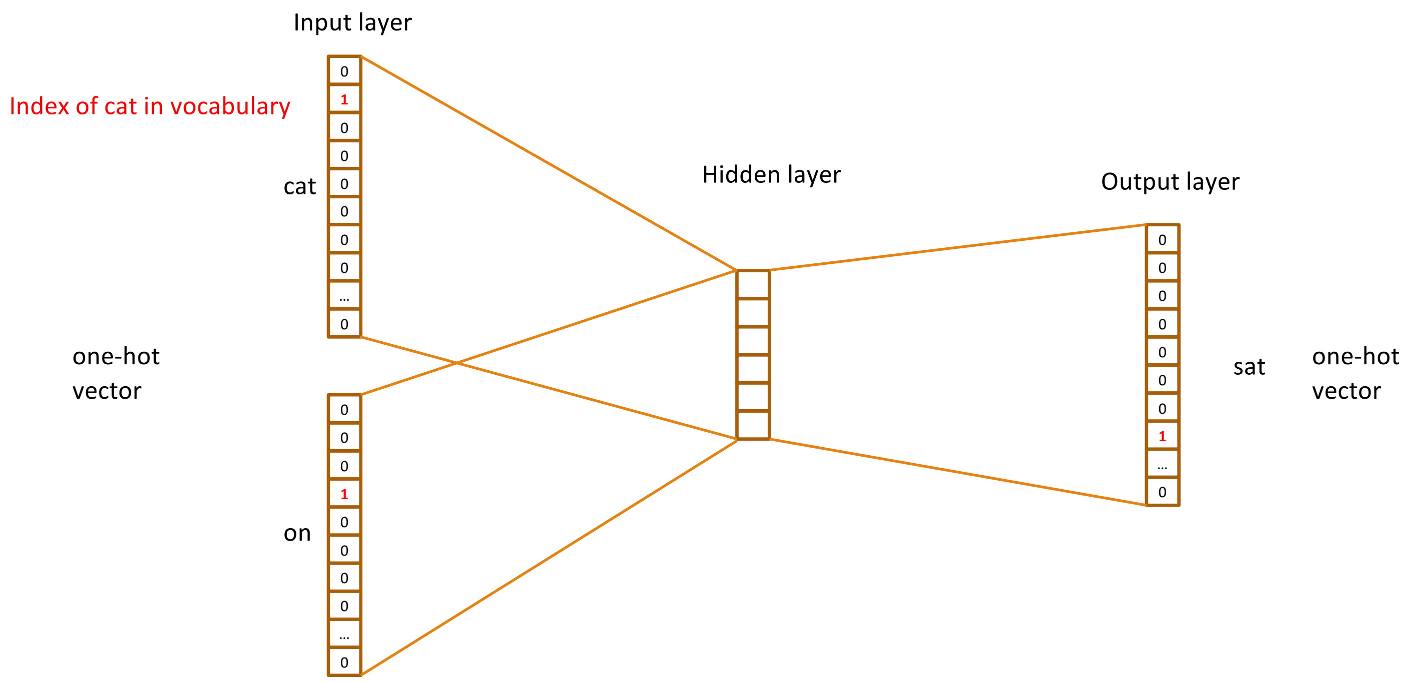
- 2 basic neural network models:
  - Continuous Bag of Word (CBOW): use a window of word to predict the middle word
  - Skip-gram (SG): use a word to predict the surrounding ones in window.



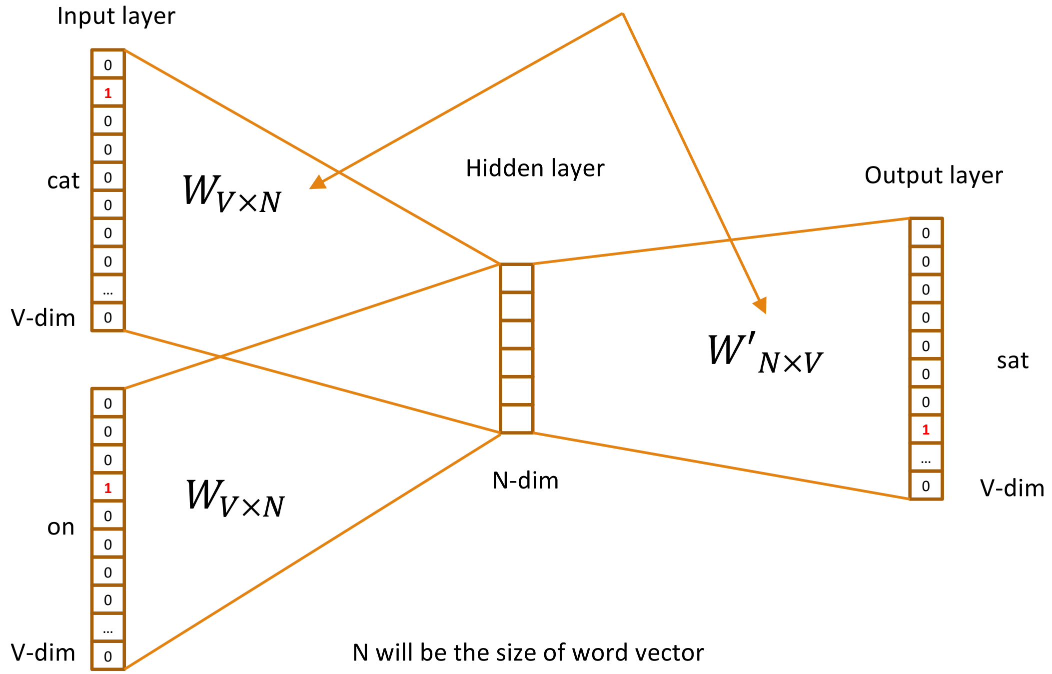
Word2vec – Continuous Bag of Word

- E.g. “The cat sat on floor”
  - Window size = 2

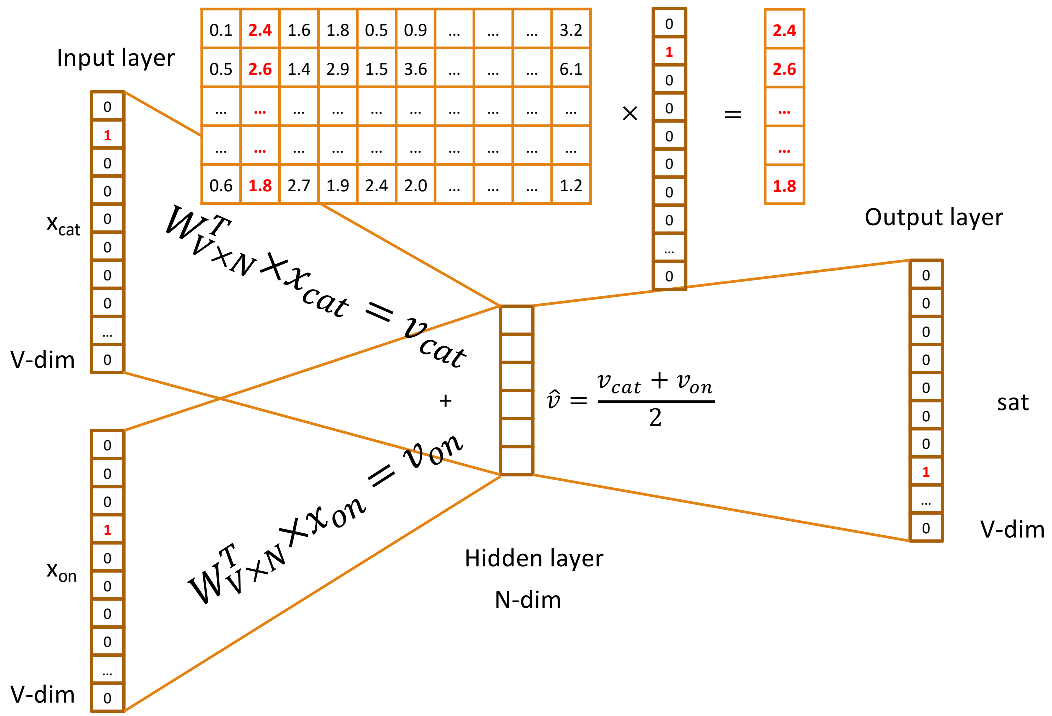



Input layer
Index of cat in vocabulary
cat
0
1
0
0
0
0
0
0
...
0
one-hot
vector
on
0
0
0
1
0
0
0
0
...
0
Hidden layer
Output layer
0
0
0
0
0
0
0
1
...
0
sat
one-hot
vector

