Deep Learning (CSE5DL)

Lecture 5

Lecturer: Zhen He

Department of Computer Science & Information Technology

Outline

- Deep Learning for Natural Language Processing
  - Introduction to different NLP tasks
  - Language modelling
  - Embedding words into vectors
  - Introduction to recurrent neural networks
  - Memory networks
  - Convolutional neural networks for NLP

How can deep learning be used for natural language processing (NLP)?

Sequences

- It turns out deep learning is really good at dealing with sequences
- Words
  - Sequence of characters
- Sentences
  - Sequence of words
- Paragraphs
  - Sequence of sentences
- Document
  - Sequence of paragraphs

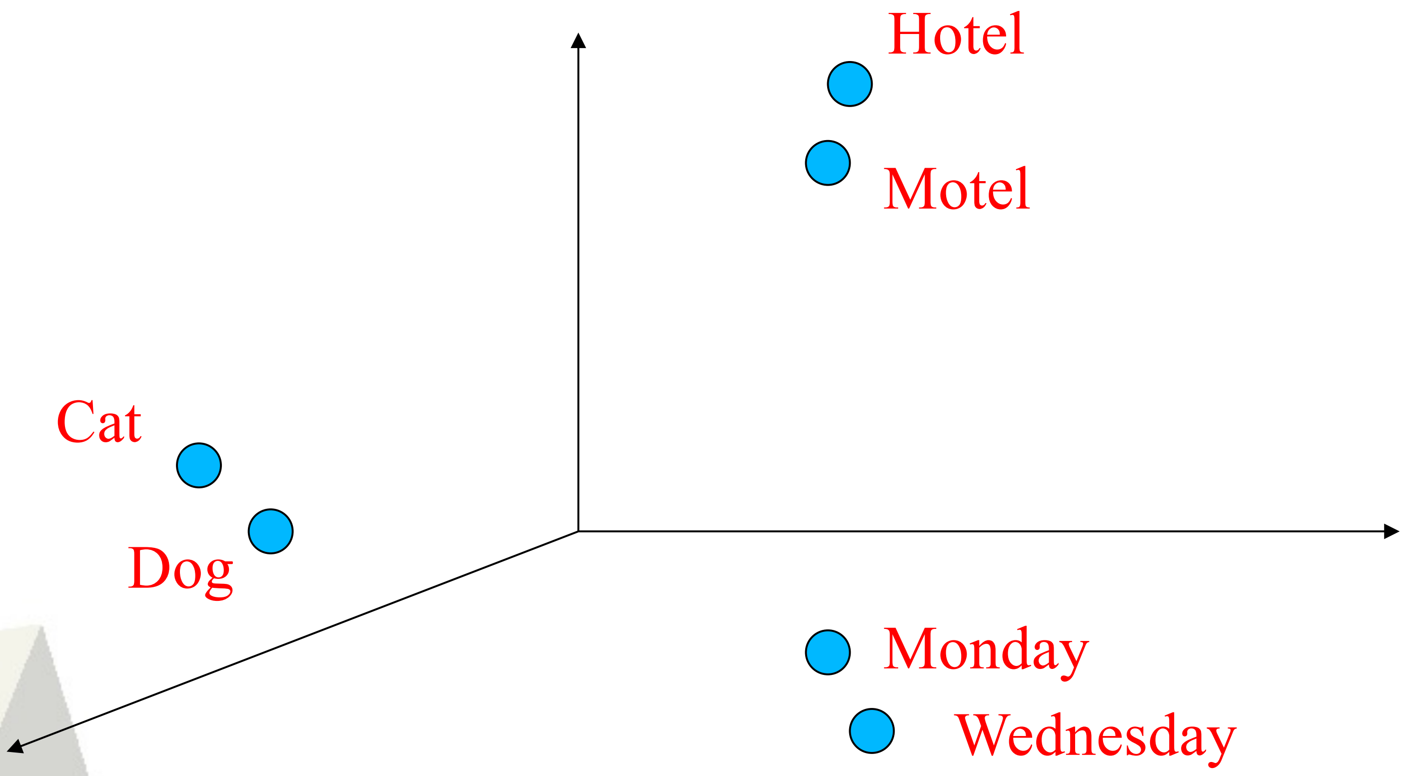
Three types of NLP problems and it is tackled by Deep Learning

- Turning a word / sentence / paragraph / document into a vector
  - Word2Vect
  - Recurrent neural networks
  - Transformer networks

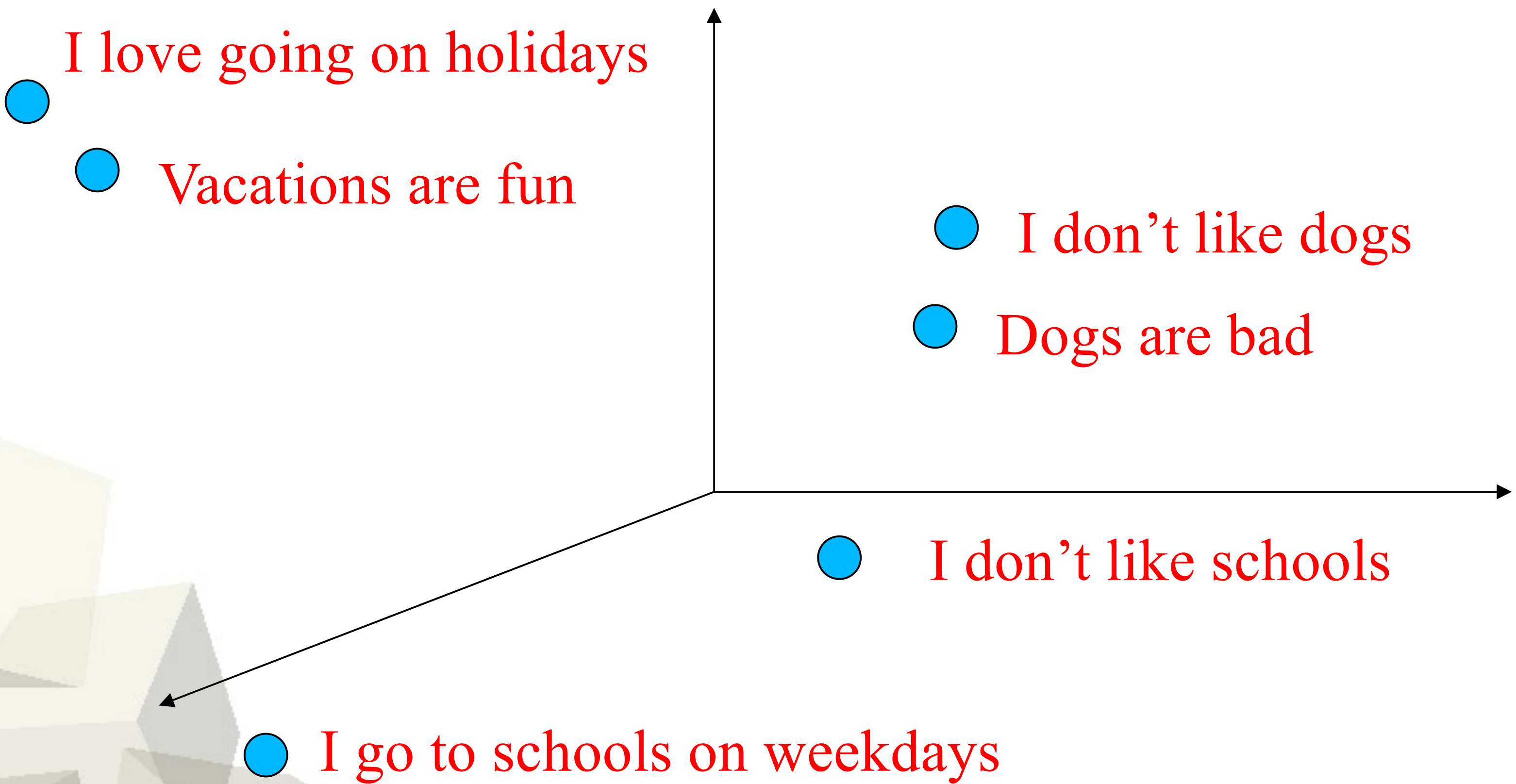
- Sequence to sequence transformations (e.g. translation)
  - Recurrent neural networks with attention
  - Transformer networks

- Question and answering
  - Memory Networks
  - Transformer networks

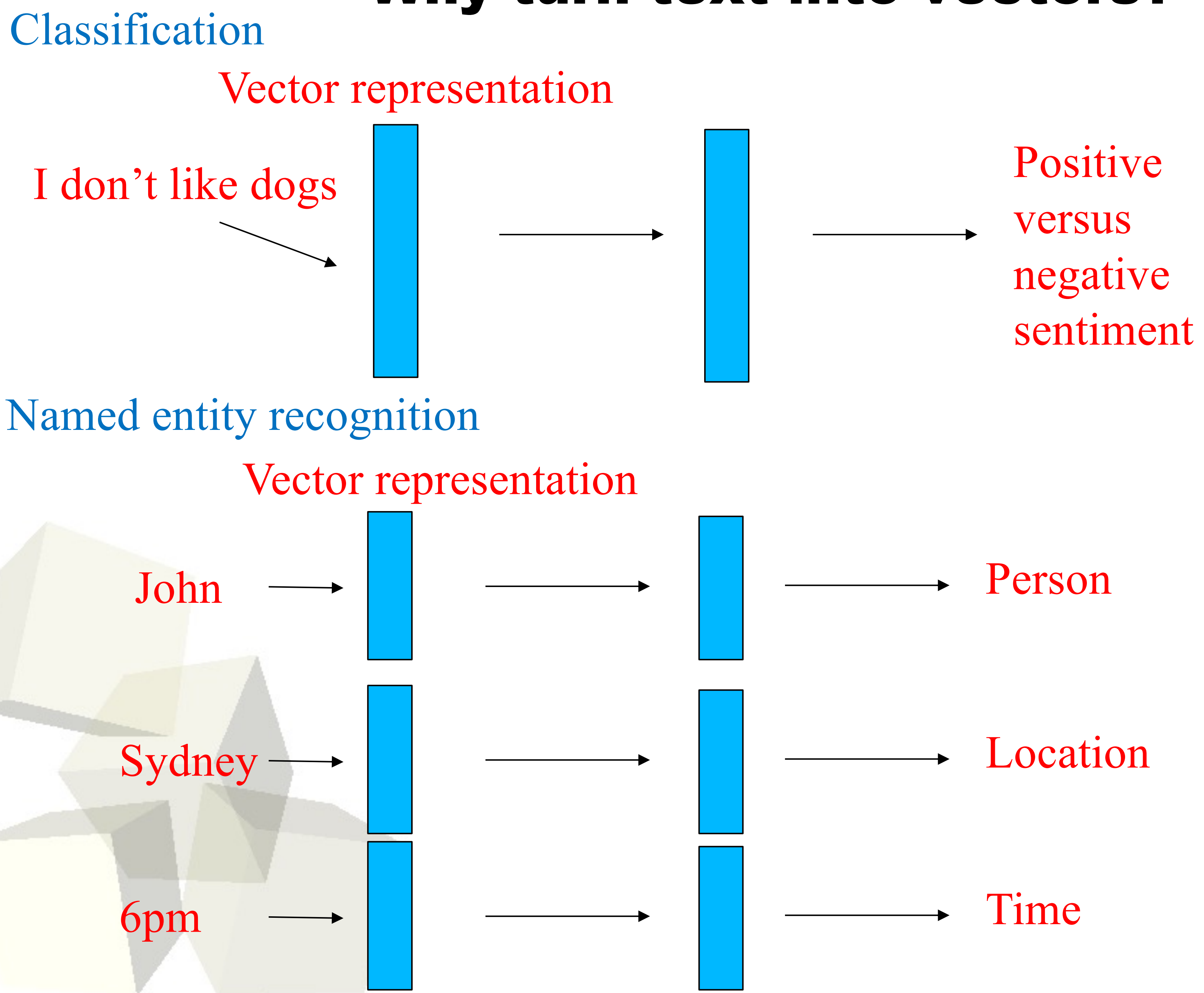
Turning a word / sentence / paragraph / document into a vector



Turning a word / sentence / paragraph / document into a vector



Why turn text into vectors?



Why turn text into vectors?

This bird has a yellow belly and tarsus, grey back, wings, and brown throat, nape with a black face

StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks

- Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, Dimitris Metaxas
- arXiv Dec 2016

Problems with this discrete representation

The vast majority of rule-based **and** statistical NLP work regards words as atomic symbols: hotel, conference, walk

In vector space terms, this is a vector with one 1 and a lot of zeroes

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

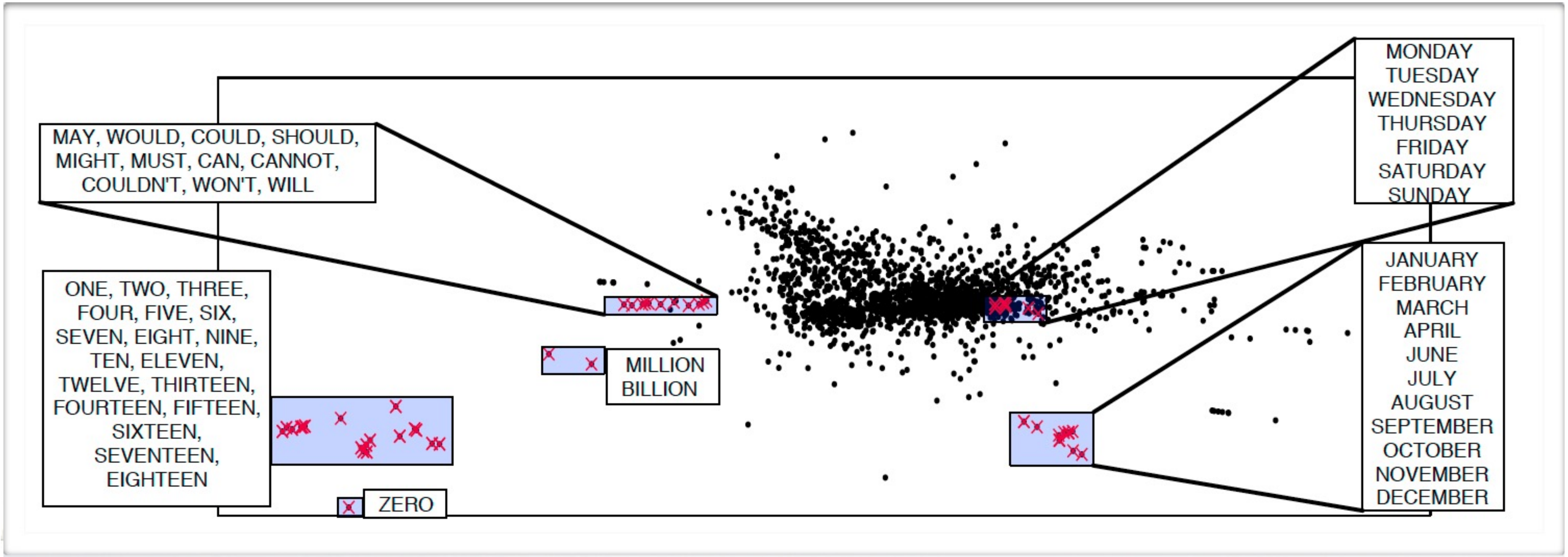
Dimensionality: 20K (speech) – 50K (PTB) – 500K (big vocab) – 13M (Google 1T)

We call this a "one-hot" representation. Its problem:

motel [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] AND
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] = 0

Slide Credit: http://cs224d.stanford.edu

Distributed Word Representation



(from Blitzer et al. 2004)

Distributed Word Representation

- Each word is associated with a vector
- The vectors are stored in a lookup table

Word	w	$C(w)$
" the "	1	[0.6762, -0.9607, 0.3626, -0.2410, 0.6636]
" a "	2	[0.6859, -0.9266, 0.3777, -0.2140, 0.6711]
" have "	3	[0.1656, -0.1530, 0.0310, -0.3321, -0.1342]
" be "	4	[0.1760, -0.1340, 0.0702, -0.2981, -0.1111]
" cat "	5	[0.5896, 0.9137, 0.0452, 0.7603, -0.6541]
" dog "	6	[0.5965, 0.9143, 0.0899, 0.7702, -0.6392]
" car "	7	[-0.0069, 0.7995, 0.6433, 0.2898, 0.6359]
...	...	...

Distributed Word Representation (Word2Vec)

- One of the most popular distributed word representations is the word2vec algorithm from Mikolov et al.
  - Word2Vec: Efficient Estimation of Word Representations in Vector Space
    - Tomas Mikolov, et. al.
    - arXiv 2013
- It is a very simple use of neural networks.
- The learning can be done very quickly.
  - Large document collection with 100 billion words trained in 1 day.
- Can download pre-trained word vectors

https://code.google.com/archive/p/word2vec/

How does word2vec work?