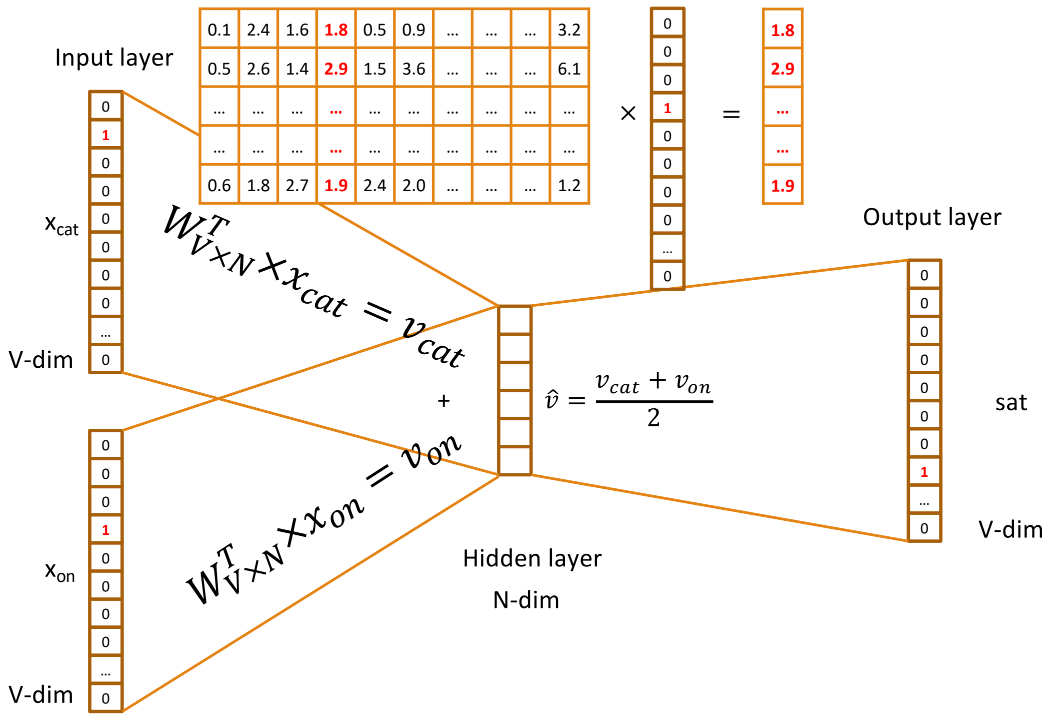
We must learn W and W′

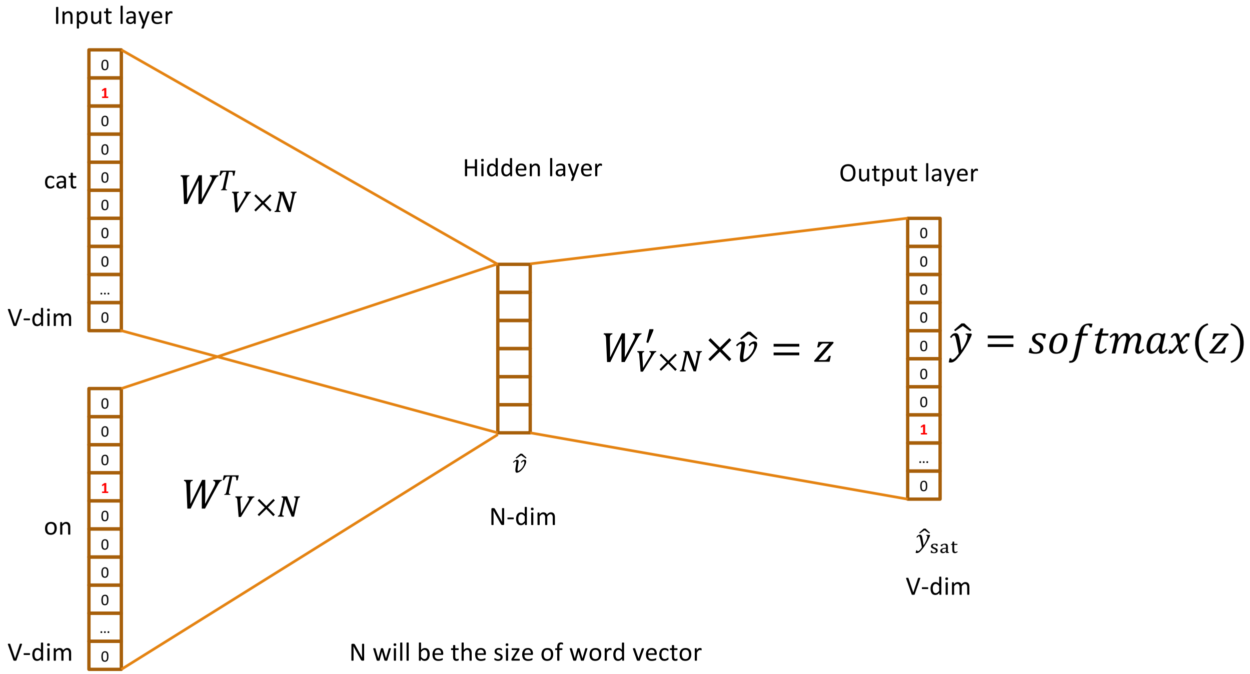


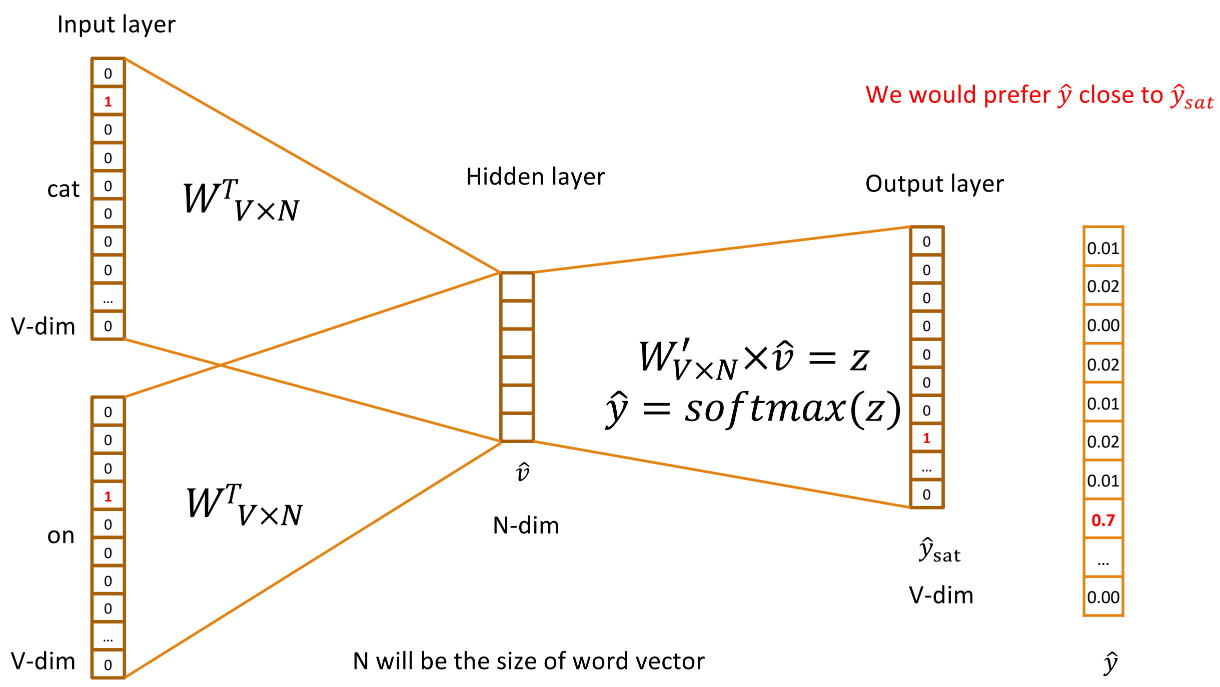
$$W_{V\times N}^{T} \qquad \times x_{cat} = v_{cat}$$



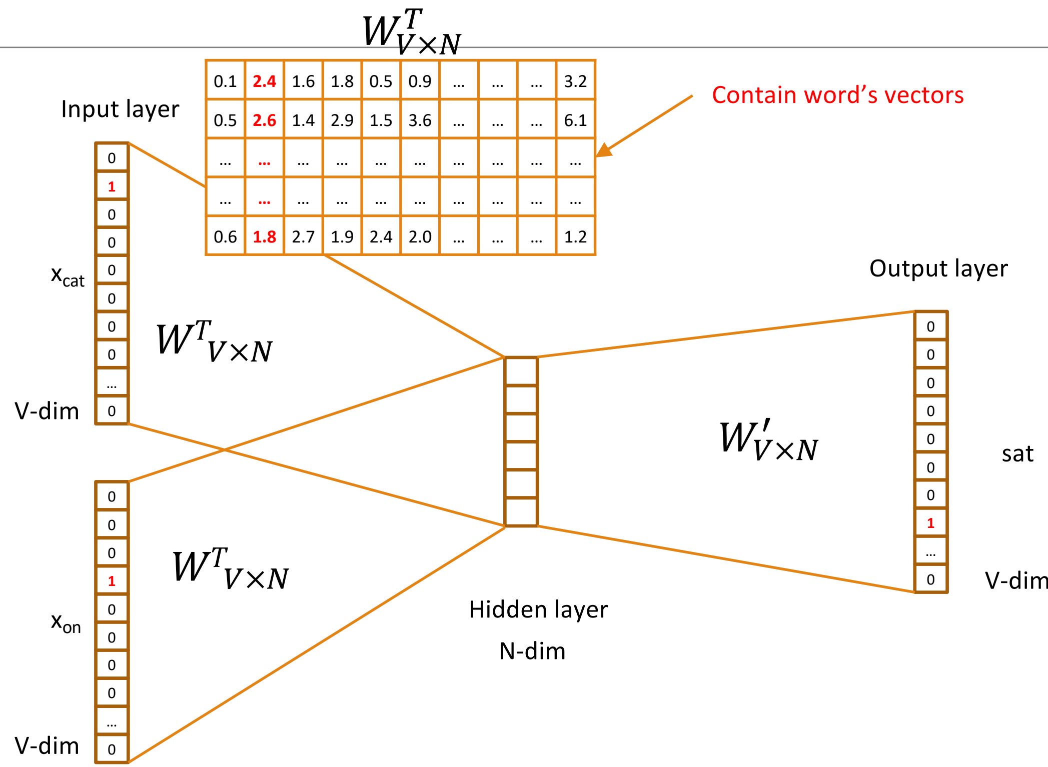
$$W^T_{V\times N} \qquad \times x_{on} = v_{on}$$




Input layer
cat
V-dim
on
V-dim
$W^T_{V\times N}$
$W^T_{V\times N}$
Hidden layer
$\hat{v}$
N-dim
$W'_{V\times N}\times\hat{v} = z$
Output layer
$\hat{y} = softmax(z)$
$\hat{y}_{sat}$
V-dim
N will be the size of word vector



Input layer
cat
V-dim
on
V-dim
$W^T{}_{V\times N}$
$W^T{}_{V\times N}$
Hidden layer
$\hat{v}$
N-dim
Output layer
We would prefer $\hat{y}$ close to $\hat{y}_{sat}$
$W'_{V\times N}\times\hat{v} = z$
$\hat{y} = softmax(z)$
$\hat{y}_{\text{sat}}$
V-dim
0.01
0.02
0.00
0.02
0.01
0.02
0.01
0.7
...
0.00
$\hat{y}$
N will be the size of word vector



We can consider either W or W' as the word's representation. Or even take the average.

Some interesting results

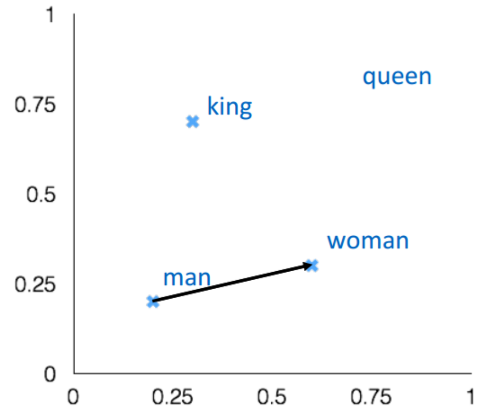
Word Analogies

Test for linear relationships, examined by Mikolov et al. (2014)

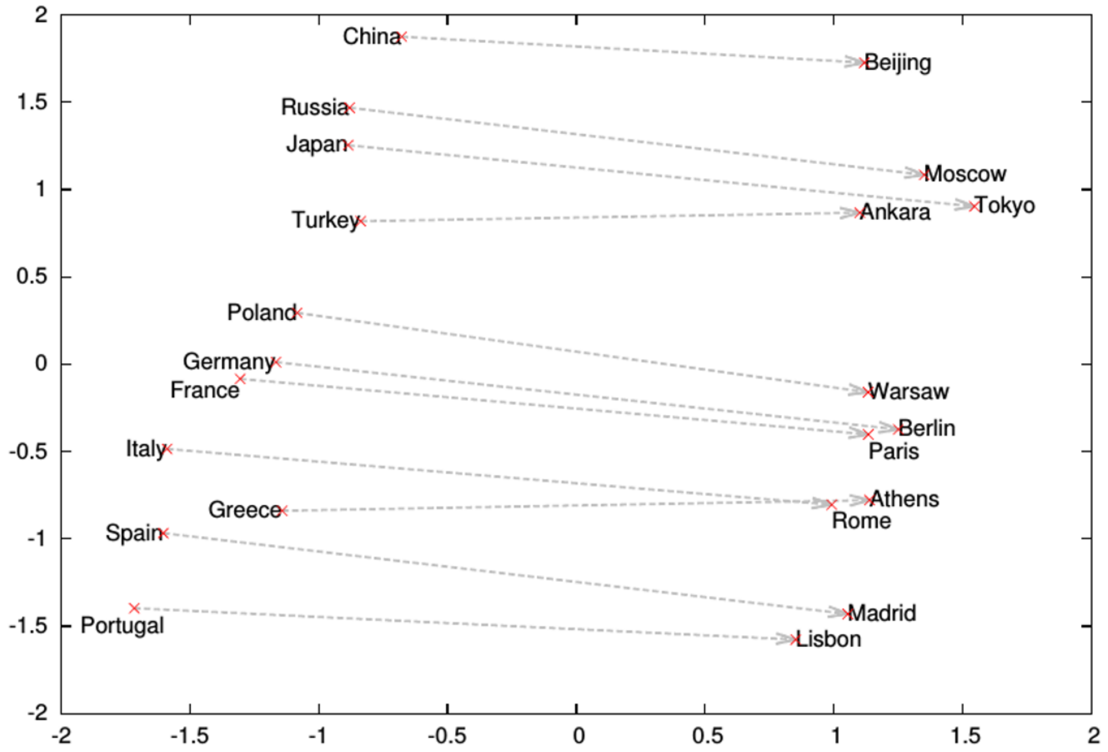
a:b :: c:? $\longrightarrow$ $$d = \arg\max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||}$$

man:woman :: king:?

+	king	[0.30 0.70]
-	man	[0.20 0.20]
+	woman	[0.60 0.30]
	queen	[0.70 0.80]

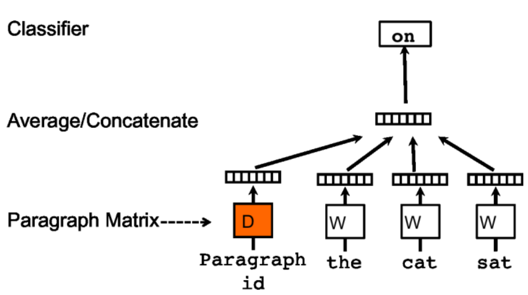


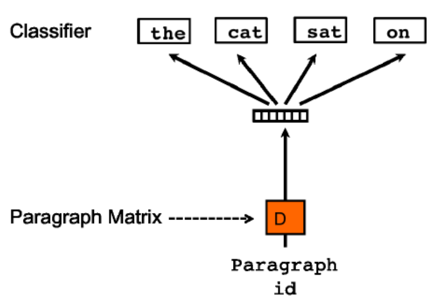
Word analogies



Represent the meaning of sentence/text

- Simple approach: take avg of the word2vecs of its words
- Another approach: Paragraph vector (2014, Quoc Le, Mikolov)
  - Extend word2vec to text level
  - Also two models: add paragraph vector as the input

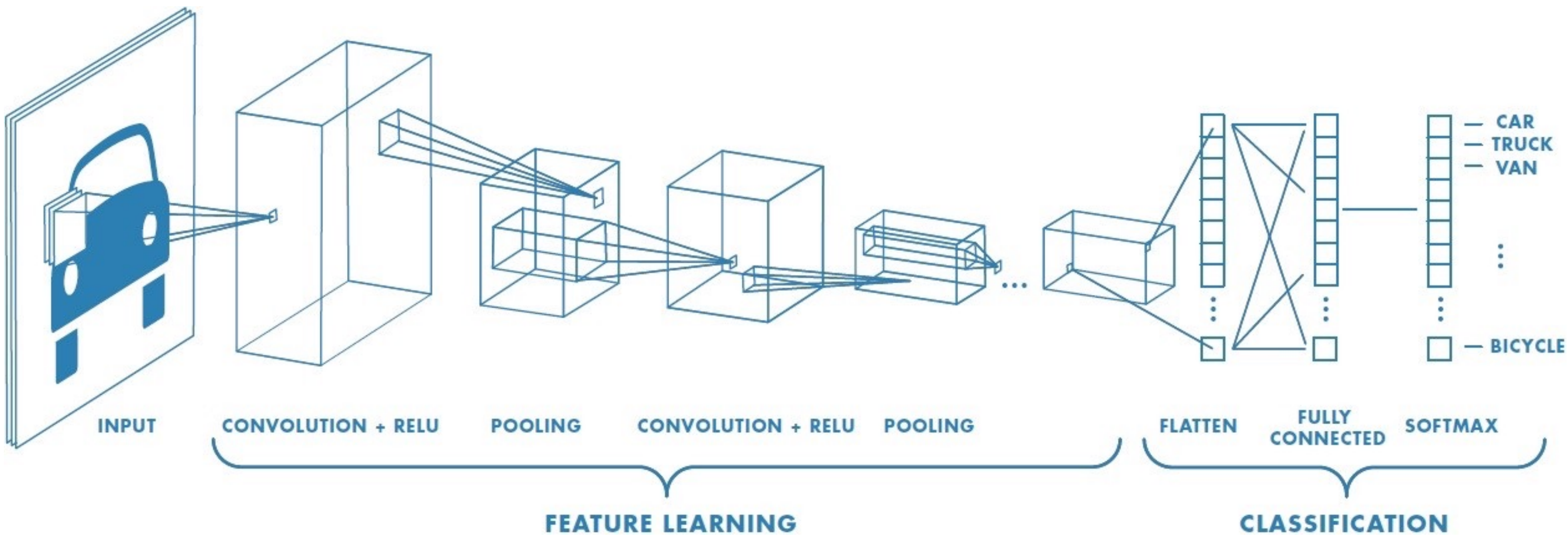




Applications

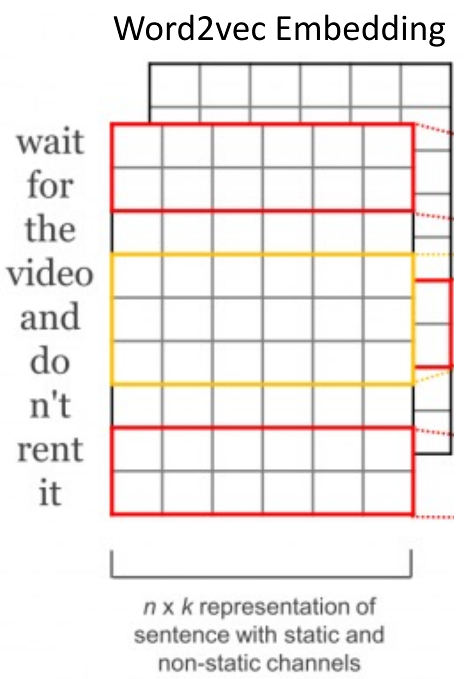
- Search, e.g., query expansion
- Sentiment analysis
- Classification
- Clustering

Recap on CNN


CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

CNN for Text

- “Shallow” CNN on sentences represented by word embeddings
  - Sentence represented as *n x k* matrix (*n* words and embedding dimension *k*)

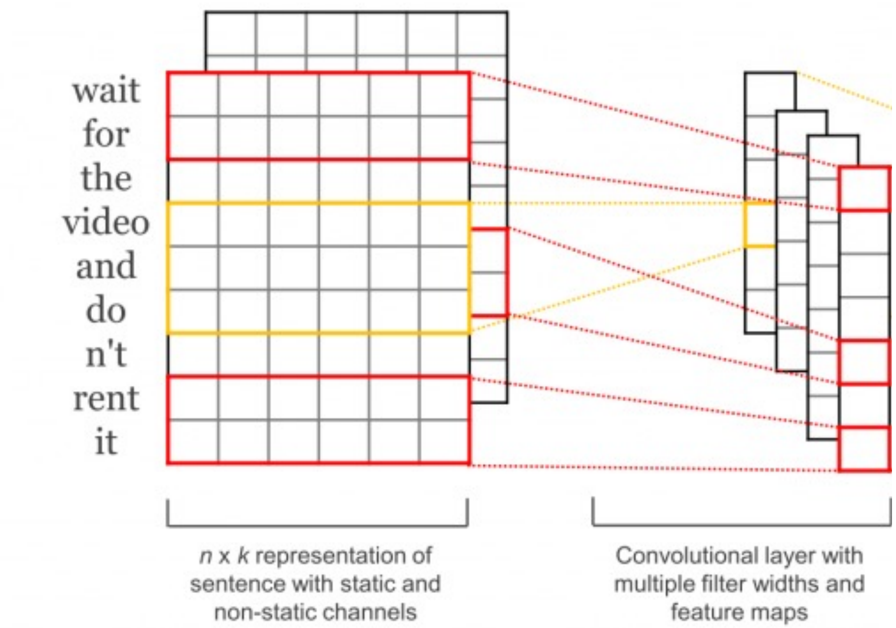


Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*(pp. 1746-1751).

CNN for Text

- "Shallow" CNN on sentences represented by word embeddings
  - Apply convolutional filters that cover 2,3,n words at a time (with width *k*)

Word2vec Embedding

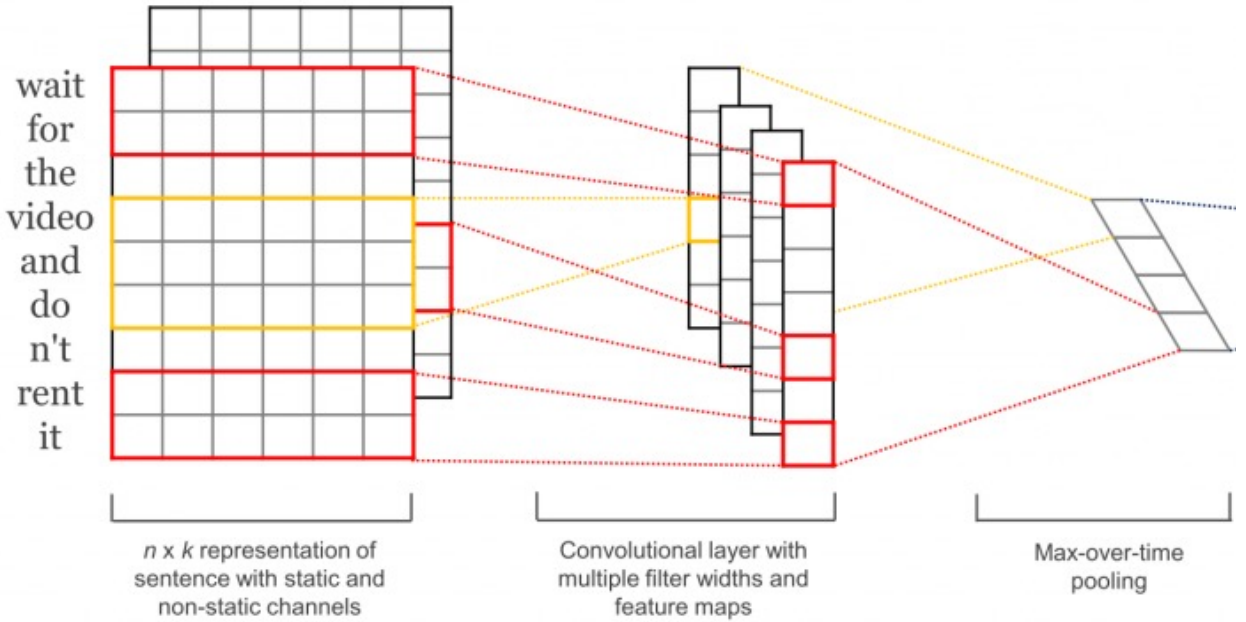


Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*(pp. 1746-1751).

CNN for Text

- "Shallow" CNN on sentences represented by word embeddings
  - Apply max pooling to select highest value, and flatten layer

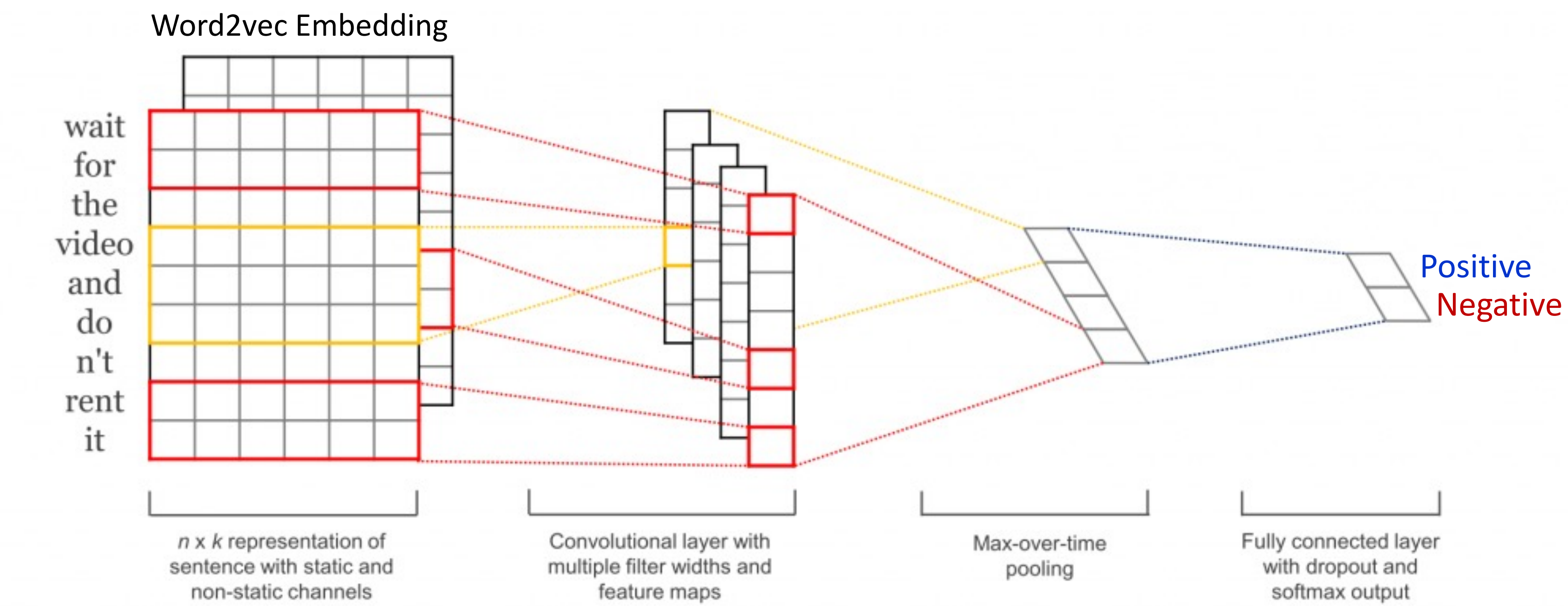
Word2vec Embedding



Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*(pp. 1746-1751).

CNN for Text

- "Shallow" CNN on sentences represented by word embeddings
  - Final softmax layer to determine most likely class



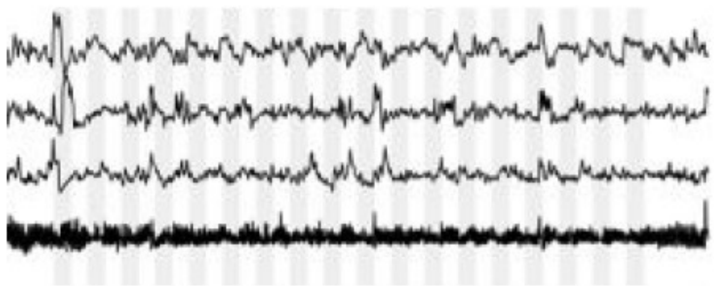
Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*(pp. 1746-1751).

Exercise 1: Standard Neural Networks

- So far, we have covered topics on the single perceptron, MLP, and CNN.
- What is the common characteristic and limitation of these neural networks?