- Use the middle word to predict the surrounding words

I play soccer on Saturday

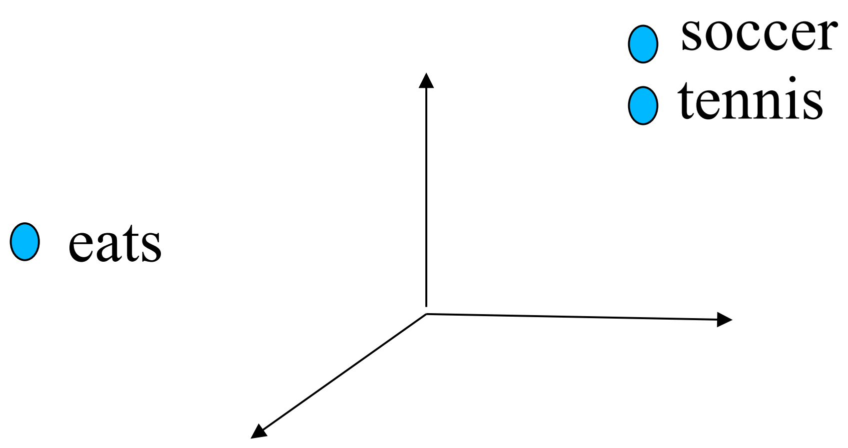
Predict these

I play tennis on Saturday

Predict these

My dog eats a lot of food

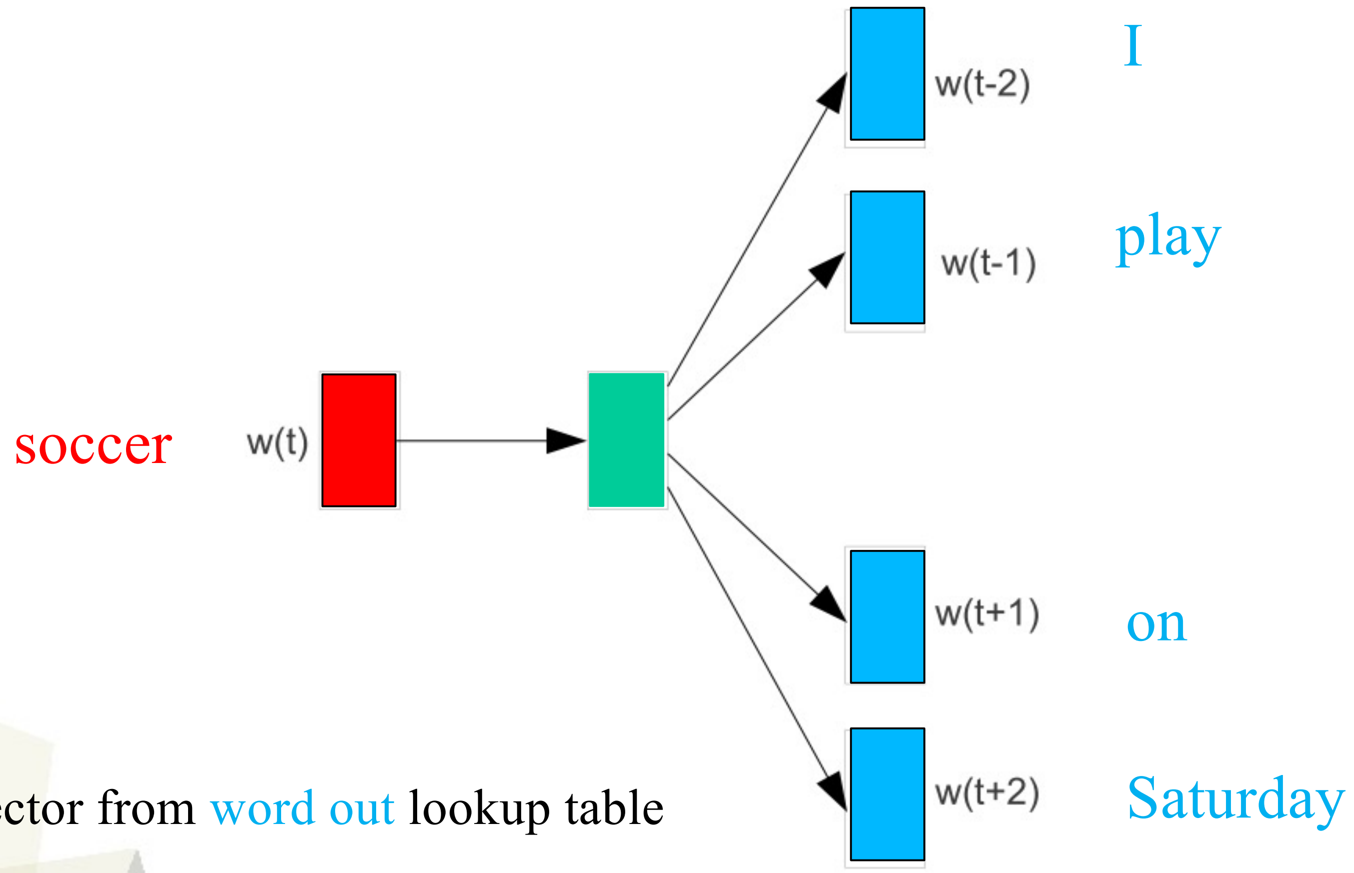
Predict these



The words soccer and tennis both predict similar surrounding words so they will get embedding vectors that are close to each other in the vector space.

The word eats appears in a different context

Word2Vec



Vector from word out lookup table

Vector from word in lookup table

Prob of word out given prob of word in

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)}$$

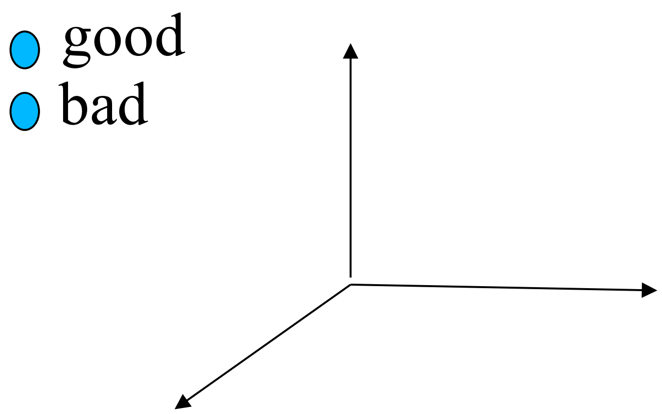
Details of Word2Vec

- Predict surrounding words in a window of length m of every word.
- Objective function: Maximize the log probability of any context word given the current center word:
- $$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0} \log p(w_{t+j}|w_t)$$
- Where θ represents all variables we optimize

Slide Credit: http://cs224d.stanford.edu

Word2Vec Downside

- Words like “good” and “bad” often occur in the same context
  - E.g. The dress looks good. The dress looks bad.
  - So in the words good and bad are likely to be close to each other in the vector space.



Unsupervised Word Embeddings

- Word2Vect
- FastText
  - Very fast implementation
  - Very popular
  - From facebook
- Glove Vectors

- You can download pre-trained word embeddings for all three above
  - Trained on large text data sets.

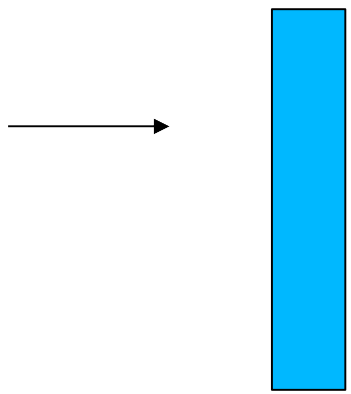
Representing a sentence or paragraph as a vector

- Word2Vec produces a vector per word.
  - Many applications require a vector per sentence or paragraph.
- How do we use word2vec when multiple words are involved?
- Do you care about the order of the words?
  - If no
    - Average word2vec vectors
    - Average word2vec vectors with TF-IDF
      - Take word2vec vectors multiple with TF-IDF
    - Paragraph vector
      - Distributed Representations of Sentences and Documents
      - Quoc Le and Tomas Mikolov
      - ICML 2014
  - If yes
    - N-grams
    - Recurrent neural networks
      - Skip thought vectors

Vector for sequence of words

- Word2Vec gives you one vector per word.
- Often you want to convert a sequence of words into one vector

This is a sentence

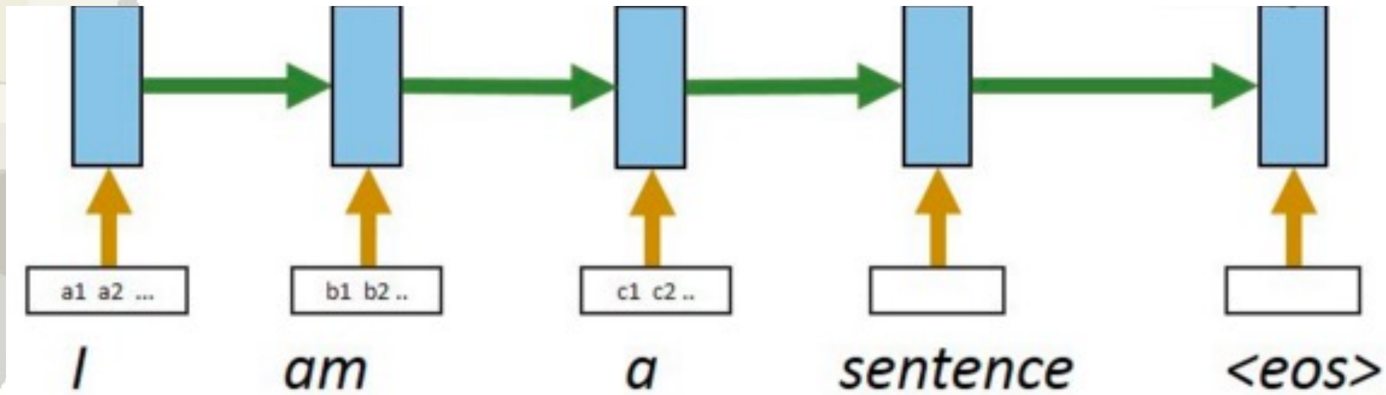


Deep Learning allows you to take variable length input

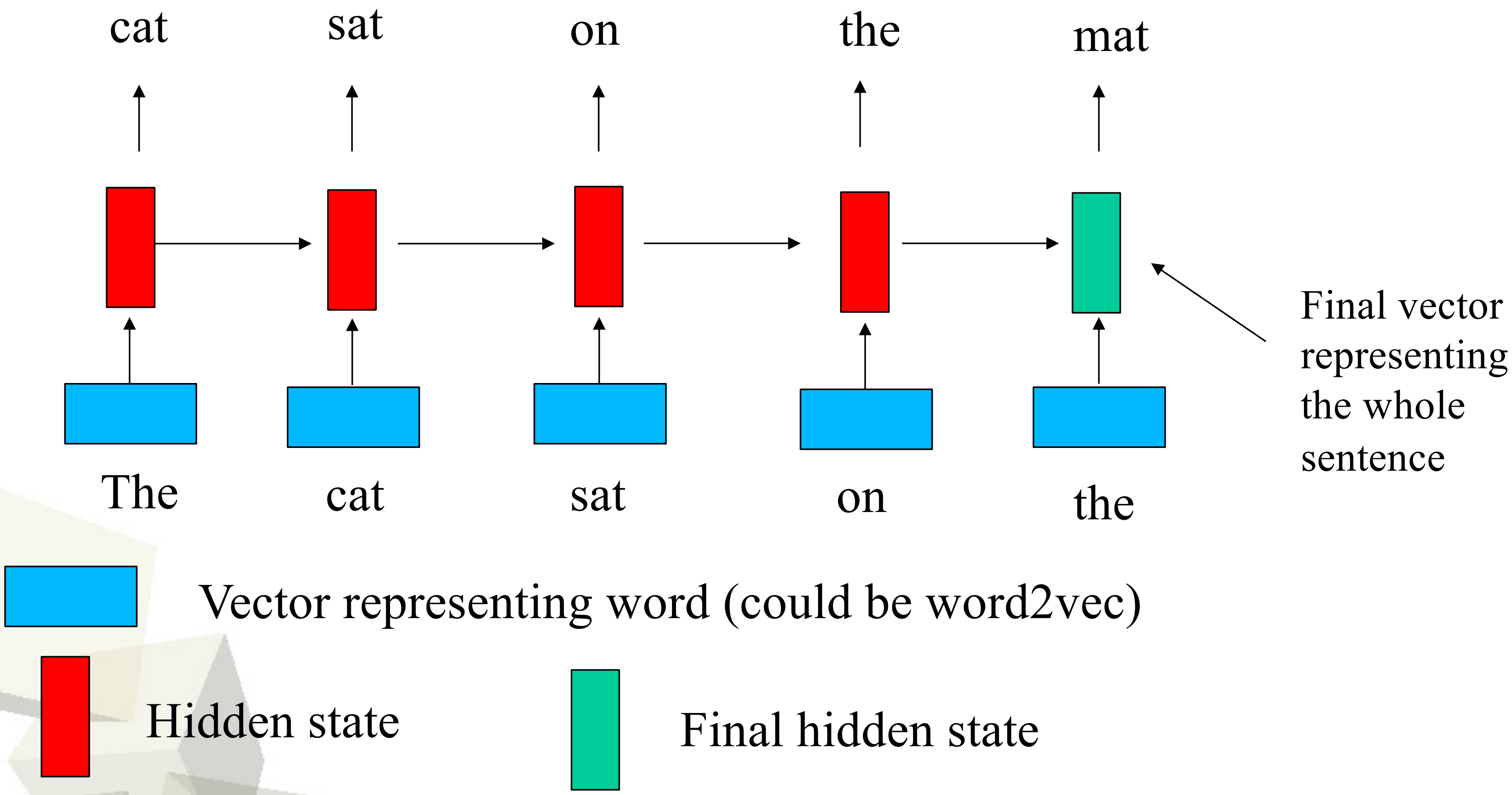
- Traditional machine learning
  - N-grams
    - unigrams(n=1):"is","a","sequence",etc.
    - bigrams(n=2): ["is","a"], ["a","sequence"],etc.
    - trigrams(n=3): ["is","a","sequence"],["a","sequence","of"],etc.

Deep Learning

- Recurrent neural networks
  - Recurrent neural networks in theory allow you to take context information from infinite number of steps in the past.



How to train a recurrent neural network?

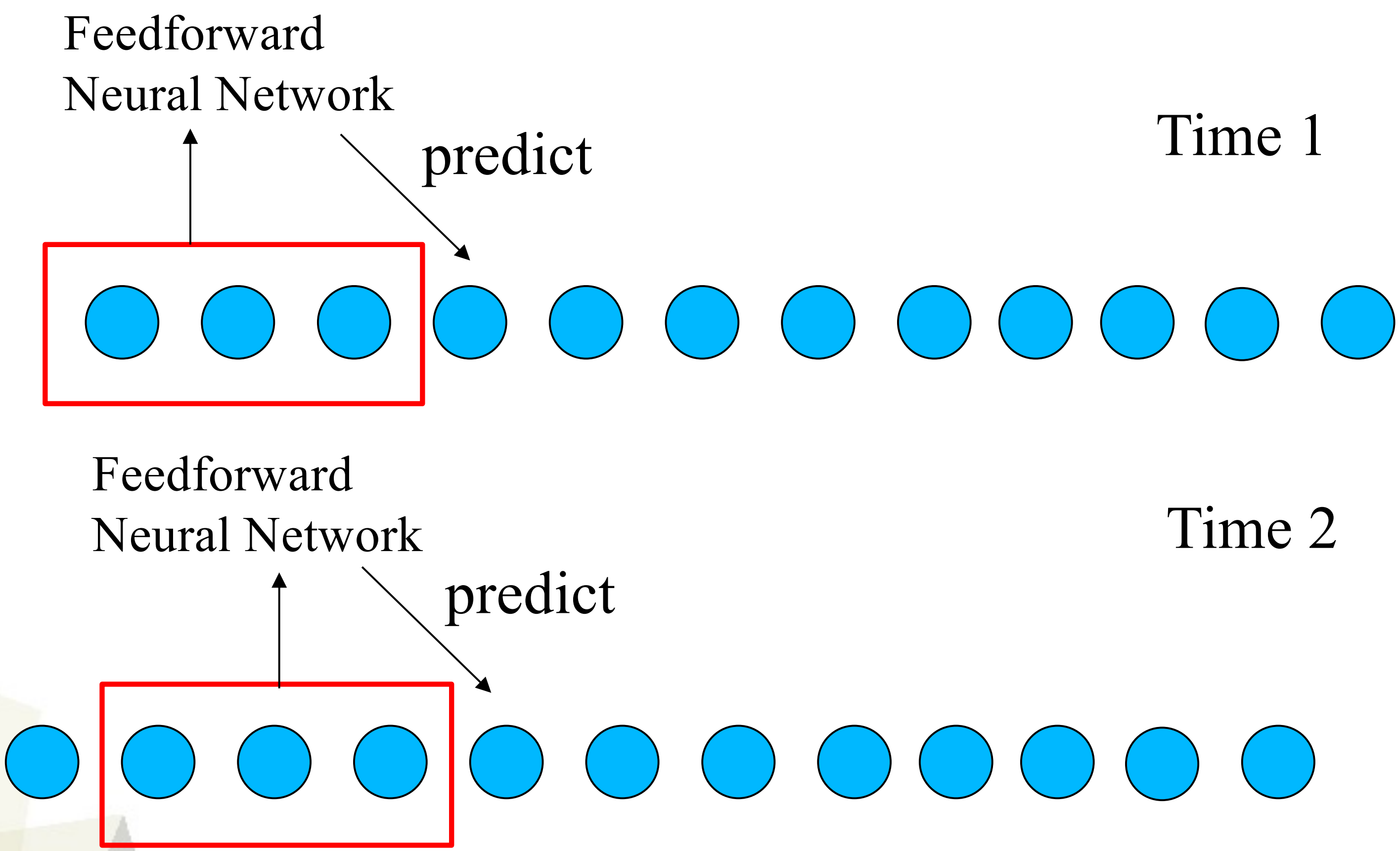


- We can train recurrent neural networks to predict the next word.
- The final hidden state of the network can be used as the representation of the sentence.

Recurrent Neural Networks (RNN)

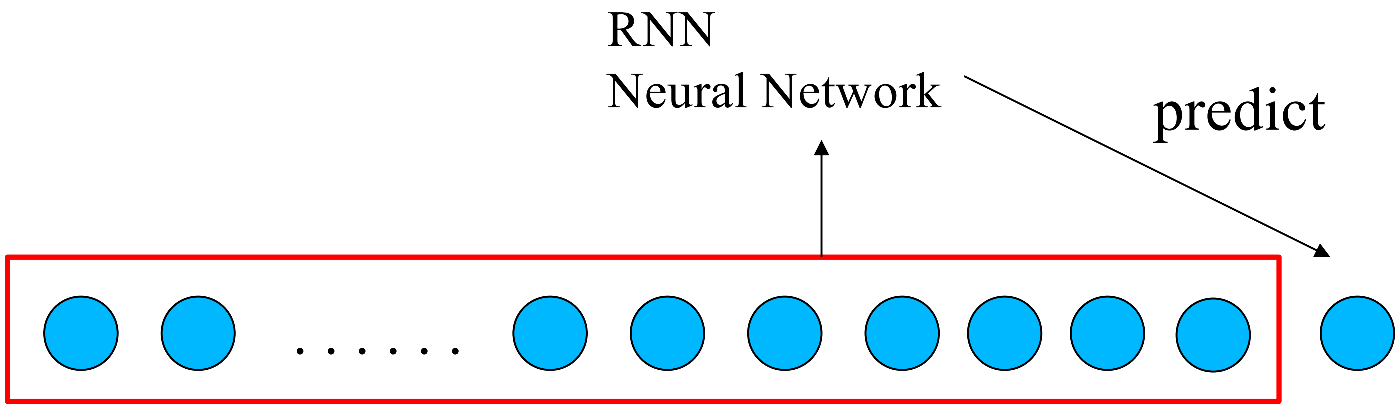
The RNN chapter from this book is really good:
http://www.deeplearningbook.org

Learning from Sequential Data



- Normal Feedforward Neural Networks can learn from sequential data
- Slide fixed sized window and feed into feedforward neural network
- But can only take context information from a fixed number of time steps in the past.

Recurrent Neural Networks (RNNs)

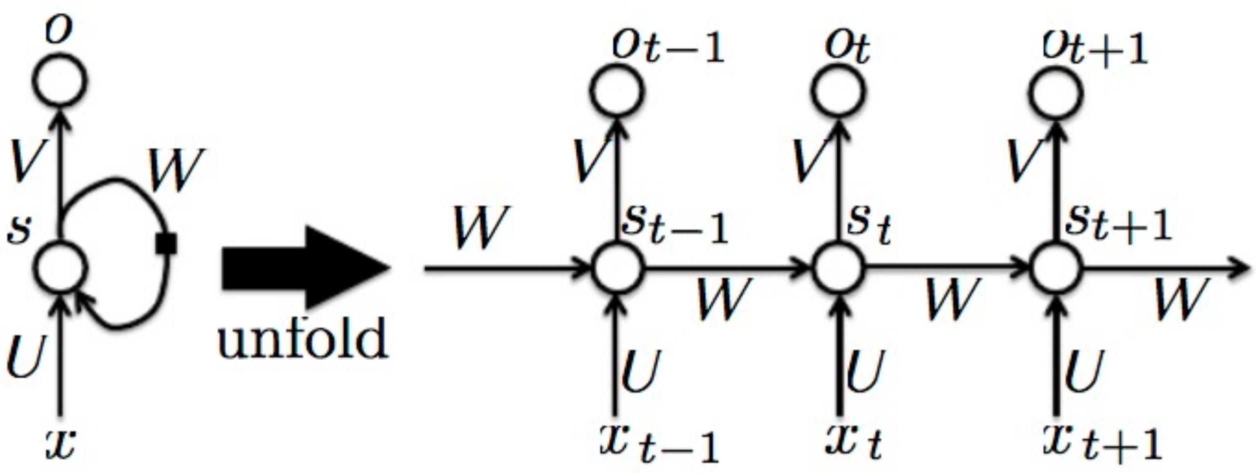


- Recurrent neural networks in theory allow you to take context information from infinite number of steps in the past.

Recurrent Neural Networks (RNNs)

Output

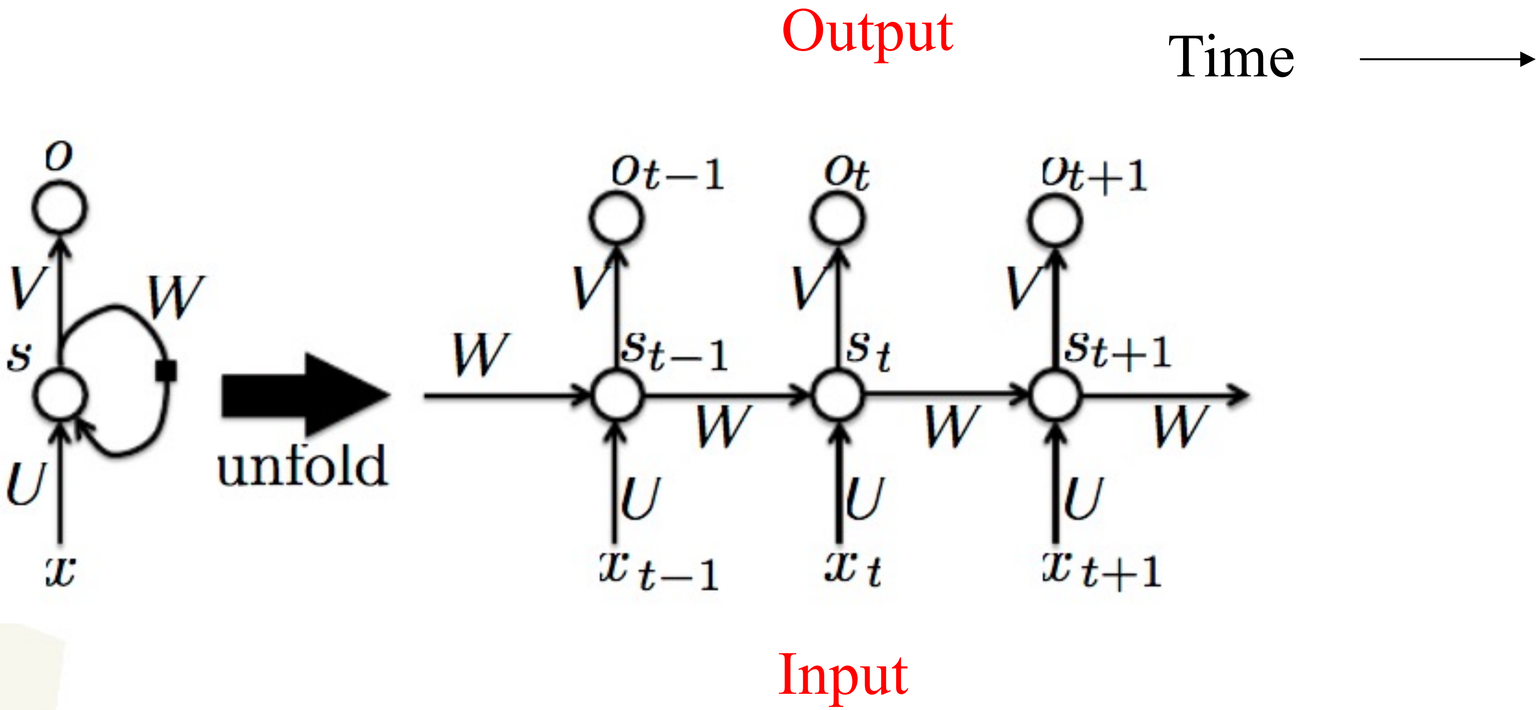
Time



Input

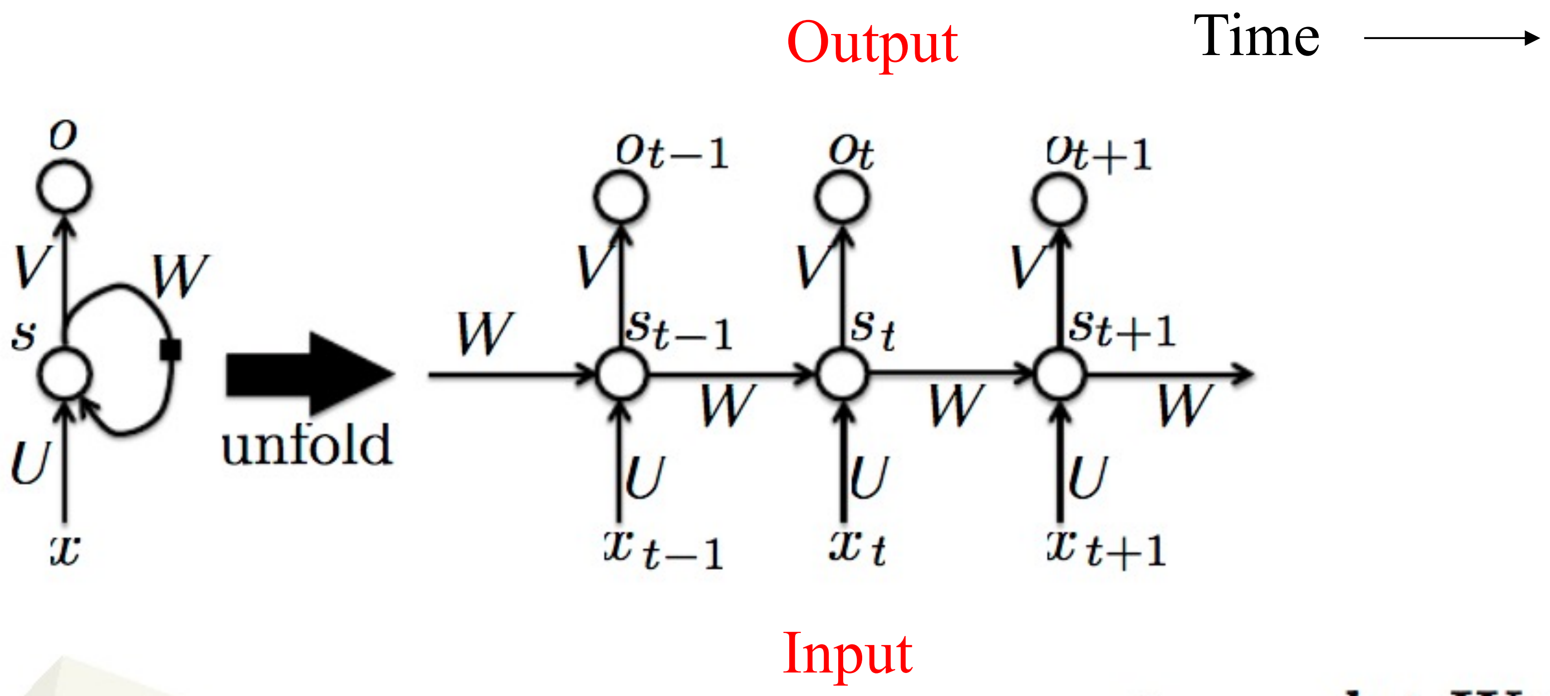
- RNNs allow information to be backpropagated across time

Recurrent Neural Networks (RNNs)



- RNNs are universal so in theory any function that can be computed by a Turing machine can be computed by a RNN of finite size.

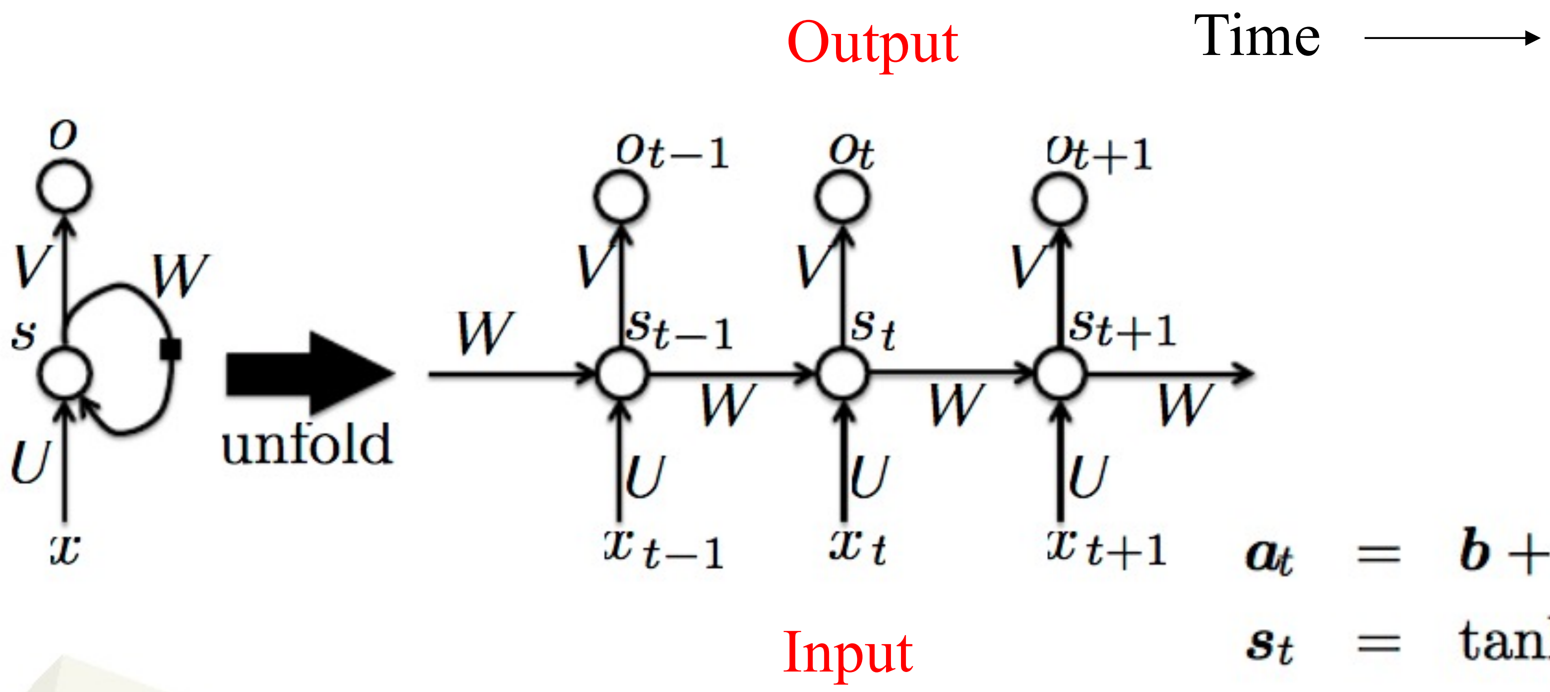
Recurrent Neural Networks



- x_t
  - Input at time t
- s_t
  - Hidden state at time t
- o_t
  - Output at time t

$$
\begin{aligned}
\boldsymbol{a}_t &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{s}_{t-1} + \boldsymbol{U}\boldsymbol{x}_t \\
\boldsymbol{s}_t &= \tanh(\boldsymbol{a}_t) \\
\boldsymbol{o}_t &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{s}_t \\
\boldsymbol{p}_t &= \operatorname{softmax}(\boldsymbol{o}_t)
\end{aligned}
$$

Recurrent Neural Networks



$$\begin{aligned} \boldsymbol{a}_t &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{s}_{t-1} + \boldsymbol{U}\boldsymbol{x}_t \\ \boldsymbol{s}_t &= \tanh(\boldsymbol{a}_t) \\ \boldsymbol{o}_t &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{s}_t \\ \boldsymbol{p}_t &= \mathrm{softmax}(\boldsymbol{o}_t) \end{aligned}$$

- U, V, W are weights of fully connected (linear layers) shared across time
  - This means it is the same weights used at each time step

Recurrent Neural Networks Loss Function

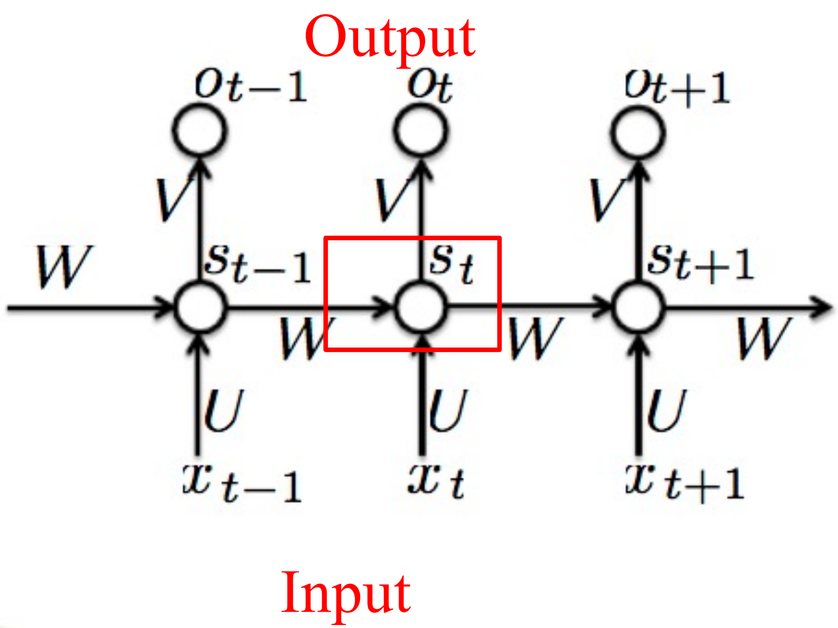
- Loss for given input and output sequence pair (x, y) is the following:

$$L(\boldsymbol{x}, \boldsymbol{y}) = \sum_t L_t = \sum_t - \log p_{y_t} \qquad (10.5)$$

where y_t is the category that should be associated with time step t in the output sequence.

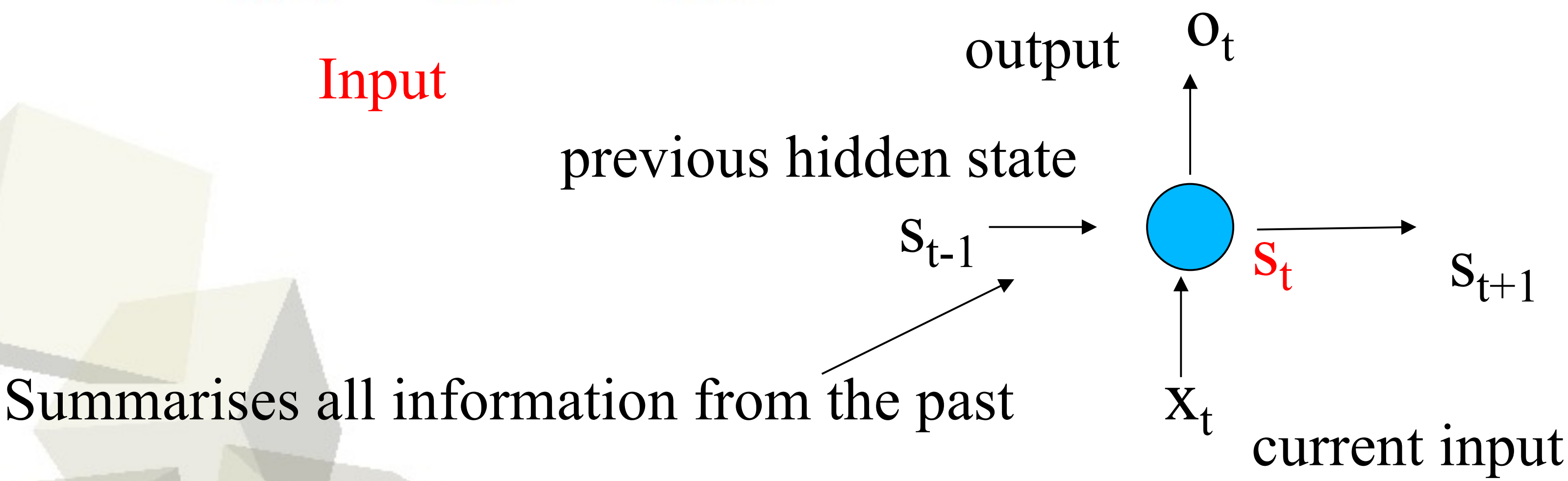
- So you want to minimize the sum of the loss across time.