Temporal Sequences

"This morning I took the dog for a walk." *Sentences*

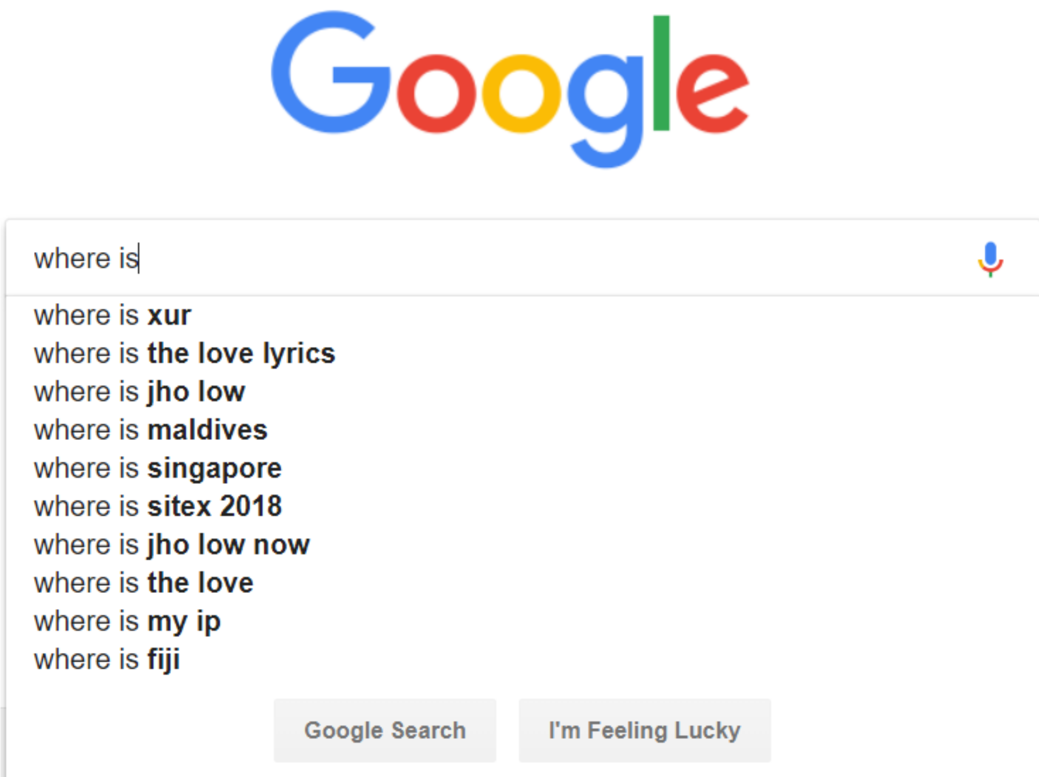


Heart-rates

Audio Recordings

Sequence Modelling Problem

- E.g., Sentence completion



Sequence Modelling Problem

- E.g., Sentence completion

"This morning I took the dog for a walk."

given these words

predict what comes next?

Exercise 2: Approaches to Sequence Modelling Problem

- Using a Sentence Completion example, what are the possible problems?
  - Solution 1: Using a fixed window of preceding words

"This morning I took the dog for a walk."

given these 2 words, predict the next word

  - Solution 2: Model entire sentence as Bag-of-words

This morning I took the dog for a

↓

[0 1 0 0 1 0 0 ... 0 0 1 1 0 0 0 1]

  - Solution 3: Like solution 1 but with a very large window

"This morning I took the dog for a walk."

given these 7 words, predict the next word

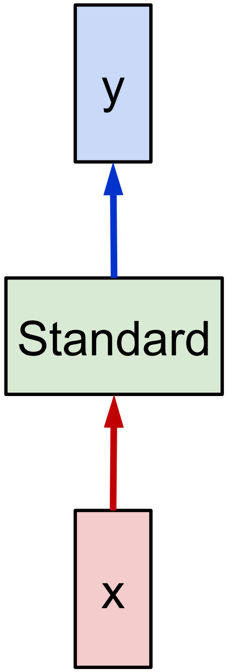
[1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 ...]
morning I took the dog ...

Motivations behind RNN

- Inputs and output may not be of a fixed length
  - E.g., An input paragraph of variable word count
- Want to model temporal aspect (sequence order) as context
  - E.g., language translation or stock prediction
- Hard to determine appropriate window size for context
  - E.g., past 3 days VS 6 months, previous word VS last 8^{th} word
- Want to share parameters across sequence
  - E.g, patterns that appear in different parts of the temporal sequence

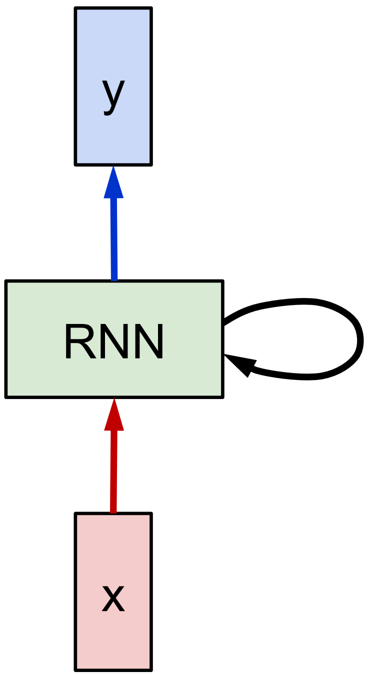
Recurrent Neural Networks

- Standard Neural Networks generate a single output



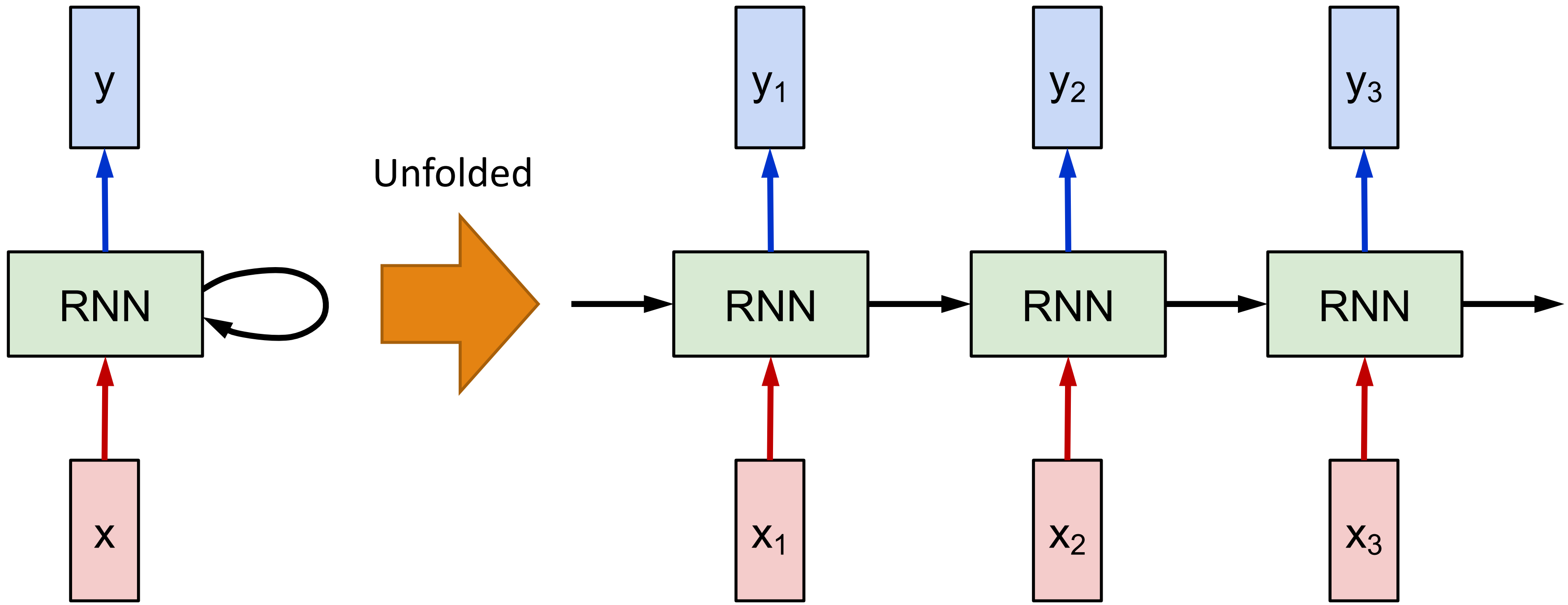
Recurrent Neural Networks

- Recurrent Neural Networks aim to process a sequence of inputs to generate a sequence of outputs at different time steps



Recurrent Neural Networks

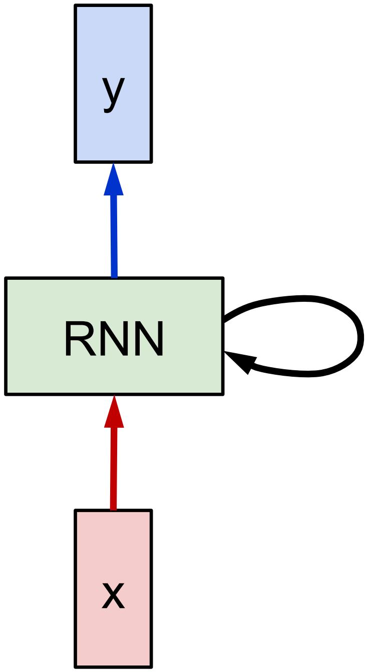
- Recurrent Neural Networks aim to process a sequence of inputs to generate a sequence of outputs at different time steps



Recurrent Neural Networks

- RNNs process a sequence of inputs *X* using a recurrence formula at each time-step *t*

$$h_t = f_W(h_{t-1}, x_t)$$



Recurrent Neural Networks

- RNNs process a sequence of inputs *X* using a recurrence formula at each time-step *t*
  - Each time-step *t* depends on its previous time-step *t-1*

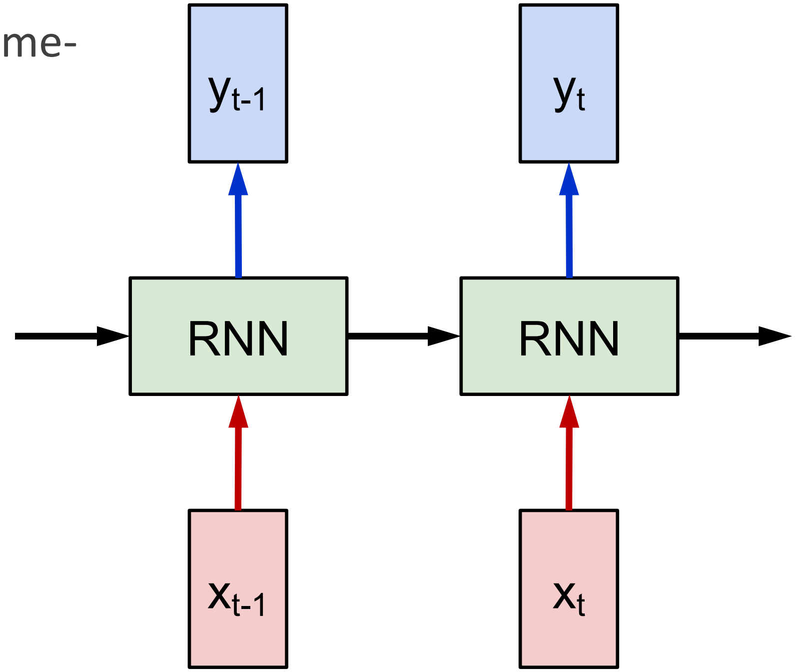
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

some function
with parameters W

old state

input vector at
some time step



Recurrent Neural Networks

- RNNs process a sequence of inputs X using a recurrence formula at each time-step t
  - The same function f_W and parameters W are shared across time-steps

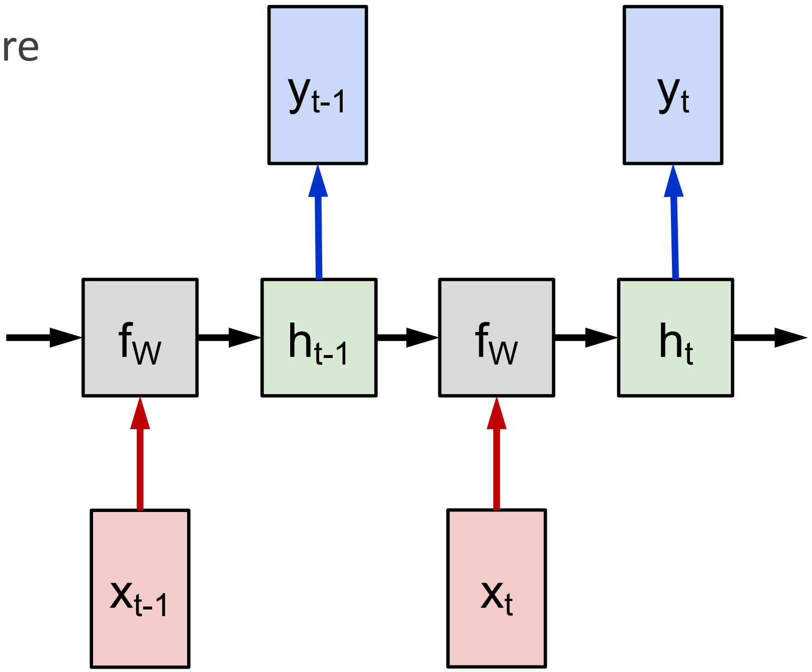
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

some function
with parameters W

old state

input vector at
some time step



Recurrent Neural Networks

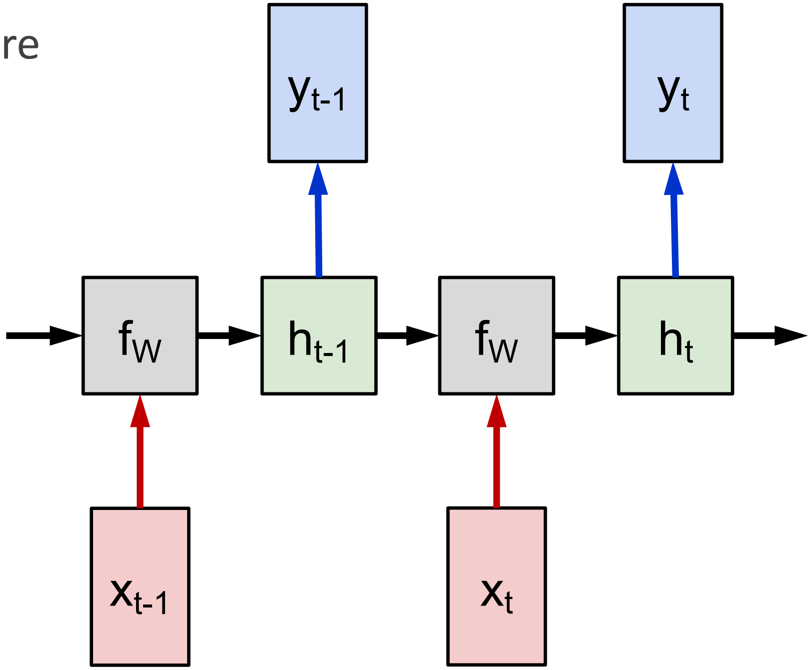
- RNNs process a sequence of inputs X using a recurrence formula at each time-step t
  - The same function f_W and parameters W are shared across time-steps

$$h_t = f_W(h_{t-1}, x_t)$$

$$\downarrow$$

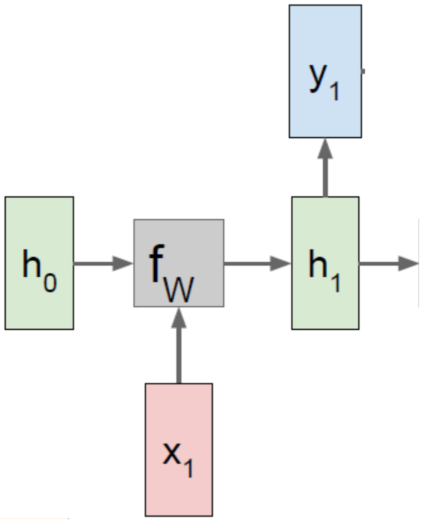
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$



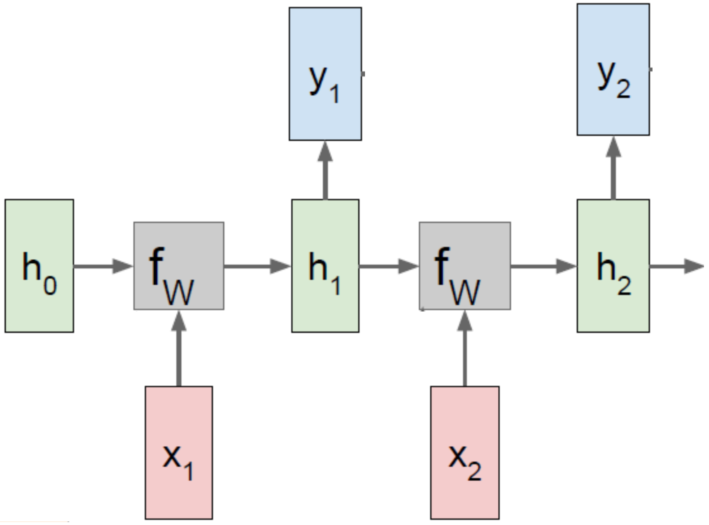
RNN Example

- Start from time-step 1
  - Output y_1,



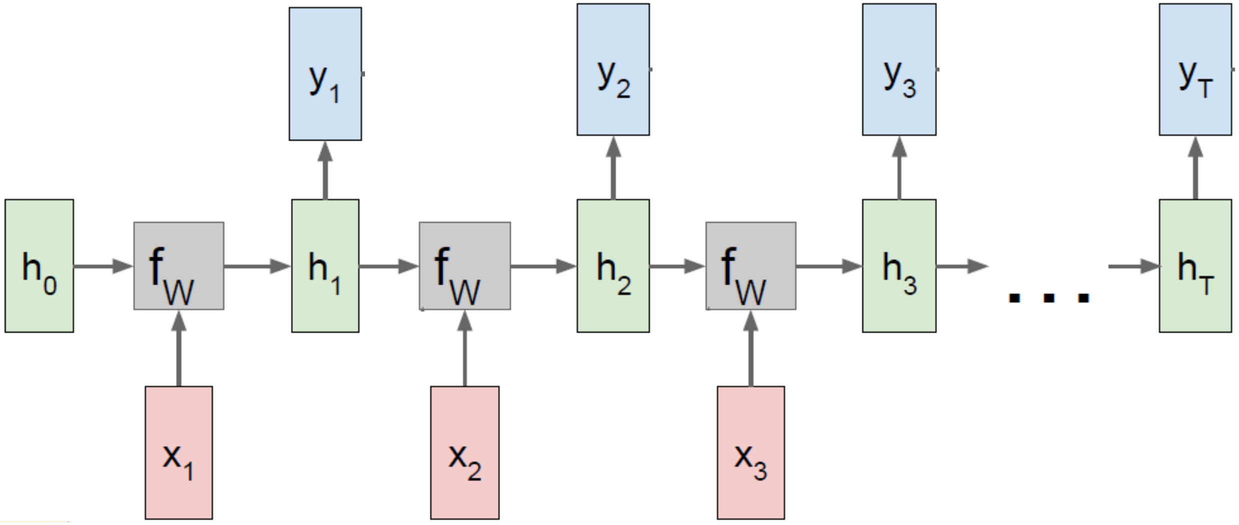
RNN Example

- Start from time-step 1, then time-step 2,
  - Output y_1, y_2 generated



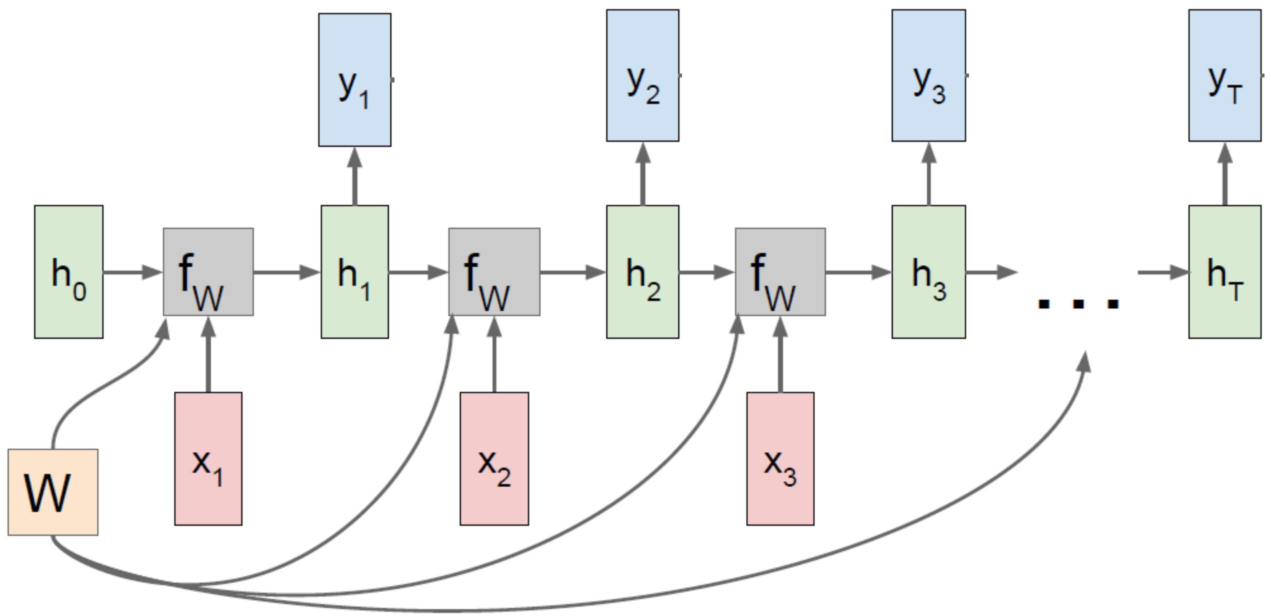
RNN Example

- Start from time-step 1, then time-step 2, until time-step *T*
  - Output y_1, y_2, …, y_T generated



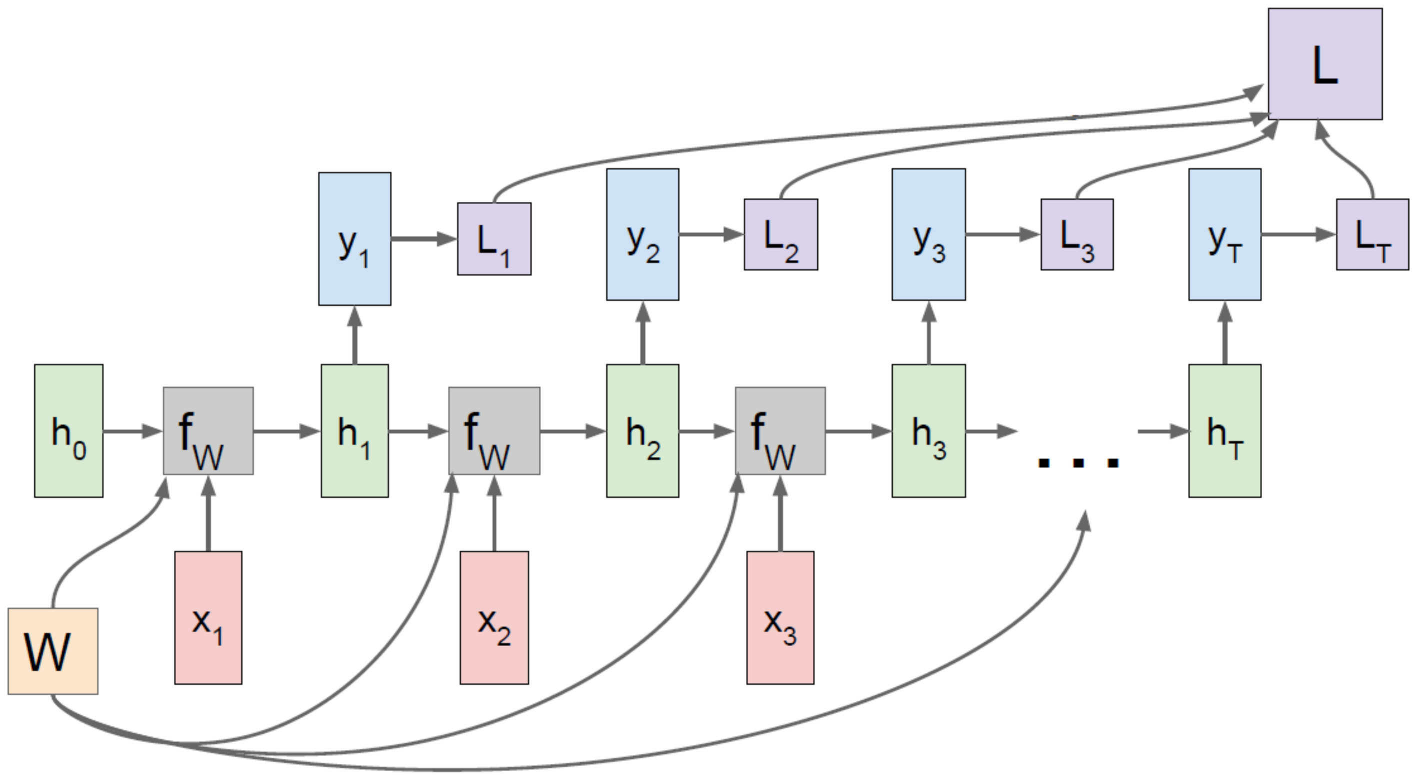
RNN Example

- Same weights *W* used throughout all time-steps



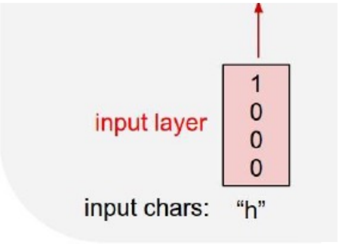
RNN Example

- Same weights *W* used throughout all time-steps
  - Learn weights by minimizing overall loss *L* from all time-steps