The Hidden Layer (S_t) of a RNN



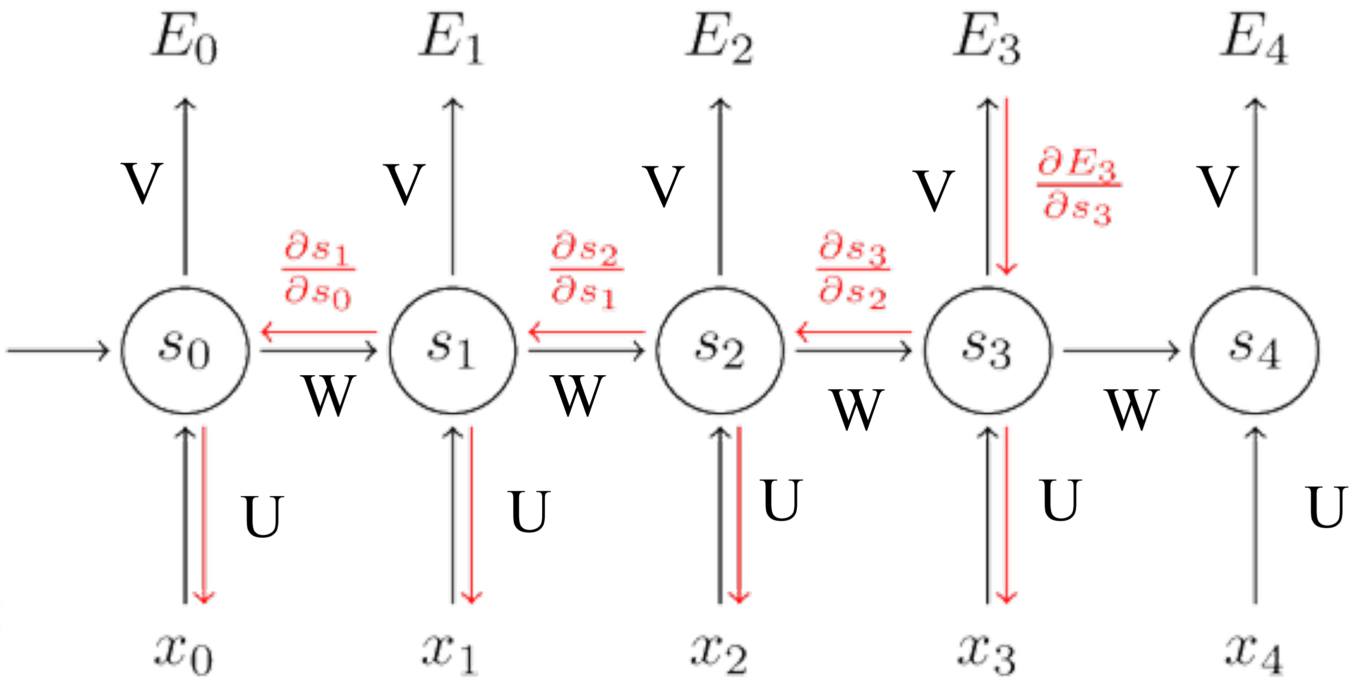
Time ⟶

$$\begin{aligned} \boldsymbol{a}_t &= \boldsymbol{b} + \boldsymbol{W}\boxed{\boldsymbol{s}_{t-1}} + \boldsymbol{U}\boxed{\boldsymbol{x}_t} \\ \boxed{\boldsymbol{s}_t} &= \tanh(\boldsymbol{a}_t) \\ \boldsymbol{o}_t &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{s}_t \\ \boldsymbol{p}_t &= \operatorname{softmax}(\boldsymbol{o}_t) \end{aligned}$$



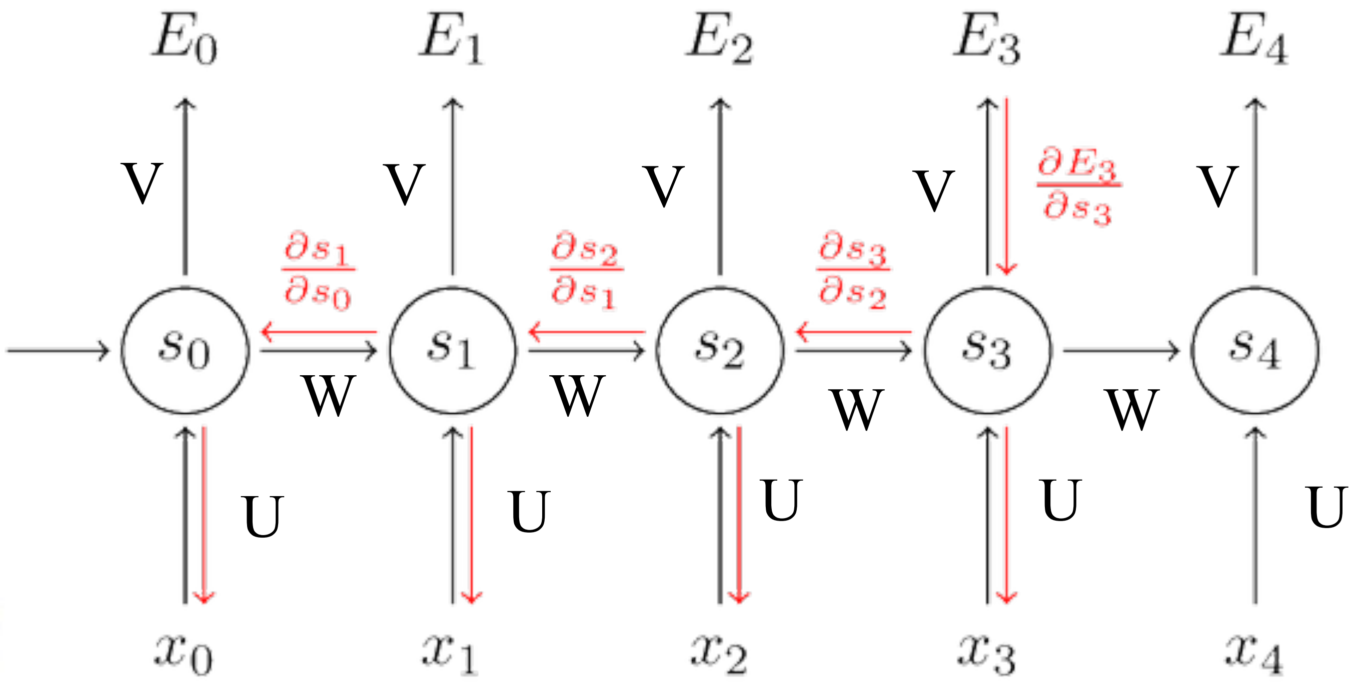
- So the RNN uses s_{t-1} which contains all the context information from the past and the current input and then decides what to output.

RNN Backpropagation



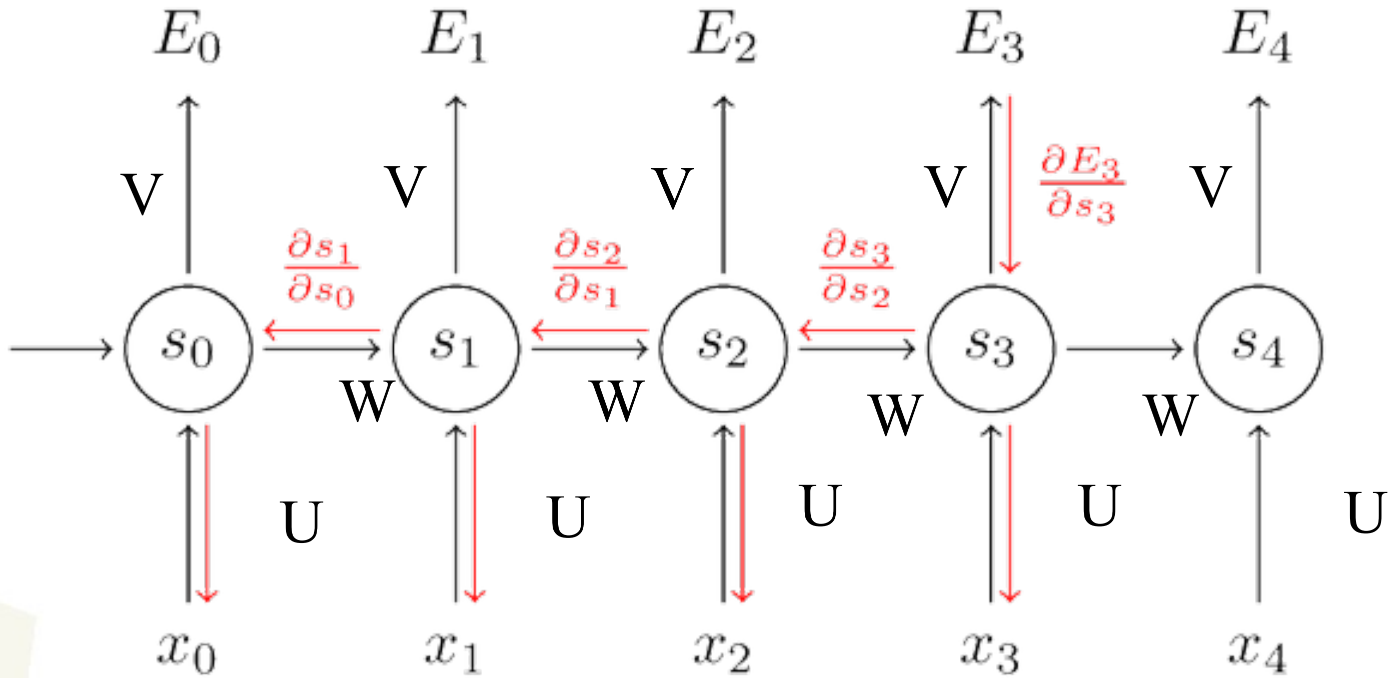
- Backpropagate a fixed number of time steps backwards.
- To simplify discussion assume no non-linearity tanh, so s_t equals the following
  - $s_t = b + Ws_{t-1} + Ux_t$
- $ds_t / ds_{t-1} = W$
  - Using chain rule $ds_t / ds_{t-2} = ds_t / ds_{t-1} * ds_{t-1} / ds_{t-2}$
- So $ds_t / ds_{t-2} = W^2$, $ds_t / ds_{t-3} = W^3$, … $ds_t / ds_{t-(k-1)} = W^{k+1}$

RNN Backpropagation



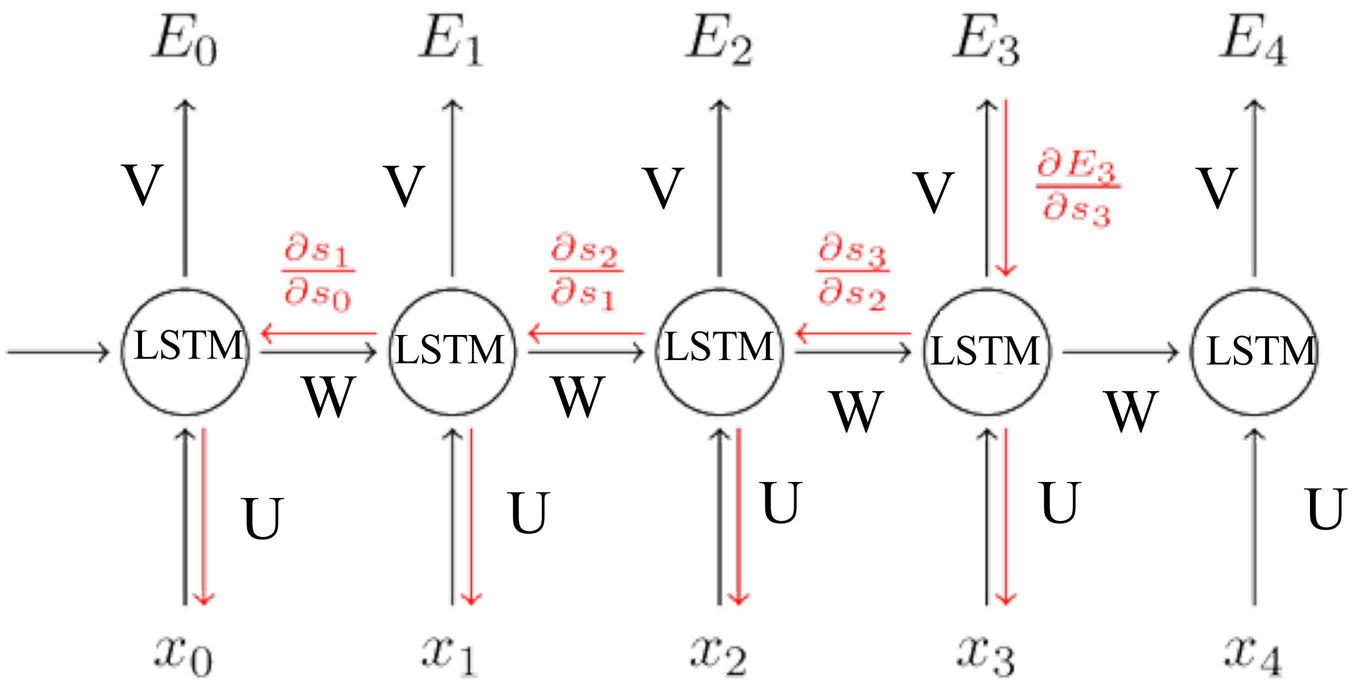
- $ds_t / ds_{t-(k-1)} = W^{k+1}$
- What happens when k is large
  - and the eigen value of W is greater than 1?
    - Gradient explodes
  - and the eigen value of W is less than 1?
    - Gradient vanishes

Dealing with exploding gradient



- $ds_t / ds_{t-(k-1)} = W^{k+1}$, and eigen value of $W > 1$
- Dealing with exploding gradient is quite easy
  - Just clip the gradient, so it is never greater than some threshold.

Dealing with vanishing gradient (LSTM)



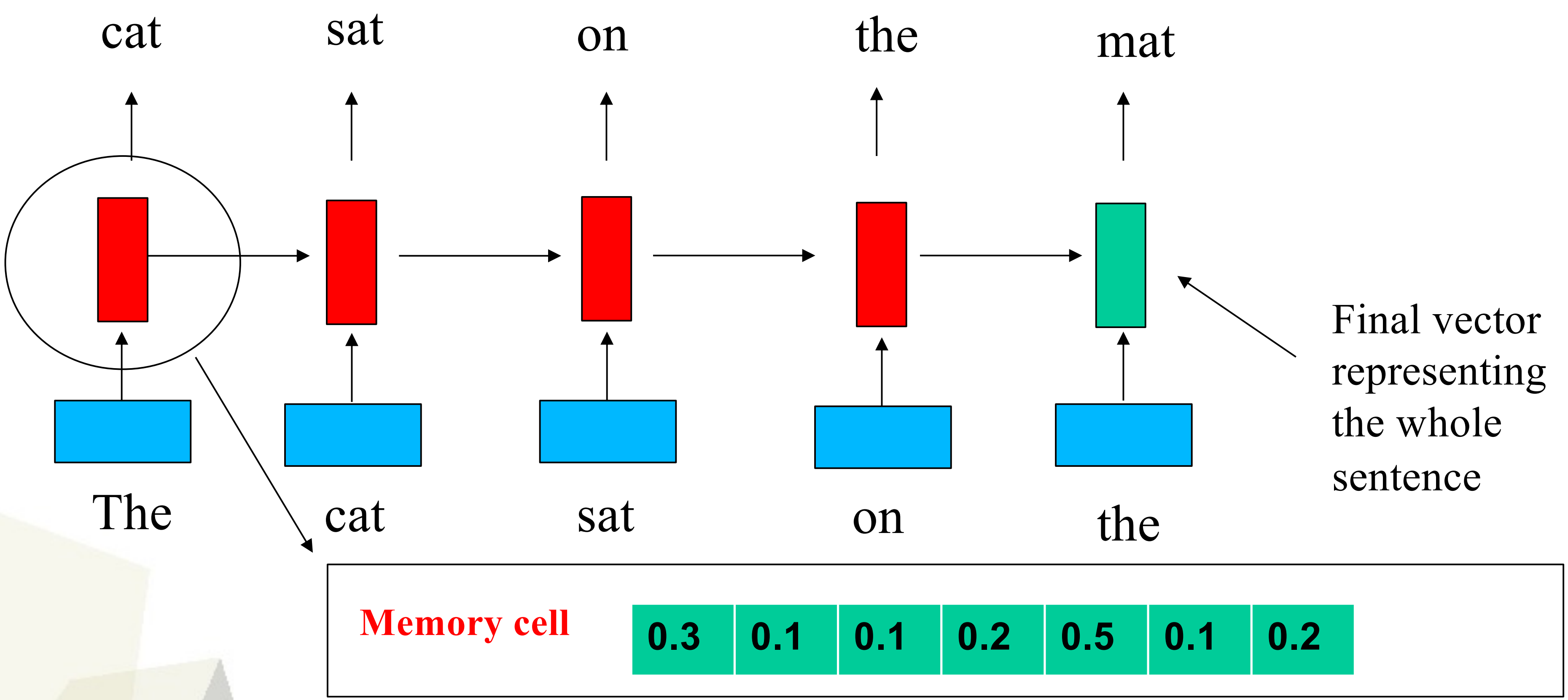
- Vanishing gradient occurs: ds_t / $ds_{t-(k-1)}$ = W^{k+1} and eigen value of W < 1
- Replace the recurrent hidden layer with a recurrent LSTM hidden layer
  - Long Short Term Memory (LSTM)
- LSTM uses special gating mechanisms to allow selective gradients to pass backwards undisturbed (effectively multiplying by a gradient of 1).

Long Short Term Memory (LSTM)

- I will give a very very simplified description of of how a Long Short Term Memory (LSTM) works

- In practice when using RNNs, people always use LSTMs or some other variants of it like GRU

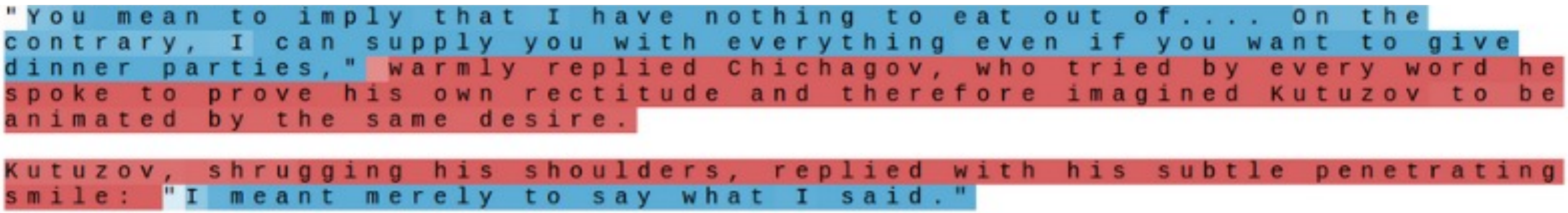
What is inside the hidden state of an LSTM?