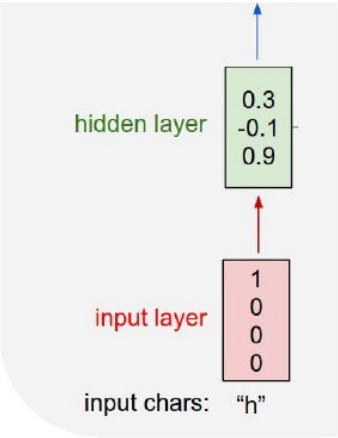
RNN Example: Character-level

- Problem: Trying to predict a sequence of characters
  - Assuming a vocabulary of [h,e,l,o]



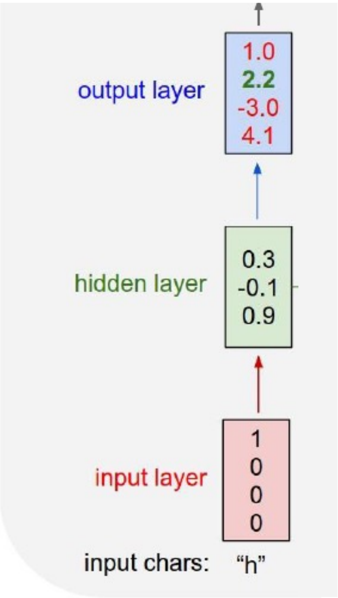
RNN Example: Character-level

- Problem: Trying to predict a sequence of characters
  - Assuming a vocabulary of [h,e,l,o]



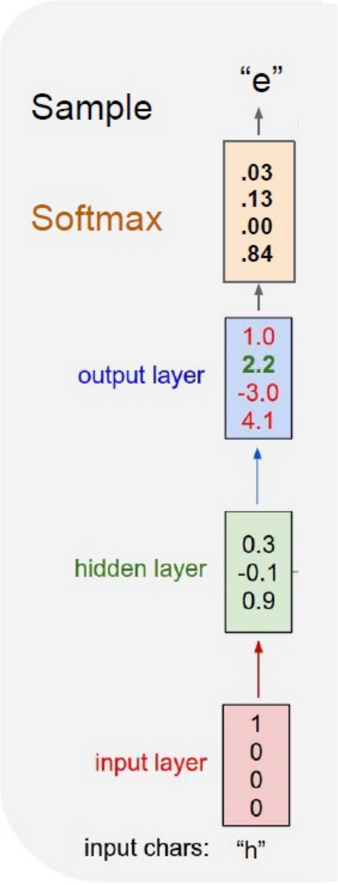
RNN Example: Character-level

- Problem: Trying to predict a sequence of characters
  - Assuming a vocabulary of [h,e,l,o]



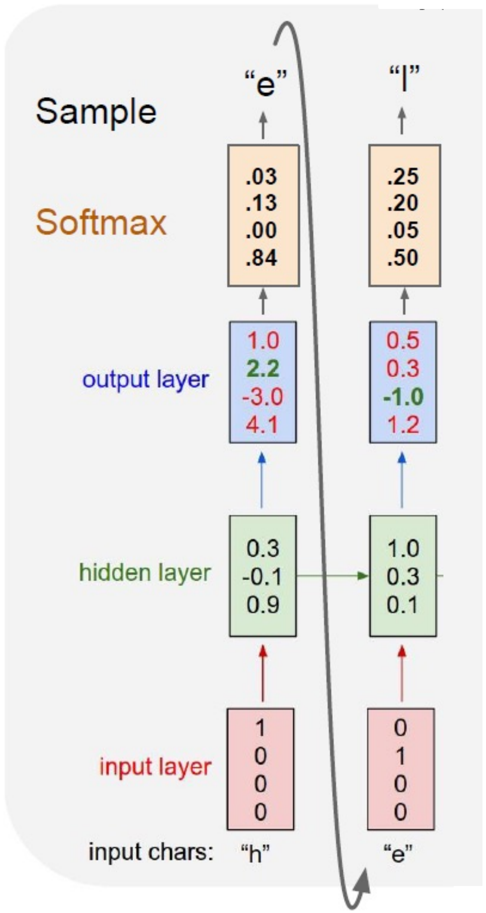
RNN Example: Character-level

- Problem: Trying to predict a sequence of characters
  - Assuming a vocabulary of [h,e,l,o]



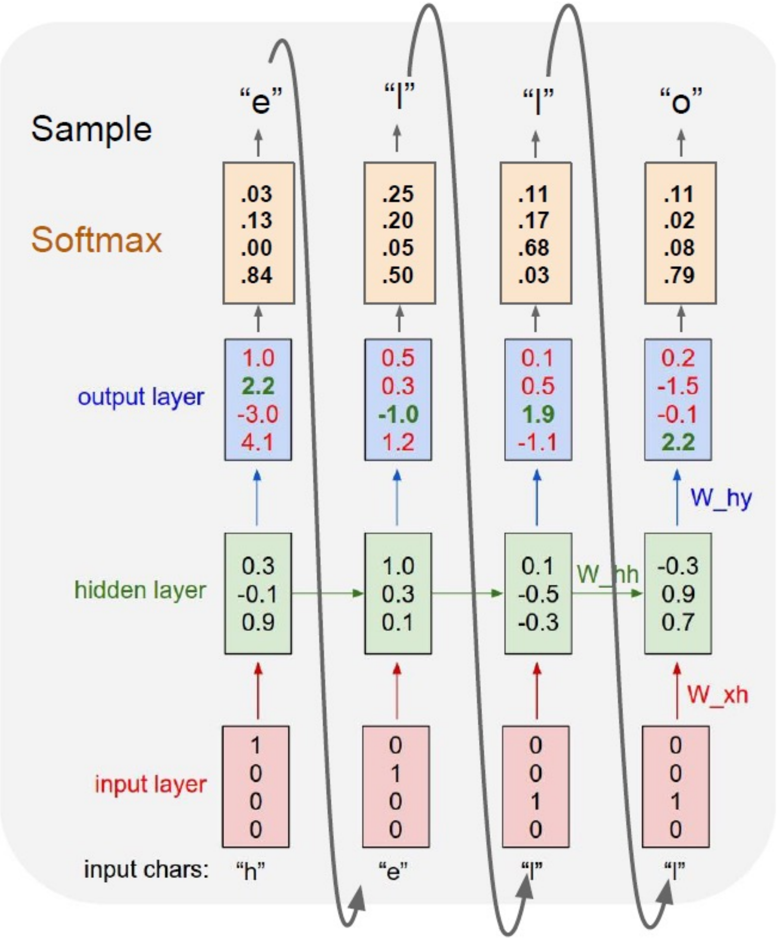
RNN Example: Character-level

- Problem: Trying to predict a sequence of characters
  - Assuming a vocabulary of [h,e,l,o]



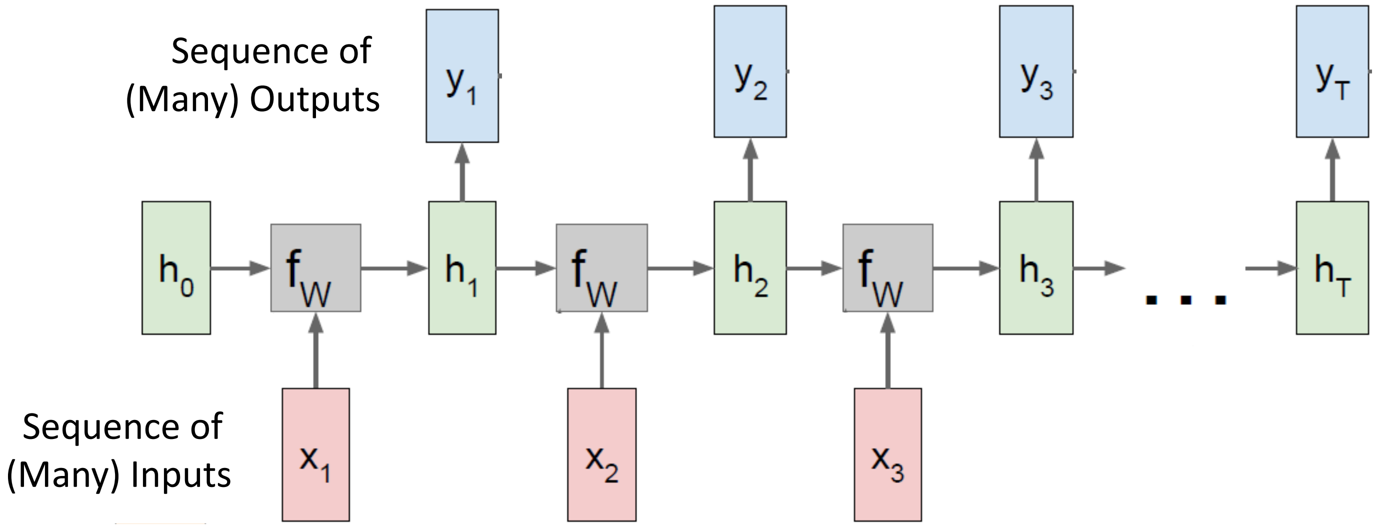
RNN Example: Character-level

- Problem: Trying to predict a sequence of characters
  - Assuming a vocabulary of [h,e,l,o]



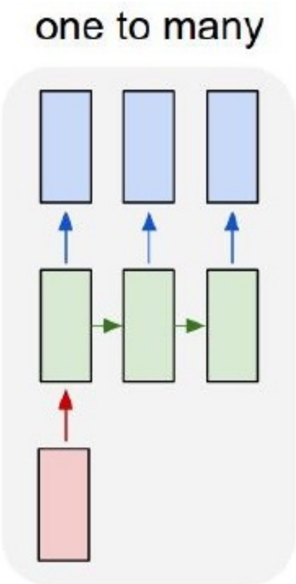
Exercise 3: Types of RNN

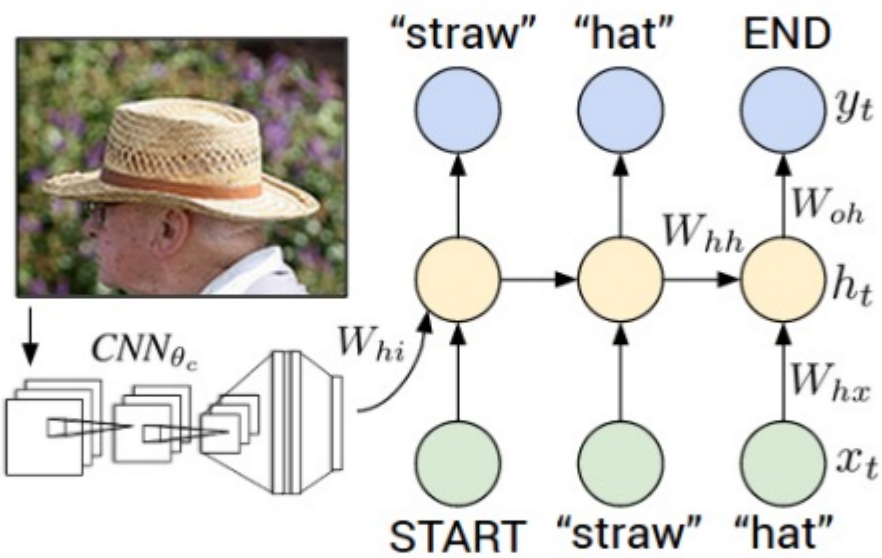
- Based on the input/output sequence, we have previously examined a many-to-many type of RNN. What other types can you think of?



RNN Applications

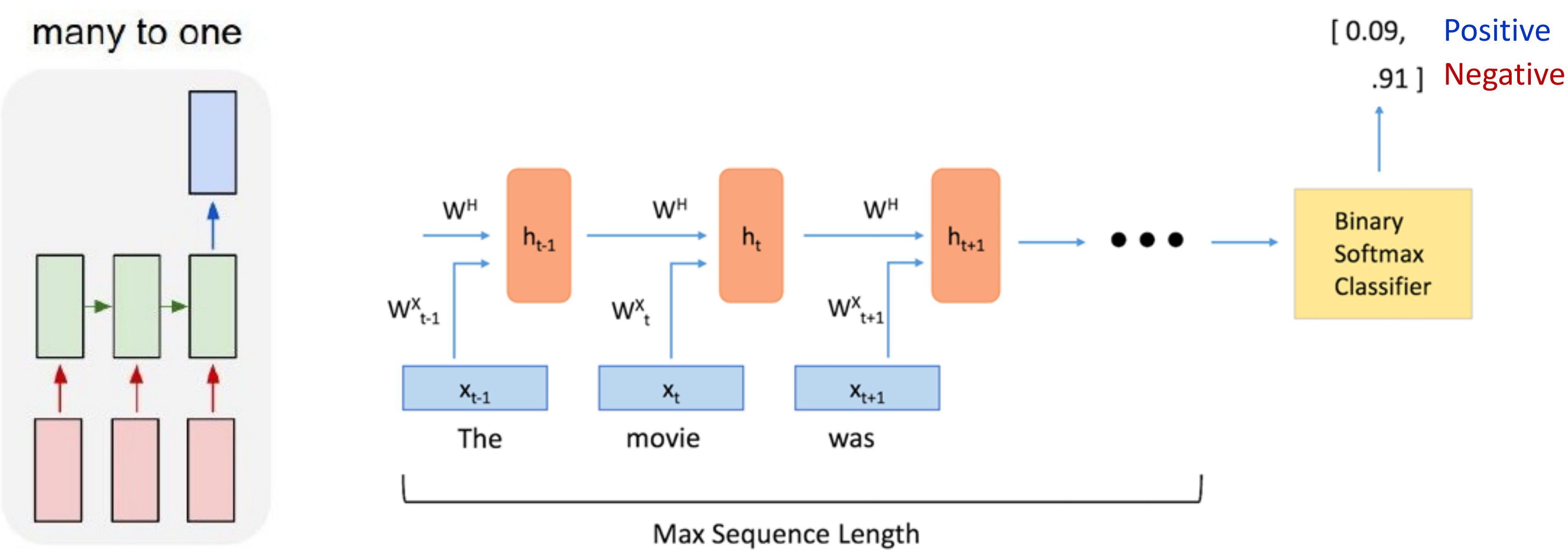
- One-to-many: Image Captioning





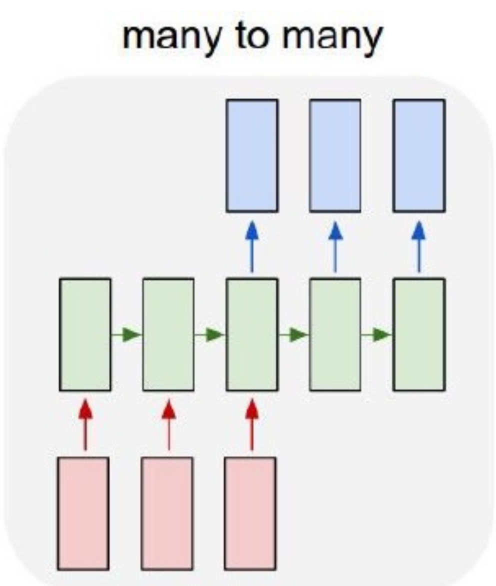
RNN Applications

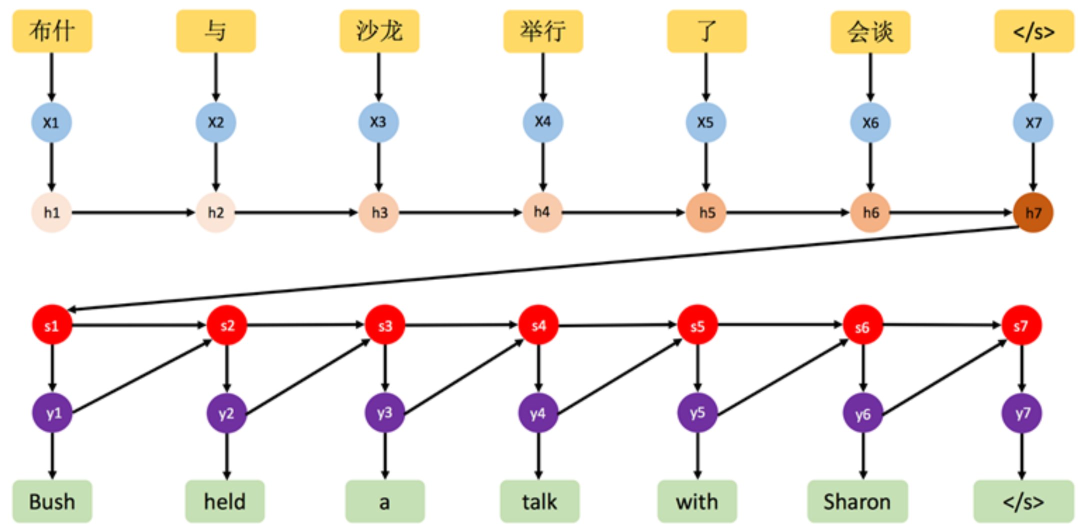
- Many-to-one: Sentiment Classification



RNN Applications

- Many-to-many: Language Translation

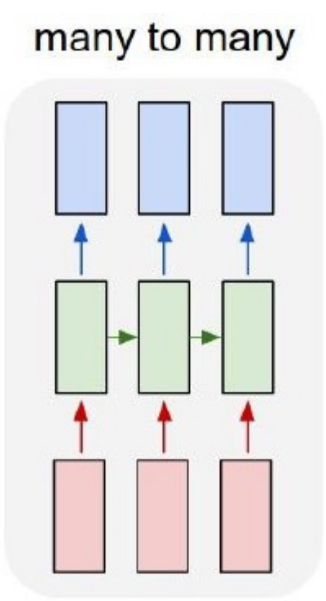


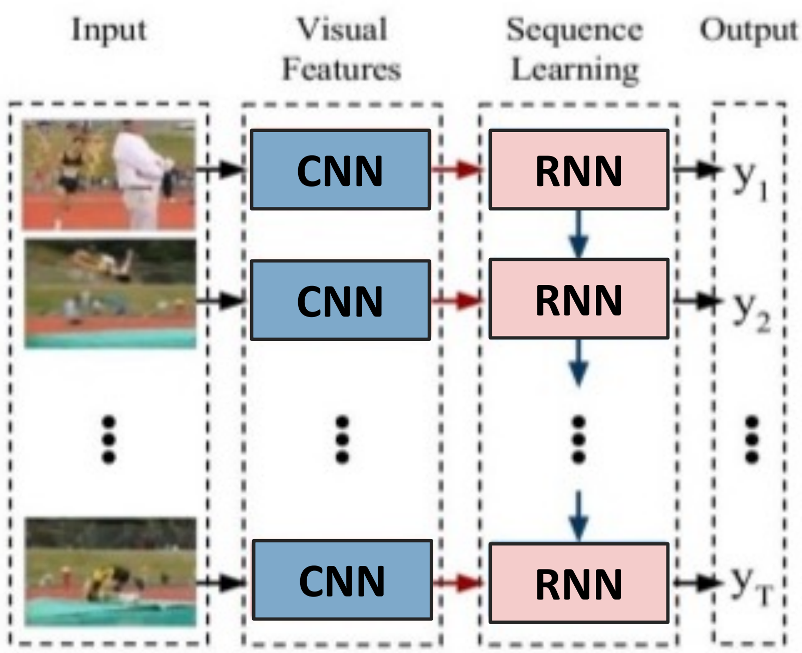


(Sutskever et al., 2014)