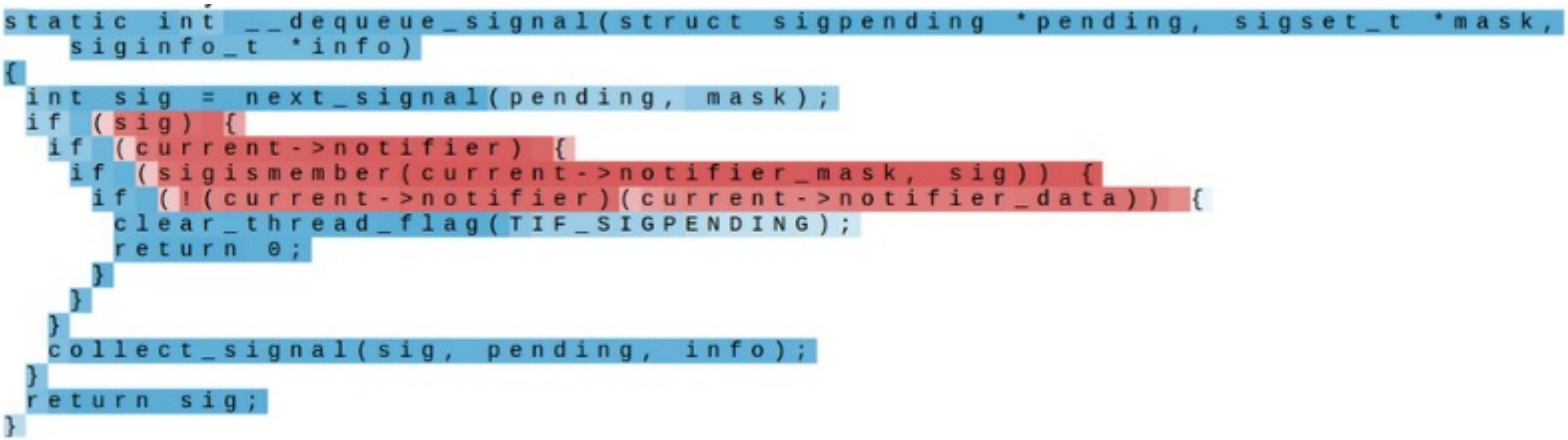
- The hidden state has a vector of memory cells
  - The system automatically assigns a different purpose for each cell.

Examples of cell functions

- Cell that turns on inside quotes

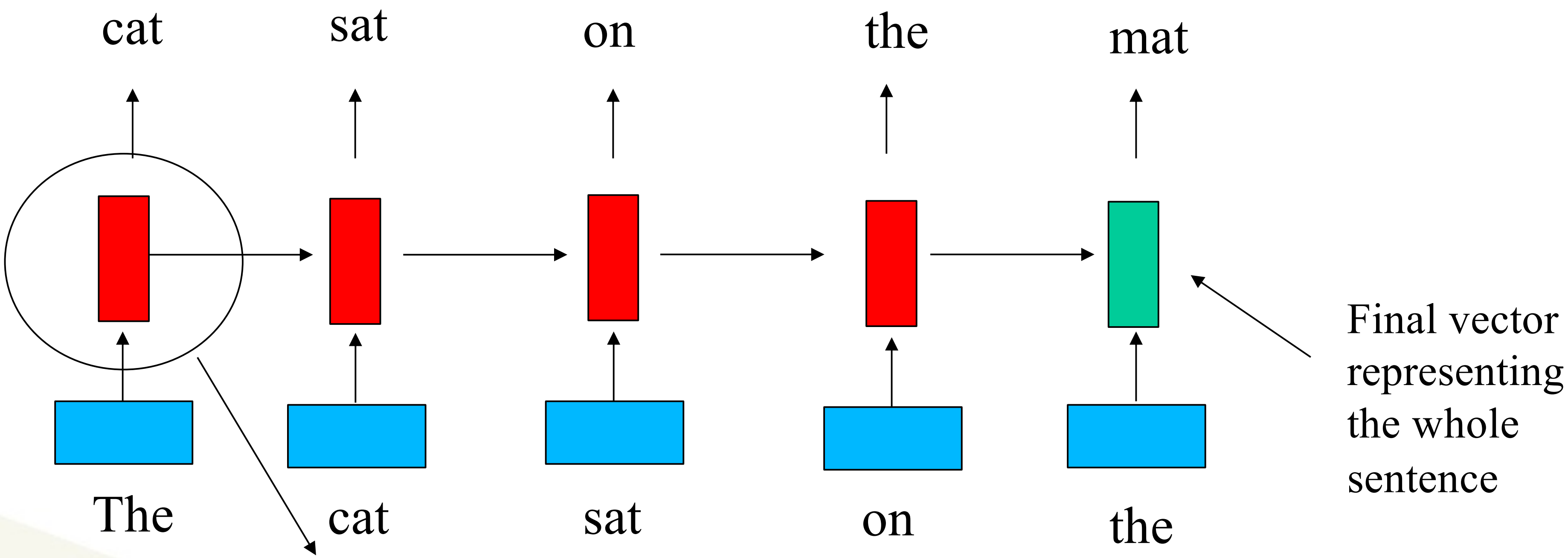


- Cell that robustly activates inside if statements



Slide credit: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

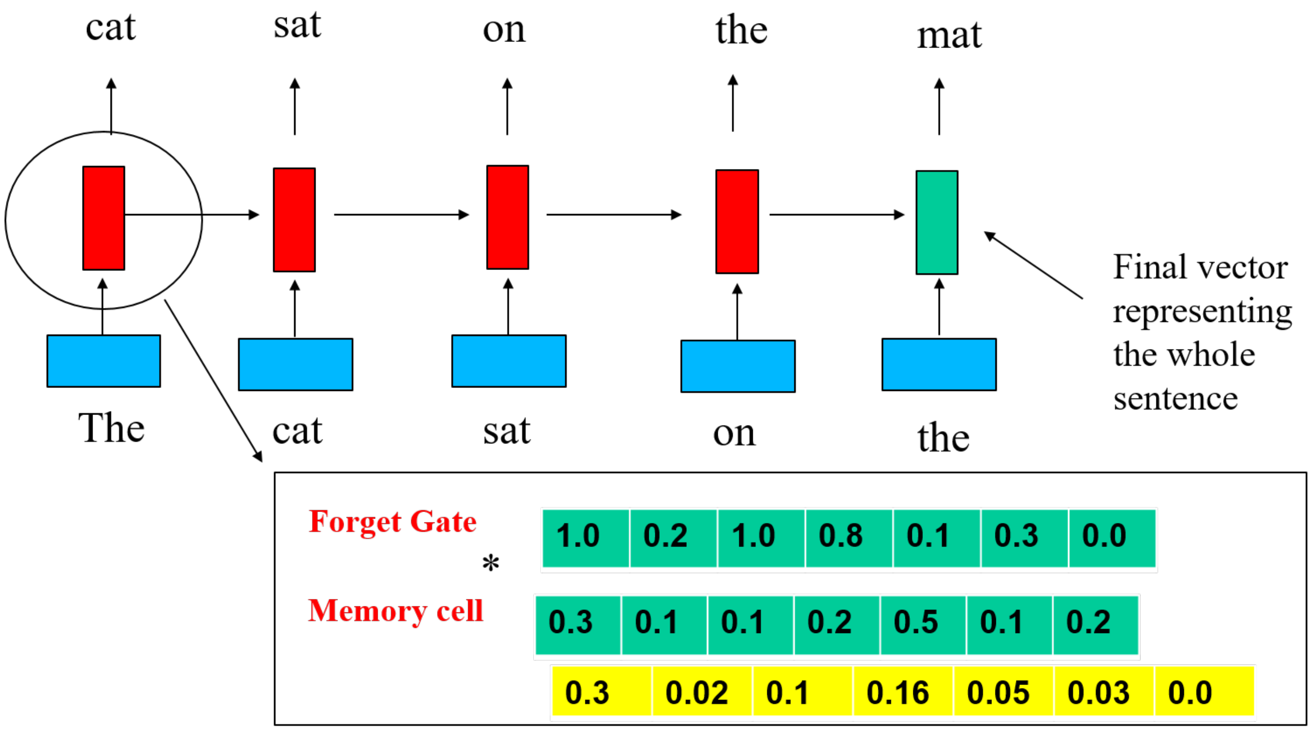
What is inside the hidden state of an LSTM?



Forget Gate		1.0	0.2	1.0	0.8	0.1	0.3	0.0
	*							
Memory cell		0.3	0.1	0.1	0.2	0.5	0.1	0.2
	=	0.3	0.02	0.1	0.16	0.05	0.03	0.0

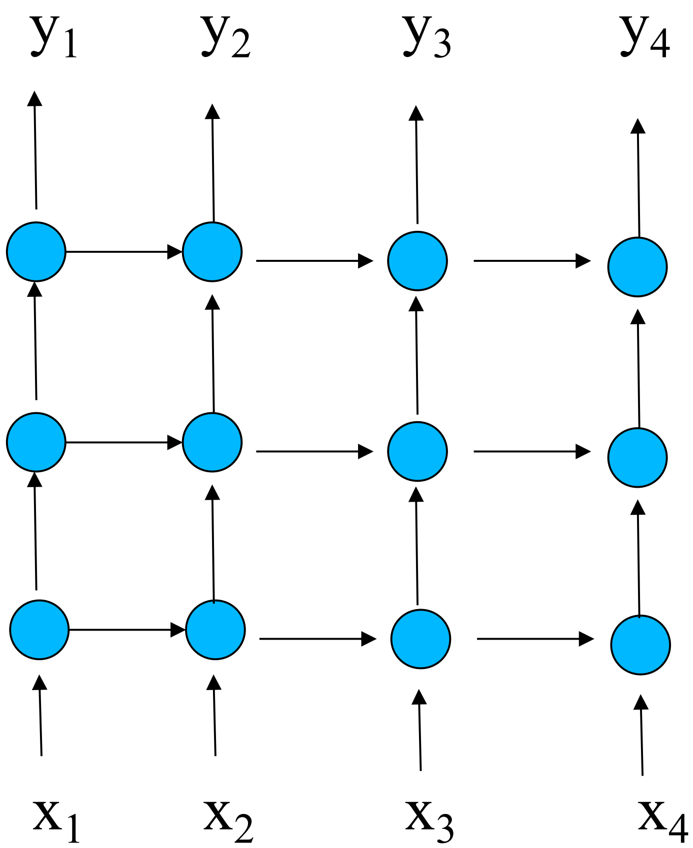
- The forget gate is used to decide when to forget things

What does the forget gate allow us to do?



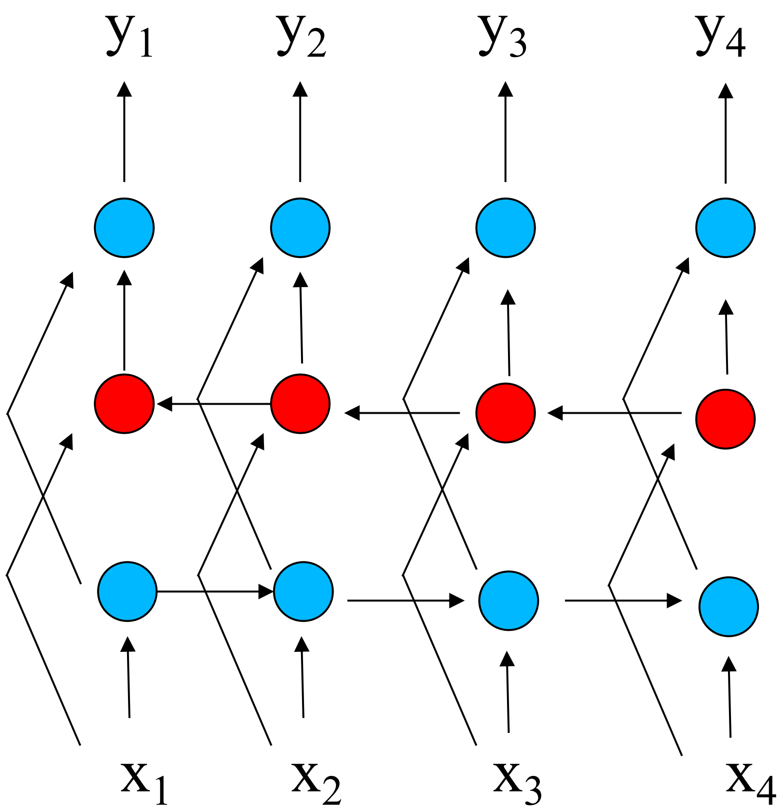
- Allow information to accumulate over a long time
- Once information is consumed and no longer needed the gate allows the RNN to forget the old state
- For example
  - Learn sequence (zhen he) (water)
  - After we encountered the first) we want to forget all the contents in the brackets and start again.

Deep RNNs



- We can add more hidden layers like the above to enable the RNN to model more complex recurrent relationships
- For example model patterns at different time scales

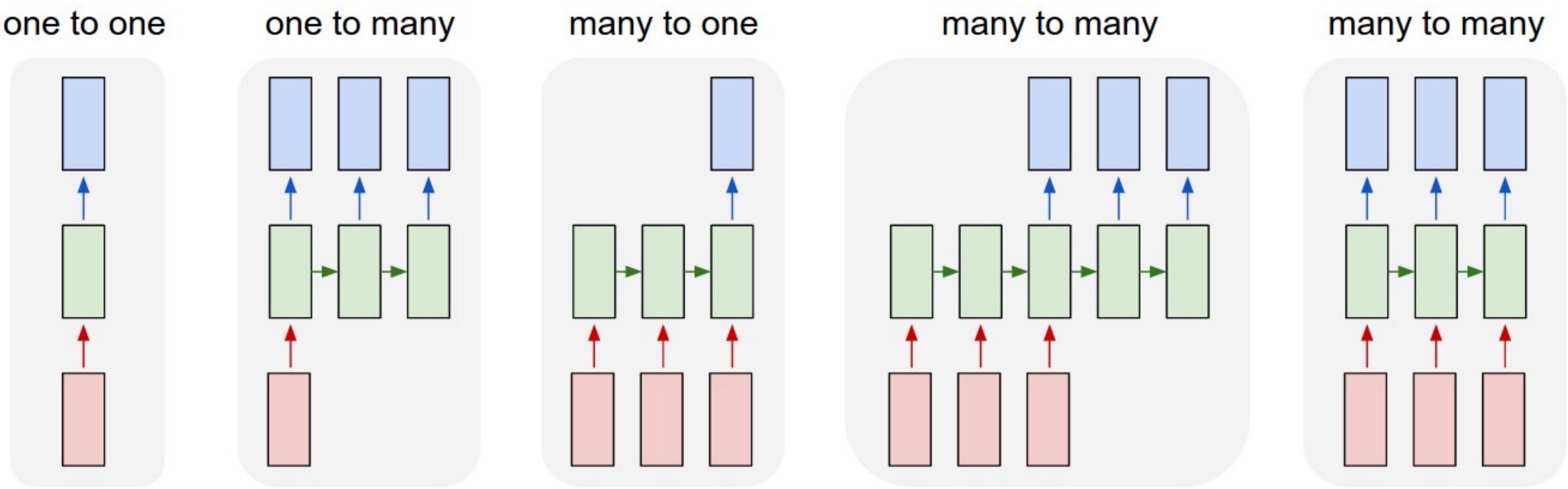
Bi-directional RNNs



- We can add a hidden layer whose connections are connected backwards.
- This allows the RNN to benefit from relevant information in the future instead of just from the past.
- For example for speech recognition the current phoneme may depend on the next few phonemes.

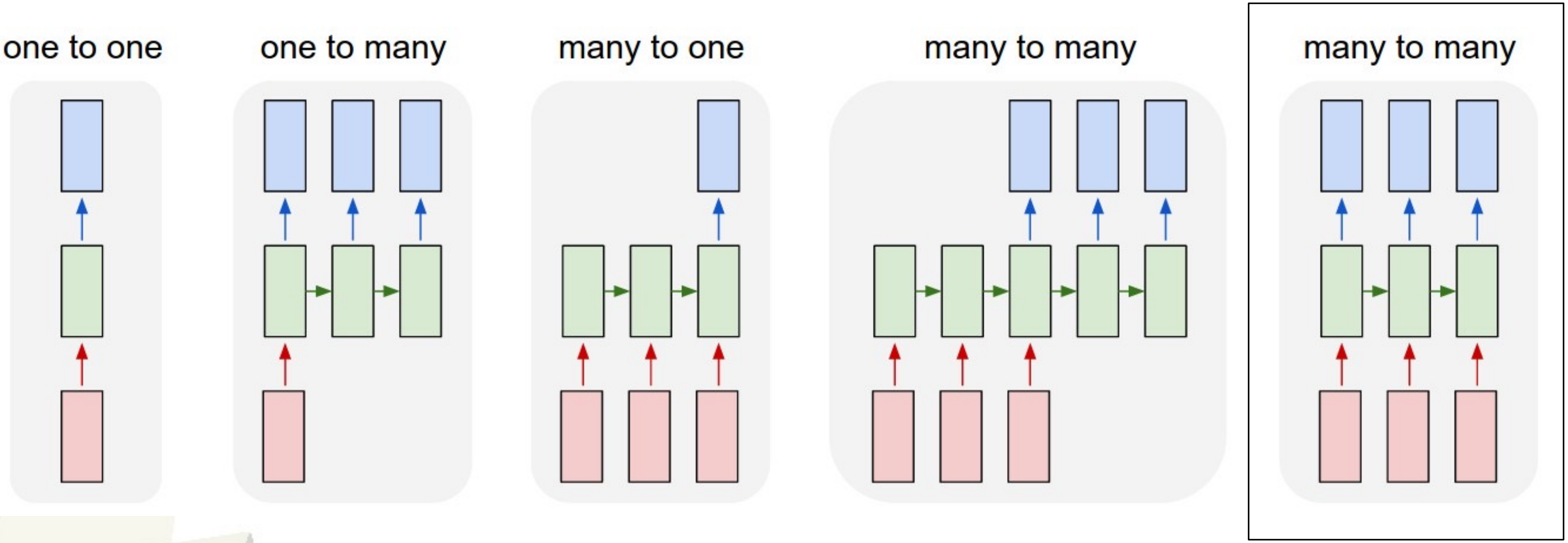
Now Lets Have Some Fun!