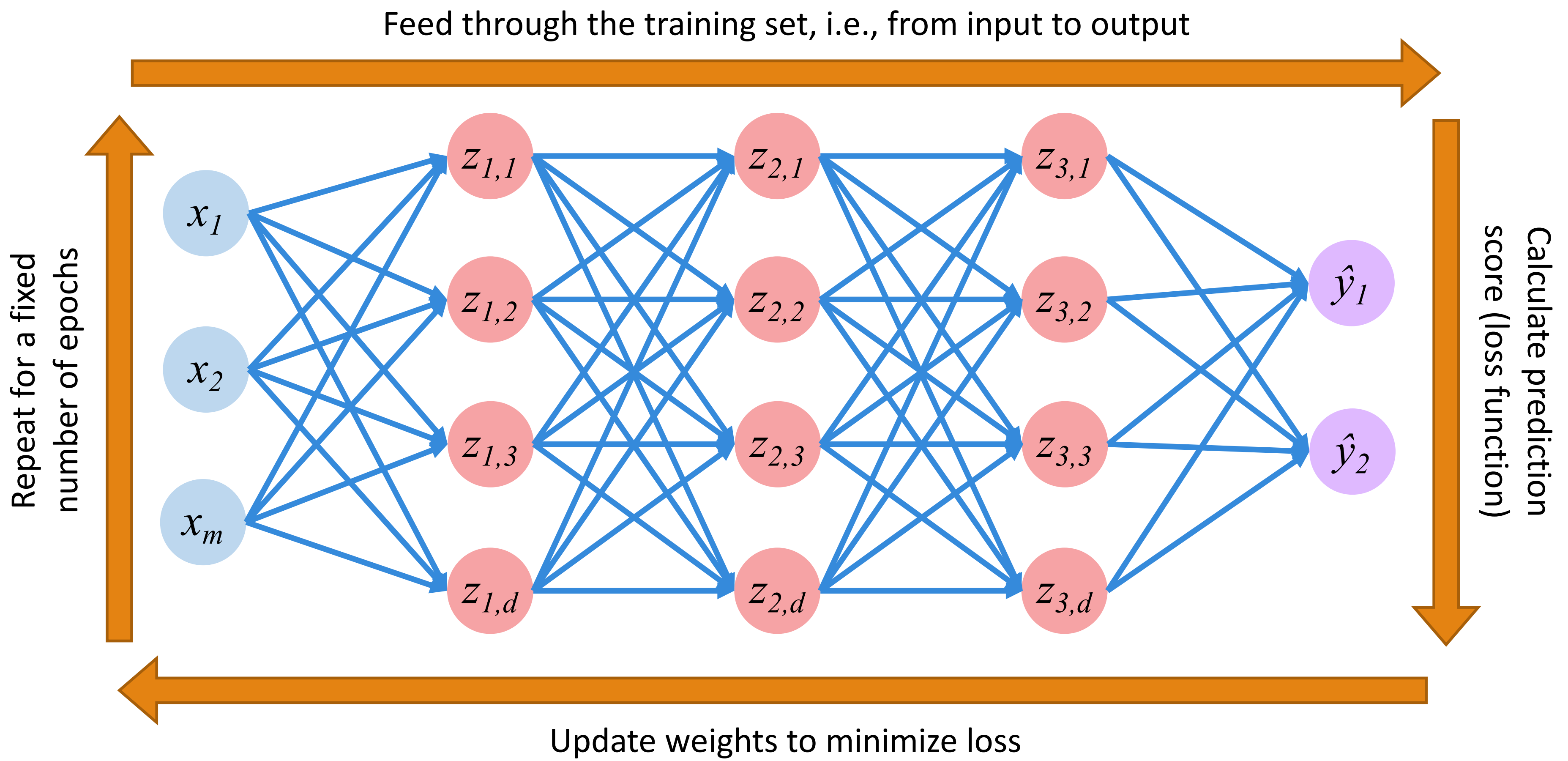
RNN Applications

- Many-to-many: Video Classification



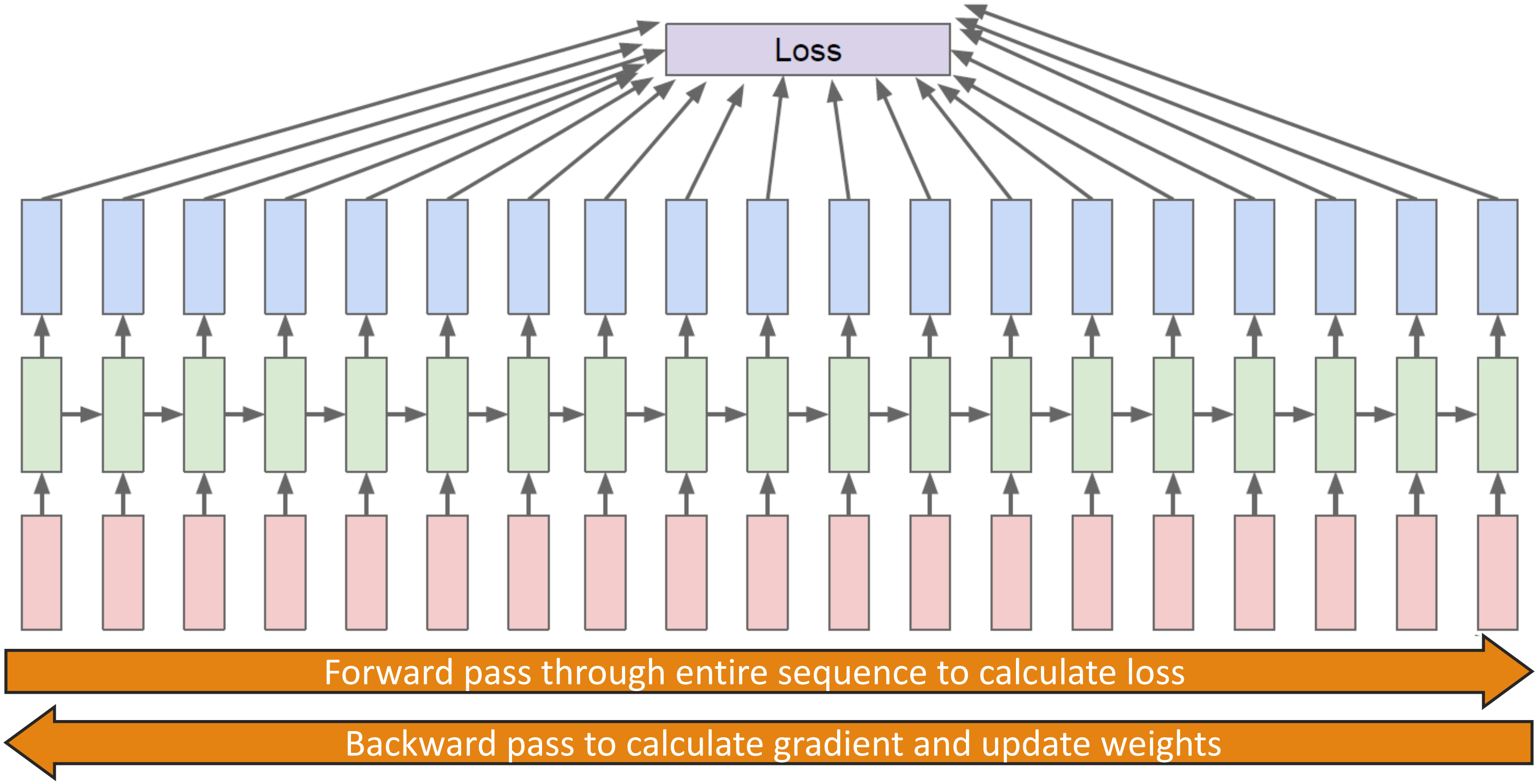


Recap on Training MLP



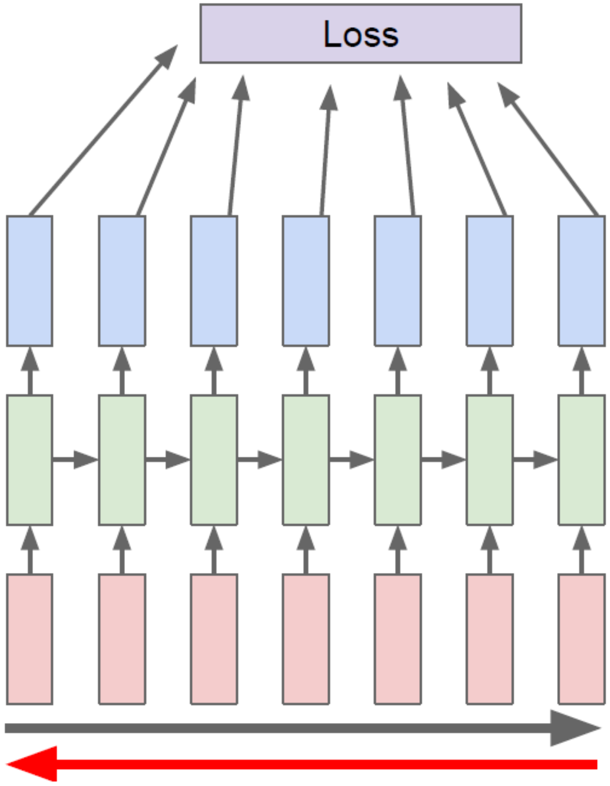
RNN Training

- Backpropagation through time



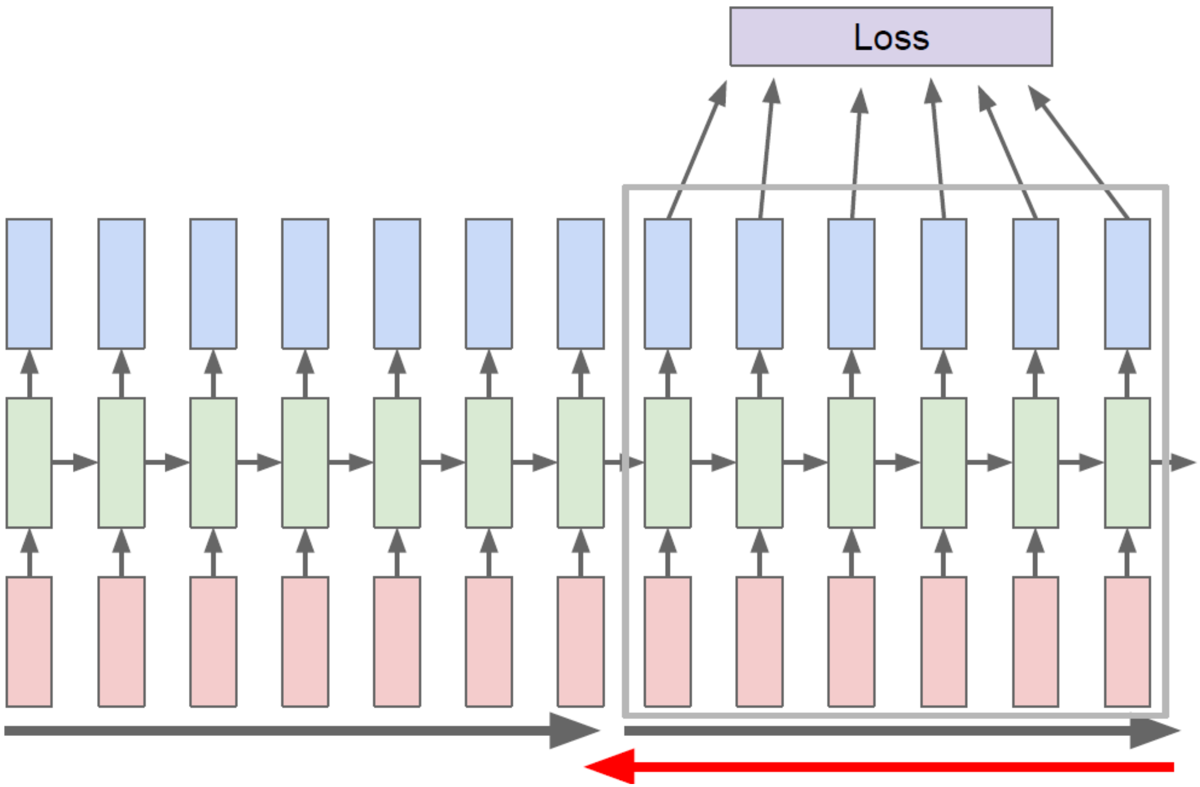
RNN Training

- Truncated Backpropagation through time
  - Instead of the entire sequence, break it up into smaller sub-sequences



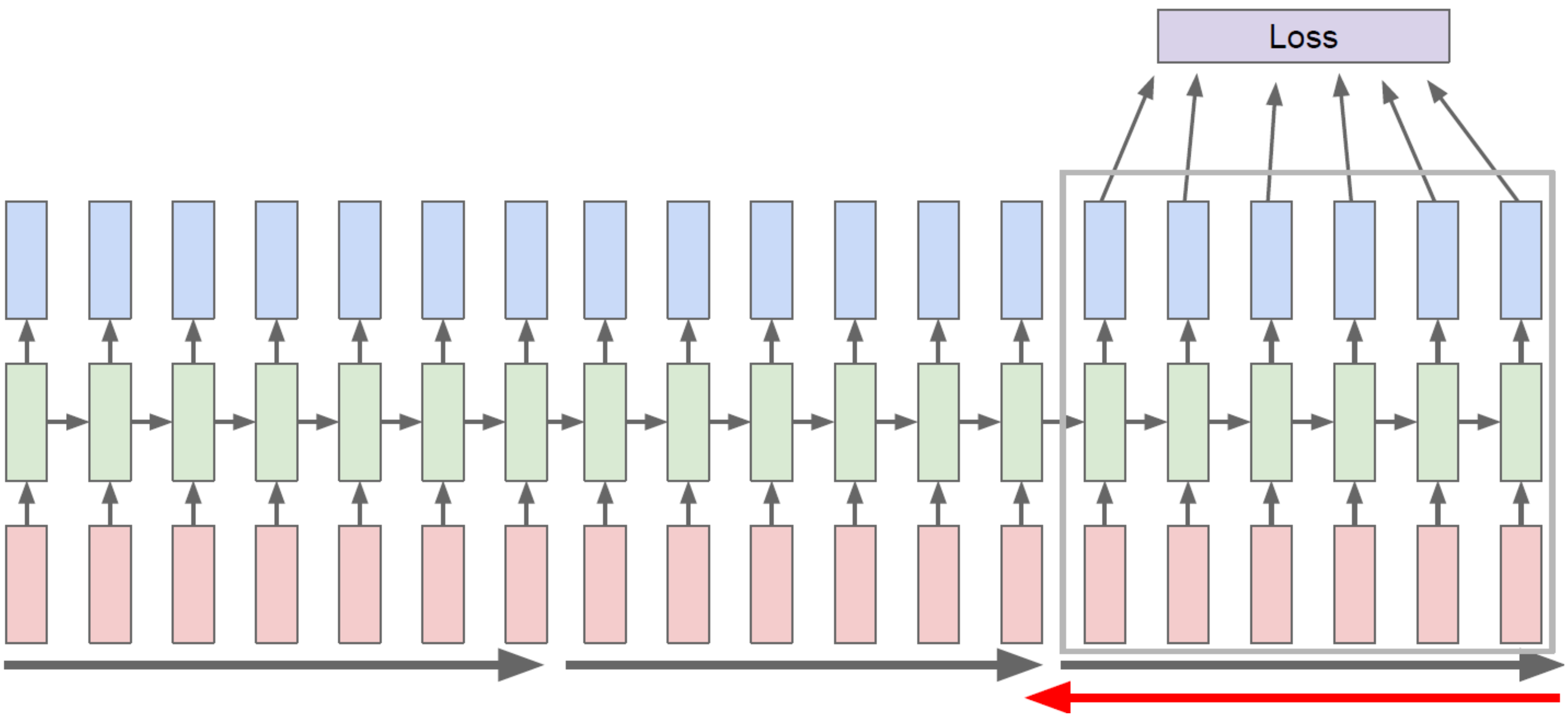
RNN Training

- Truncated Backpropagation through time
  - Perform the forward/backward pass for each sub-sequence



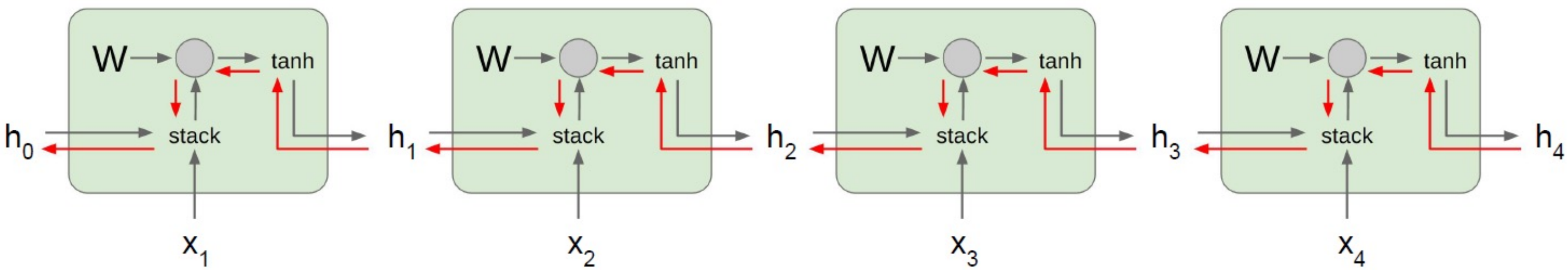
RNN Training

- Truncated Backpropagation through time



Problems with Vanilla RNNs

- Vanishing and Exploding Gradients



- Unable to remember inputs from long ago

*I live in **<u>France</u>** … … I speak fluent **<u>French</u>**.*

x_1 x_2 x_3 x_4 ⋯ ⋯ x_{57} x_{58} x_{59} x_{60}