Recurrent Neural Network Configurations

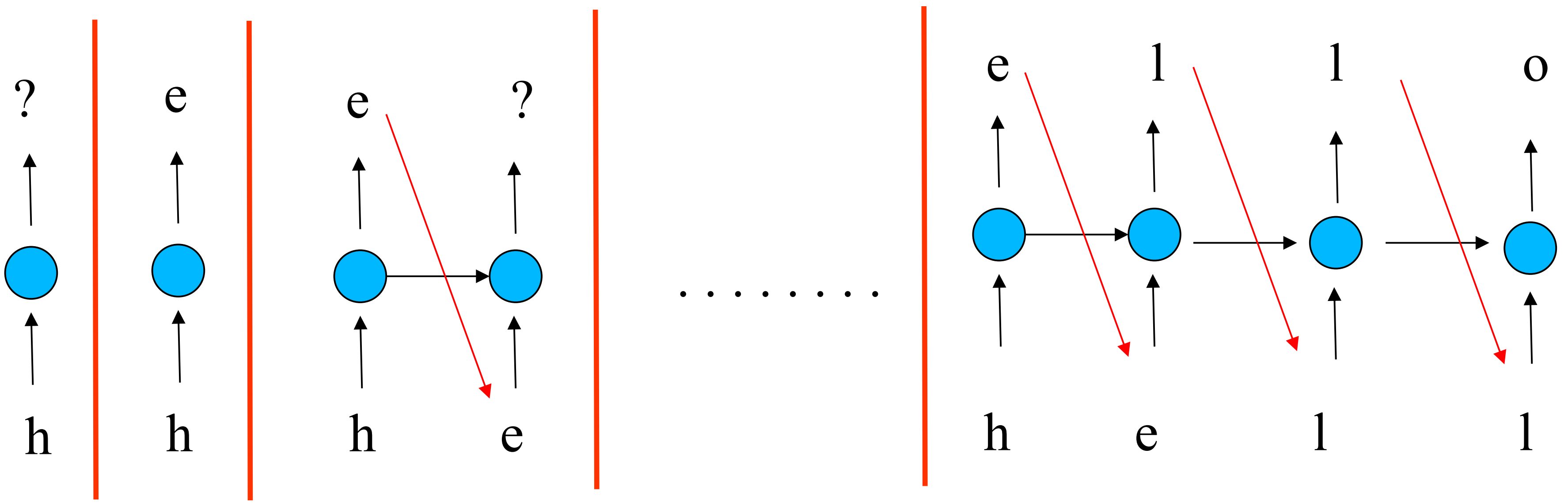


Like what regular neural networks can do

We will start with many to many



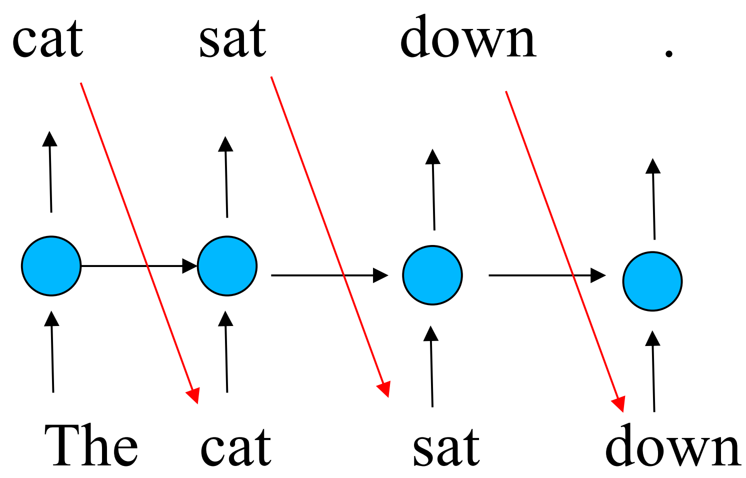
Many to Many (Generative Model)



- The RNN can be trained to read lots and lots of text (e.g. Wikipedia) one character at a time
- Once trained the RNN can be given a few start characters and asked to generate more text.
- For example below is what an RNN generated after being trained on Shakespeare's works

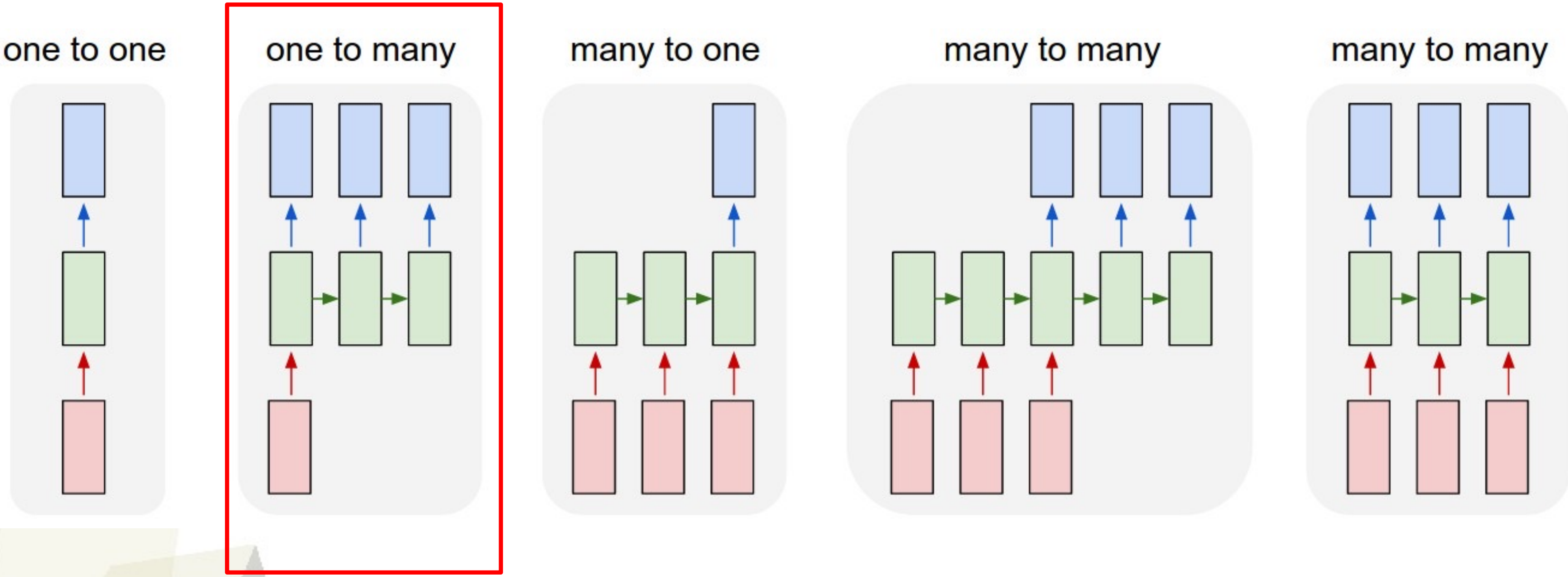
```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

Many to Many (Generative Model)



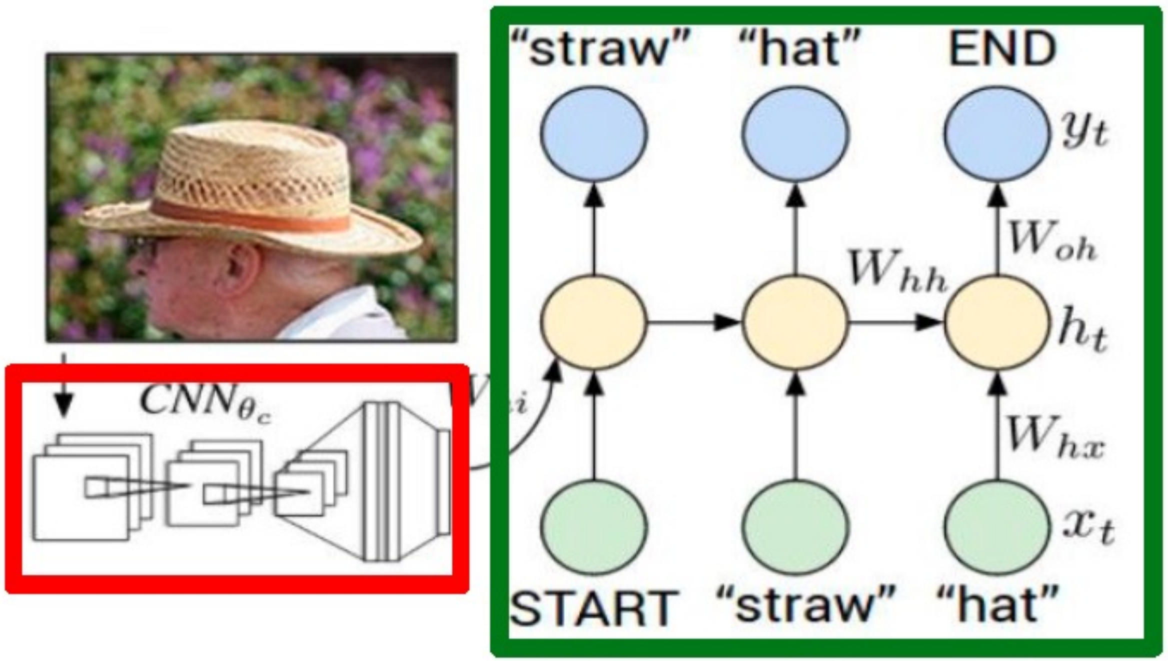
- Word at a time input instead of character at a time input

Recurrent Neural Network Configurations



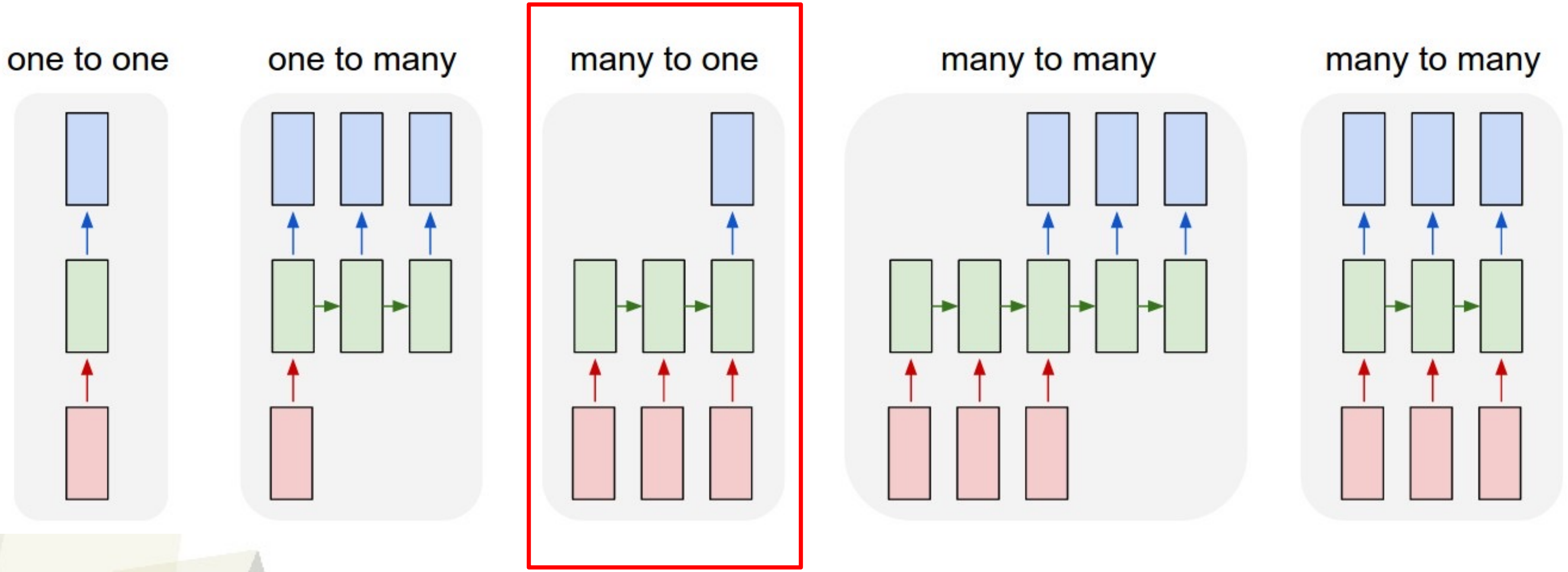
One to Many (Image Caption Generation)

Recurrent Neural Network

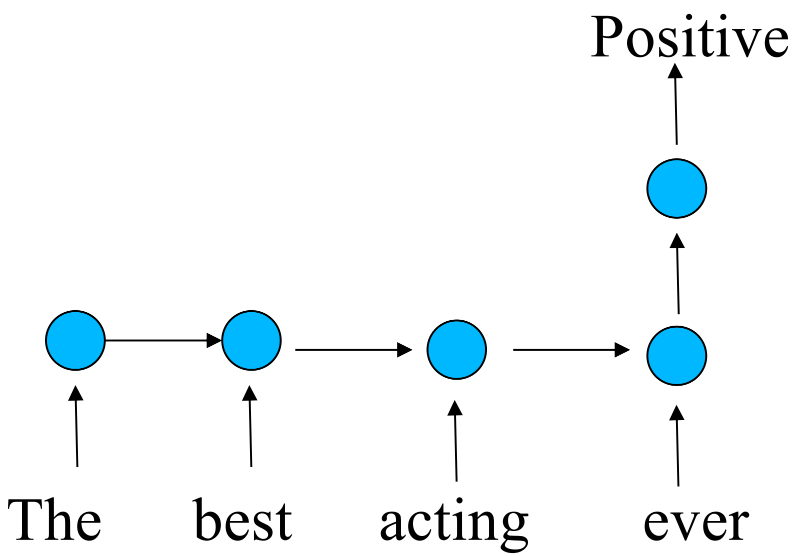


Convolutional Neural Network

Recurrent Neural Network Configurations

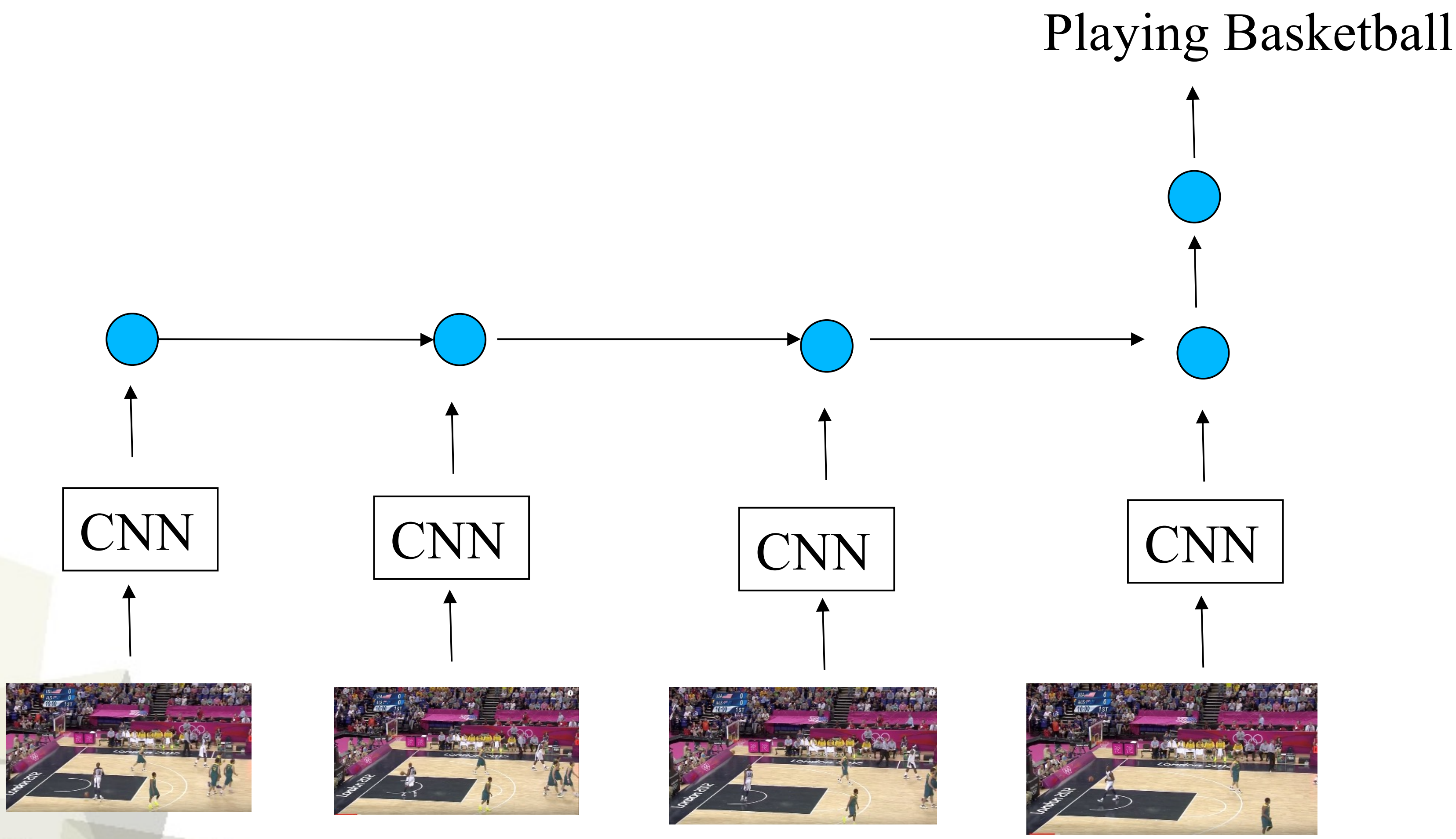


Many to One



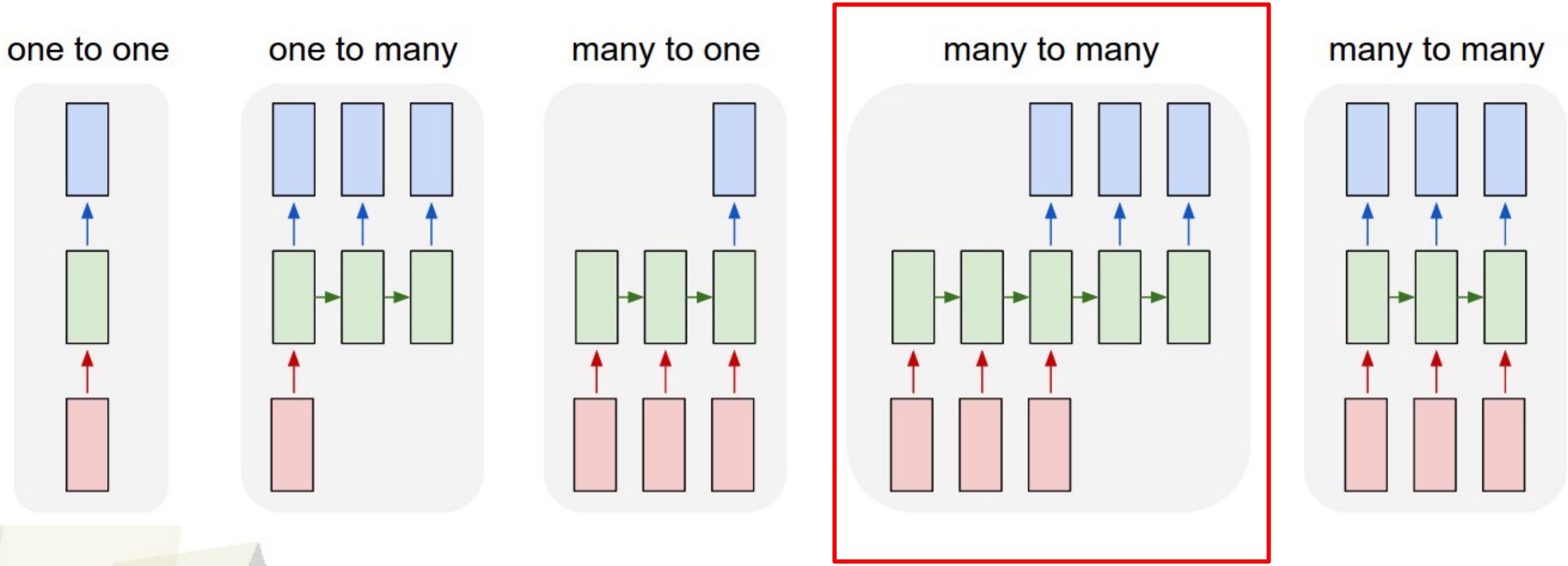
- Example use
  - Text classification

Many to One



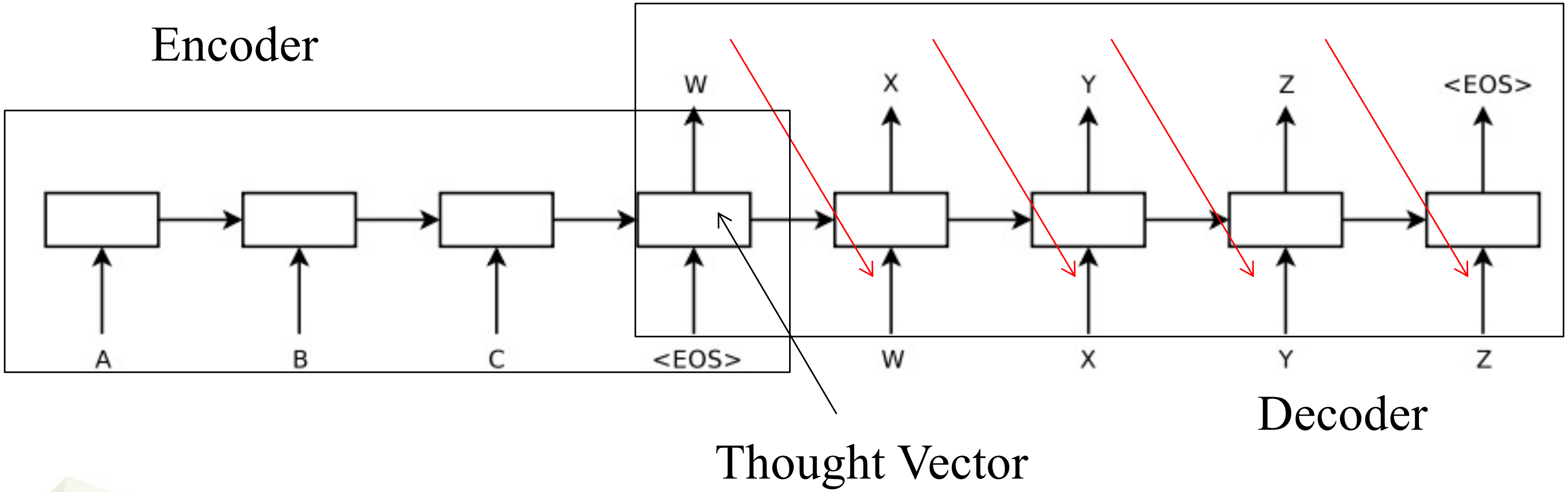
- Example use
  - Video classification

Recurrent Neural Network Configurations



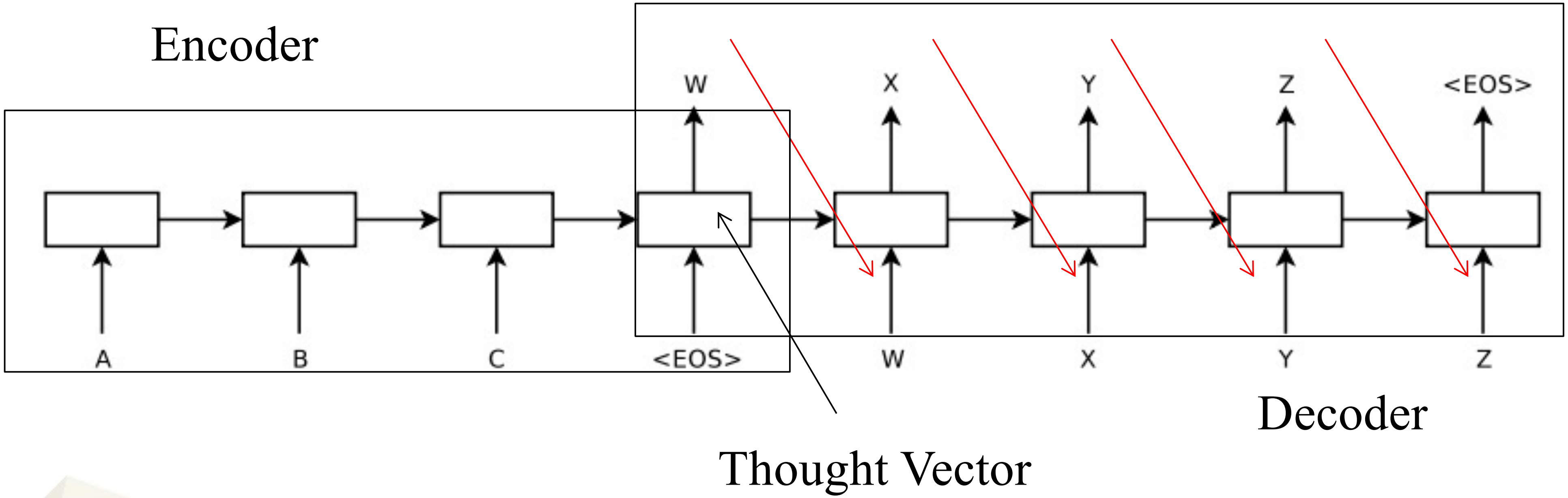
Sequence to Sequence Transformations

Many to Many (Sequence to Sequence Learning)



- Ilya Sutskever, et. al., NIPS 2014
- Applications
  - Language translation
    - English -> French
  - Question answering
  - Predicting the output of a program!

Many to Many (Sequence to Sequence Learning)



- Uses two different LSTM RNNs
  - One encoder
  - One decoder
  - Using two different LSTM means we get more parameters at low cost
  - Can also be used to train on multiple languages at the same time.
- The Thought Vector encodes the concept that the encoder outputs.
- End of sequence detected when <EOS> is found
- Feed output of decoder back into itself as input until <EOS> is outputted

Question Answering via Memory Networks

John dropped the milk.
John took the milk there.
Sandra went to the bathroom.
John moved to the hallway.
Mary went to the bedroom.
Where is the milk ? Hallway

- End-To-End Memory Networks, Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, Rob Fergus

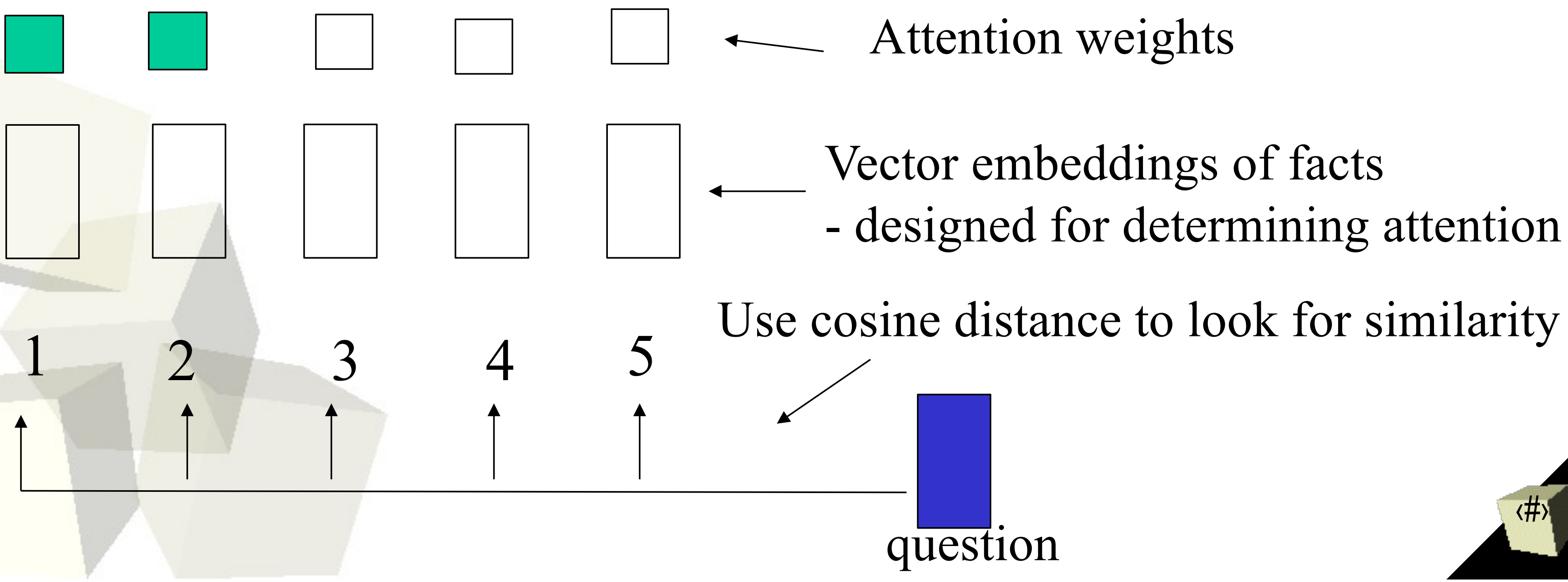
Memory Networks

facts

1. John dropped the milk.
2. John took the milk there.
3. Sandra went to the bathroom.
4. John moved to the hallway.
5. Mary went to the bedroom.

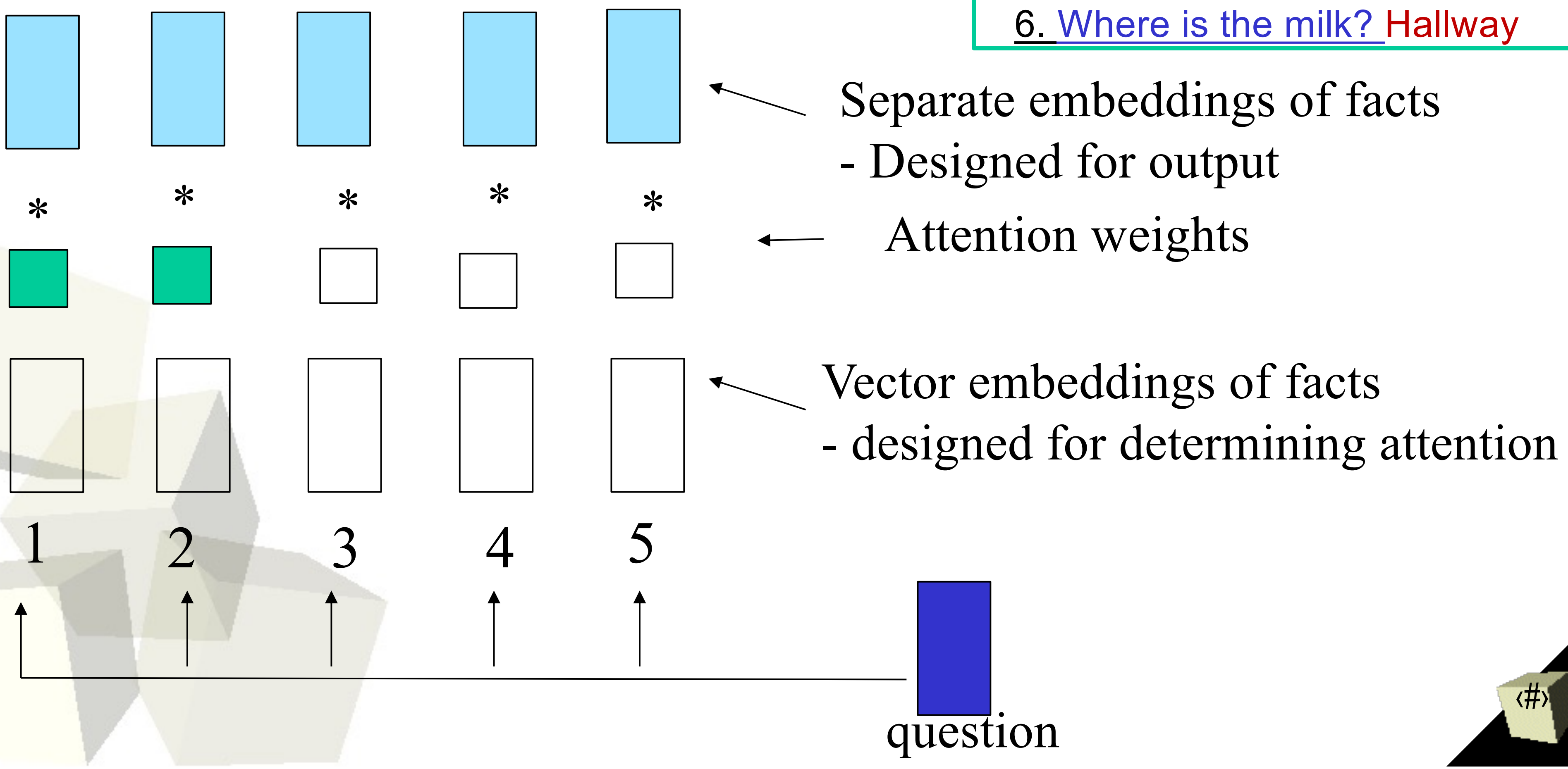
question

6. Where is the milk ?Hallway

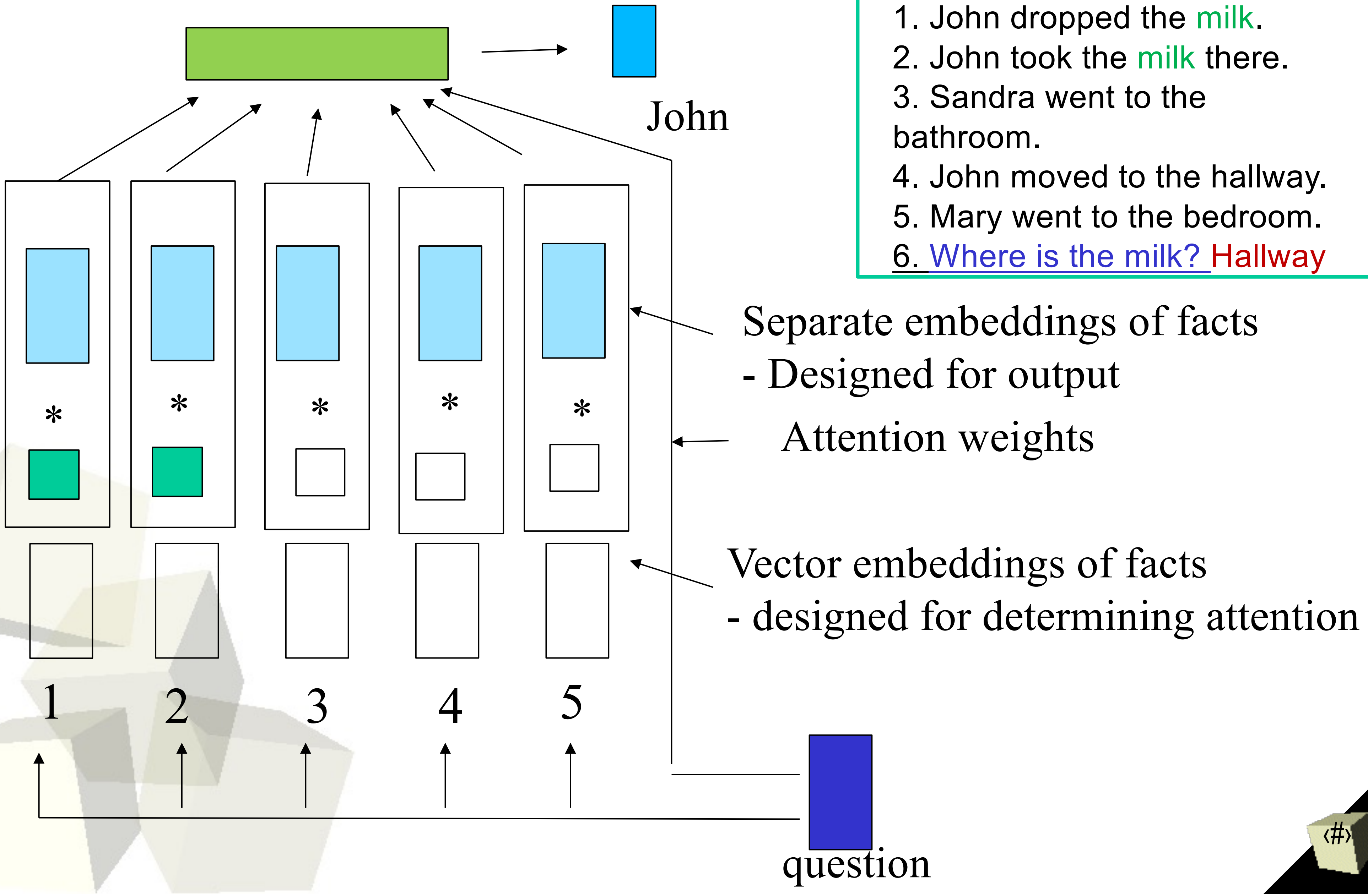


Memory Networks

1. John dropped the milk.
2. John took the milk there.
3. Sandra went to the bathroom.
4. John moved to the hallway.
5. Mary went to the bedroom.
6. Where is the milk? Hallway

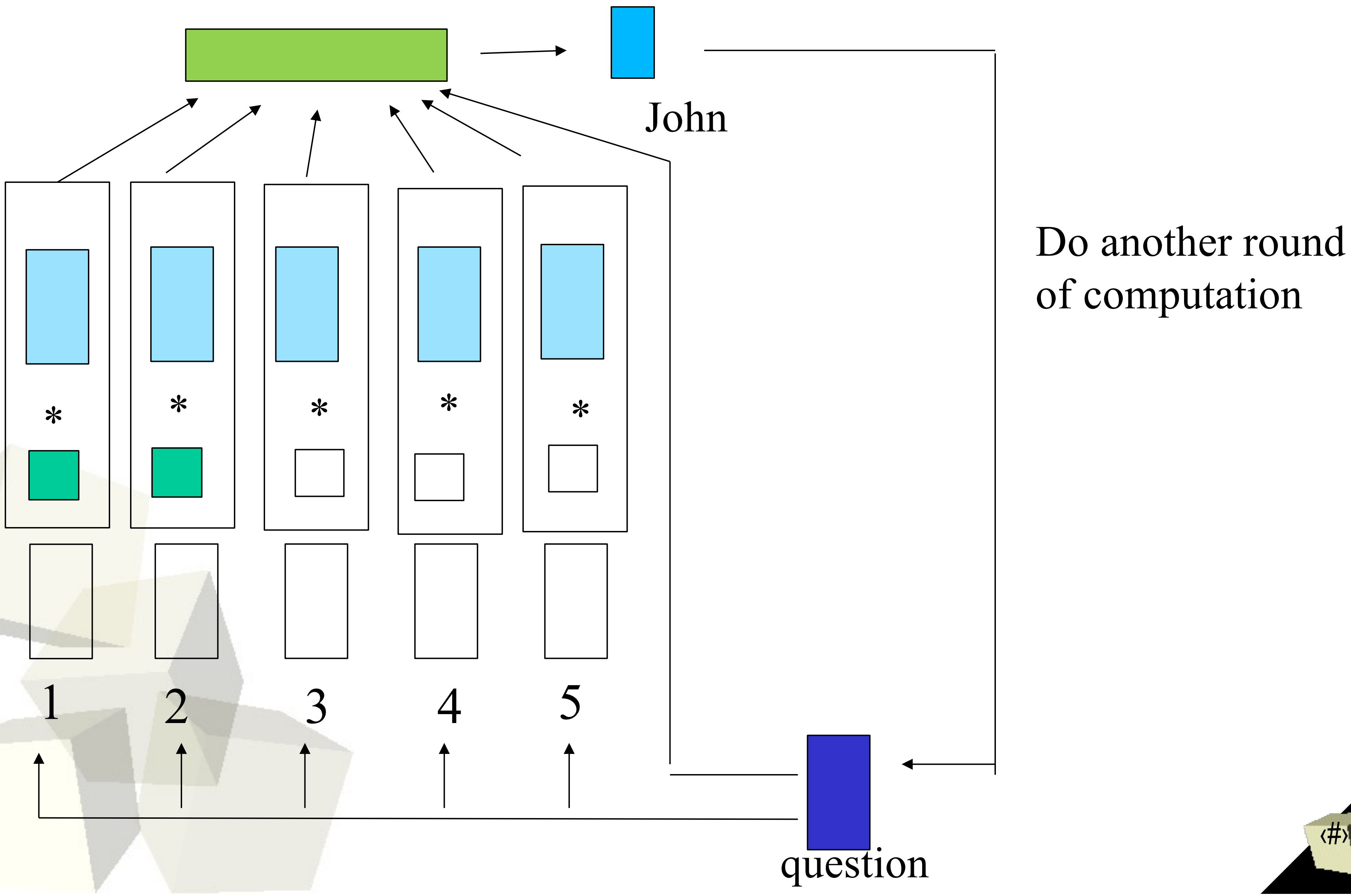


Memory Networks



1. John dropped the milk.
2. John took the milk there.
3. Sandra went to the bathroom.
4. John moved to the hallway.
5. Mary went to the bedroom.
6. Where is the milk? Hallway

Memory Networks



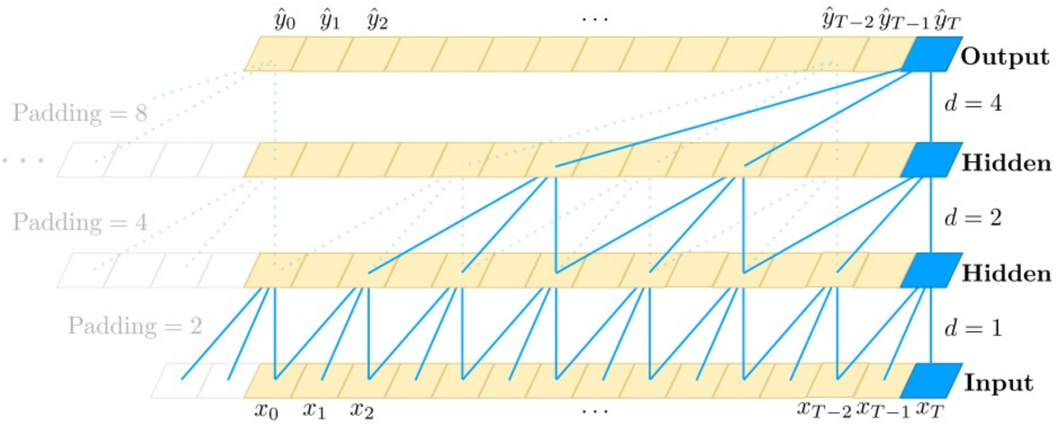
The Benefits of Memory Networks

- Unlike RNNs separates memory and computation
- Unlike RNNs store each fact separately
  - Do not need to compress everything into a single vector
- Does multiple steps of computation

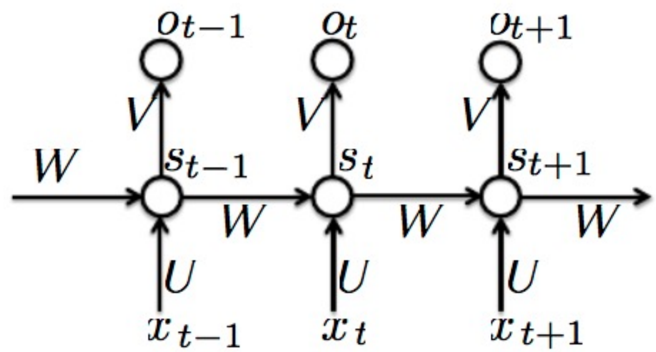
Use convolutional neural networks to model sequences

Convolutions Versus RNNs

Dilated causal convolutions



Recurrent Neural Networks



- In theory RNNs can condition on any length from the past
  - In practice can only condition at most 200 time steps in the past
    - Vanishing and exploding gradients
- CNNs can condition a finite time steps in the past
  - Using dilations and skip connections can condition on much longer history in the past than RNNs

Causal Convolutions

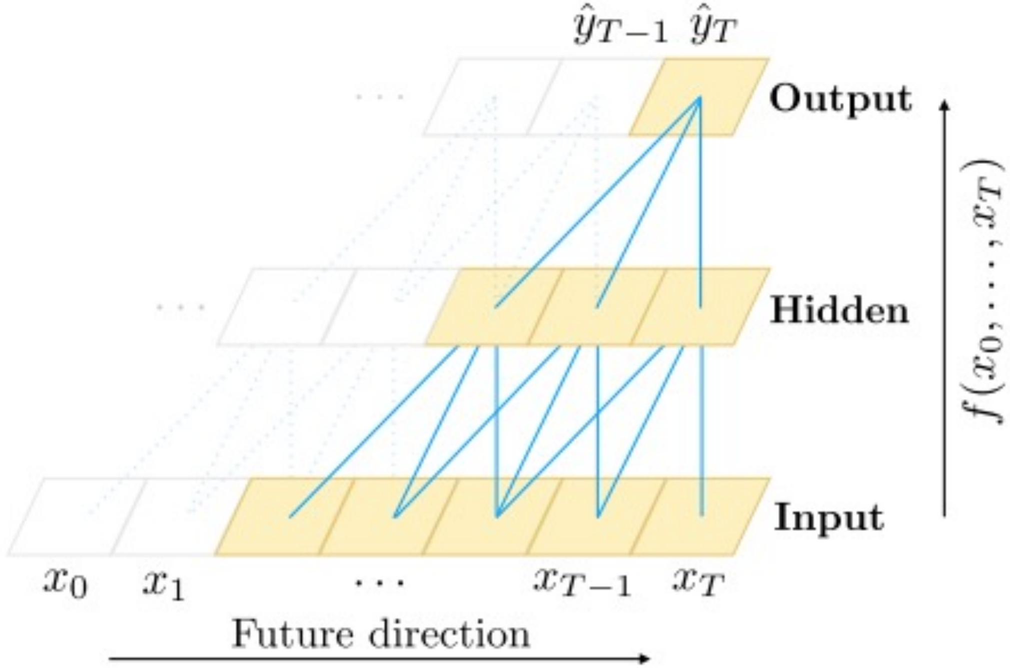
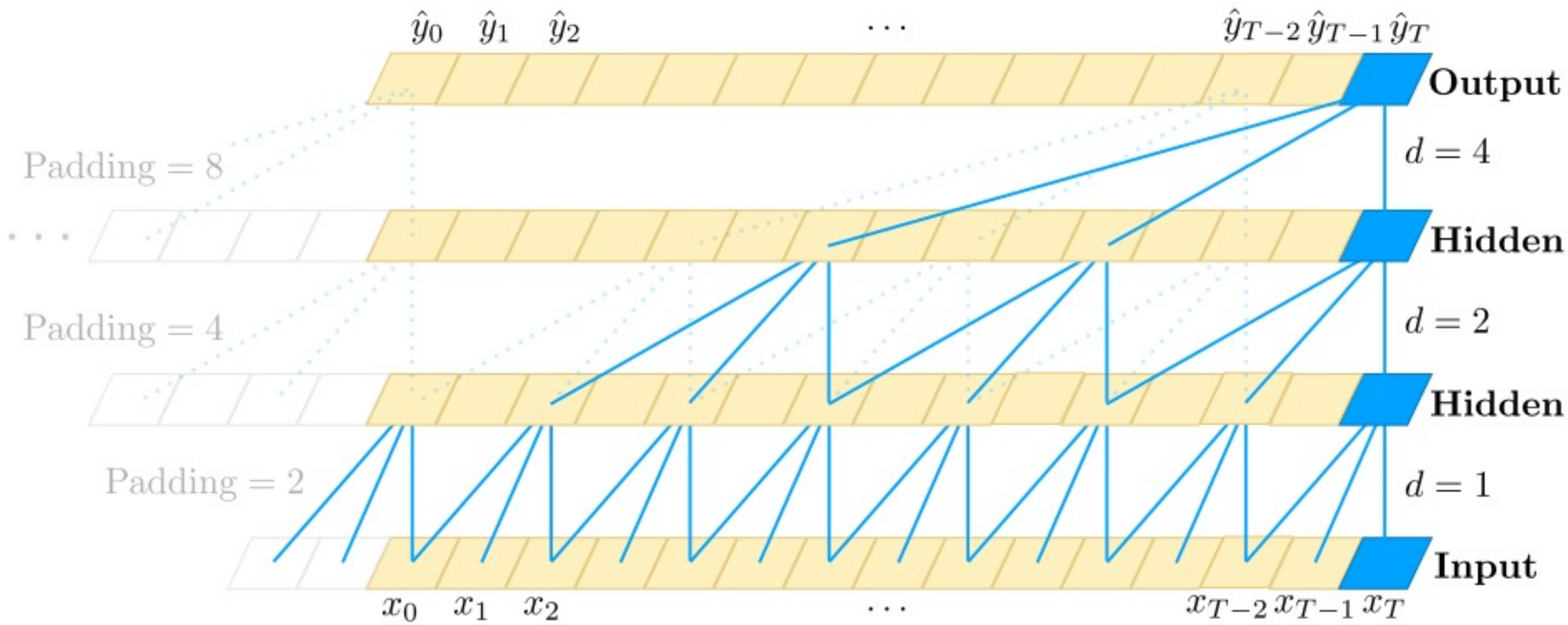


Figure 1: A simple causal convolution with filter size 3.

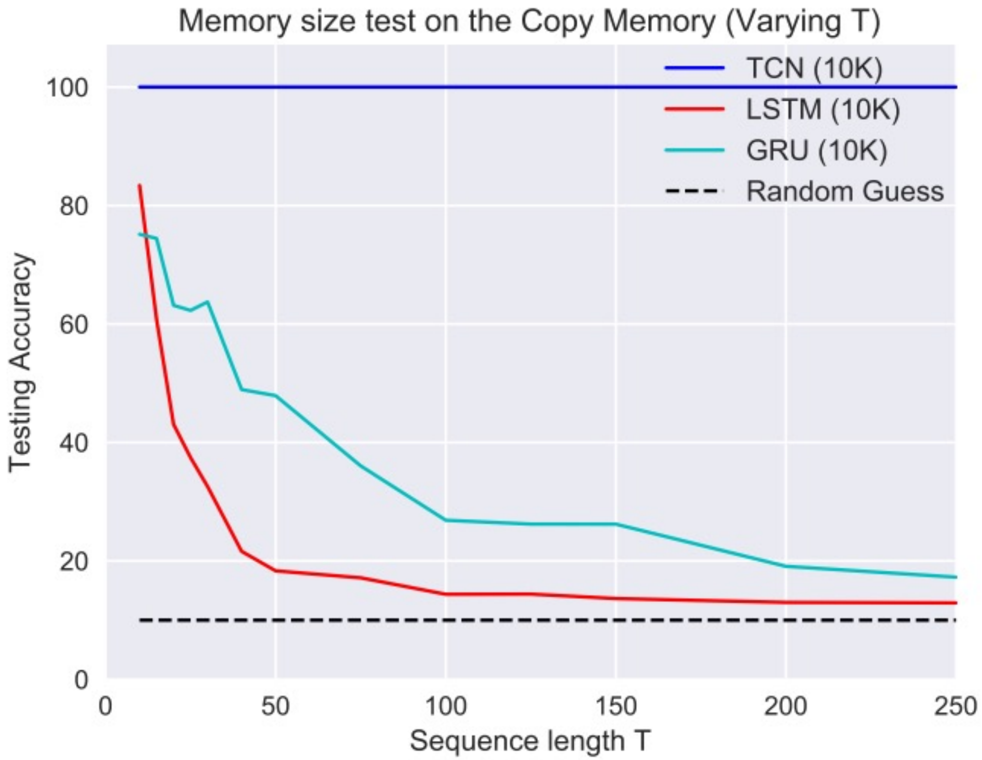
- The prediction for output y_T only depends on the future
- During training a mask is used to zero out the input from the future

Dilated causal convolutions



- A dilated causal convolution with dilation factors d = 1, 2, 4 and filter size k = 3. The receptive field is able to cover all values from the input sequence
- Increase d exponentially with depth
  - Ensures there is some filter that hits every input within effective history

Memory Size Test

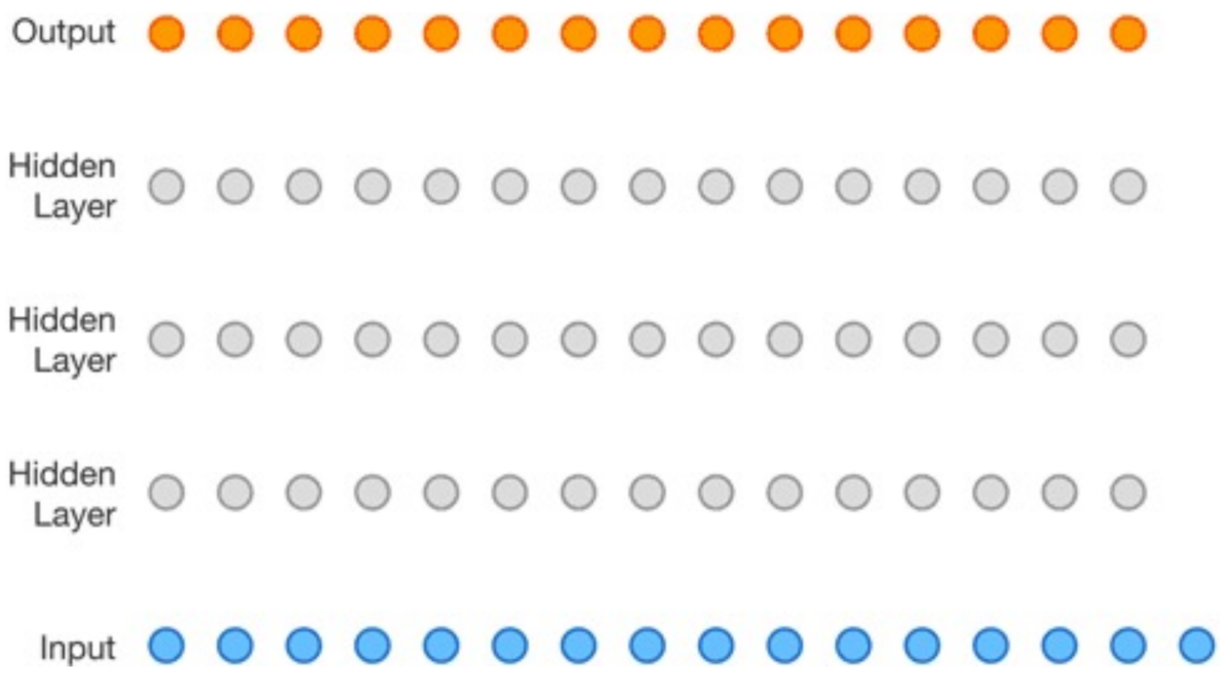


Temporal Convolutional Network (TCN)

- Memory copy test
  - Input
    - First 10 digits chosen randomly
    - Variable number of zeros (depending on length of T)
    - 11 copies of the number 9 at the end.
  - Output
    - Same length of zeros
    - Last 10 values are the same as the first 10 from the input
- This shows that Temporal Convolutional Network (TCN) has much better memory than the RNN variants

https://openreview.net/forum?id=rk8wKk-R-

WaveNet



- Uses dilated causal convolutions to generate audio
  - Turn text to speech audio
  - Generate music
- Produces much better audio compared to existing methods
- Used in Google Assistant

https://deepmind.com/blog/wavenet-generative-model-raw-audio/

Summary

- In this lecture we introduced the types of NLP problems you can solve using neural networks.
- We also introduced one of the main ways to solve these problems which is to use recurrent neural networks.
- In the next lecture we will introduce the state-of-the-art method solving NLP problems using deep learning, namely the use of transformer networks.
- In the next lecture we will also learn how to program deep learning algorithms for NLP.