Course series: Deep Learning for Machine Translation

Neural Network Language Models

Lecture # 6

Hassan Sajjad and Fahim Dalvi
Qatar Computing Research Institute, HBKU

Recap: Language Model

Lecture 2: *Language model defines "how probable a sentence is"*

John is driving a car vs. John is driving a cat

In other words, what is the probability to predict **cat** or **car** given the context "John is driving a"

Recap: Ngram Language Model

Probability of a sentence is given by the product of the probability over the sequence of words

Bigram probability of a sentence:

$$p(s) = \prod_{k-1}^{n} p(w_k | w_{k-N-1}^{k-1})$$

Issues with Ngram LM

Space inefficiency

- a large number of ngrams
- large model size

Data sparsity

- can never have enough counts of all ngrams in the data
- can never see all possible ngrams

Discrete relationship between similar concepts

- *dog* is sitting vs. *cat* is sitting

Language Model
As a Classification Problem

Q: Can we see language modeling as a classification problem?

Language Model
As a Classification Problem

Q: Can we see language modeling as a classification problem?

A: Yes! We are just predicting which word ("class") is coming next.

Language Model
As a Classification Problem

Q: Can we see language modeling as a classification problem?

A: Yes! We are just predicting which word ("class") is coming next.

$$p(\texttt{<s>}\ \text{Dan likes ham}\ \texttt{</s>}) =$$

Language Model

As a Classification Problem

Q: Can we see language modeling as a classification problem?

A: Yes! We are just predicting which word ("class") is coming next.

$$p(\texttt{<s>}\ \text{Dan likes ham}\ \texttt{</s>}) = \boxed{p(\texttt{Dan}\,|\,\texttt{<s>})}$$

Predict **Dan** given **<s>**

Language Model
As a Classification Problem

Q: Can we see language modeling as a classification problem?

A: Yes! We are just predicting which word ("class") is coming next.

$$p(\texttt{<s>}\ \text{Dan likes ham}\ \texttt{</s>}) = \boxed{p(\texttt{Dan}\,|\,\texttt{<s>})}$$

Predict **Dan** given **<s>**

What is the probability of **Dan** given **<s>**?

Language Model
As a Classification Problem

Q: Can we see language modeling as a classification problem?

A: Yes! We are just predicting which word ("class") is coming next.

$$p(\texttt{<s>}\ \text{Dan likes ham}\ \texttt{</s>}) = p(\texttt{Dan|<s>}) \cdot \boxed{p(\texttt{likes|Dan})}$$

Predict **likes** given **Dan**

Language Model

As a Classification Problem

Q: Can we see language modeling as a classification problem?

A: Yes! We are just predicting which word ("class") is coming next.

$$p(\texttt{<s>}\ \text{Dan likes ham}\ \texttt{</s>}) = p(\texttt{Dan}\,|\,\texttt{<s>})$$
$$\cdot\, p(\texttt{likes}\,|\,\texttt{Dan})$$
$$\cdot\, \boxed{p(\texttt{ham}\,|\,\texttt{likes})}$$

Predict **ham** given **likes**

Language Model
As a Classification Problem

Q: Can we see language modeling as a classification problem?

A: Yes! We are just predicting which word ("class") is coming next.

$$p(\texttt{<s>}\ \text{Dan likes ham}\ \texttt{</s>}) = p(\texttt{Dan}\,|\,\texttt{<s>}) \cdot p(\texttt{likes}\,|\,\texttt{Dan}) \cdot p(\texttt{ham}\,|\,\texttt{likes}) \cdot \boxed{p(\texttt{</s>}\,|\,\texttt{ham})}$$

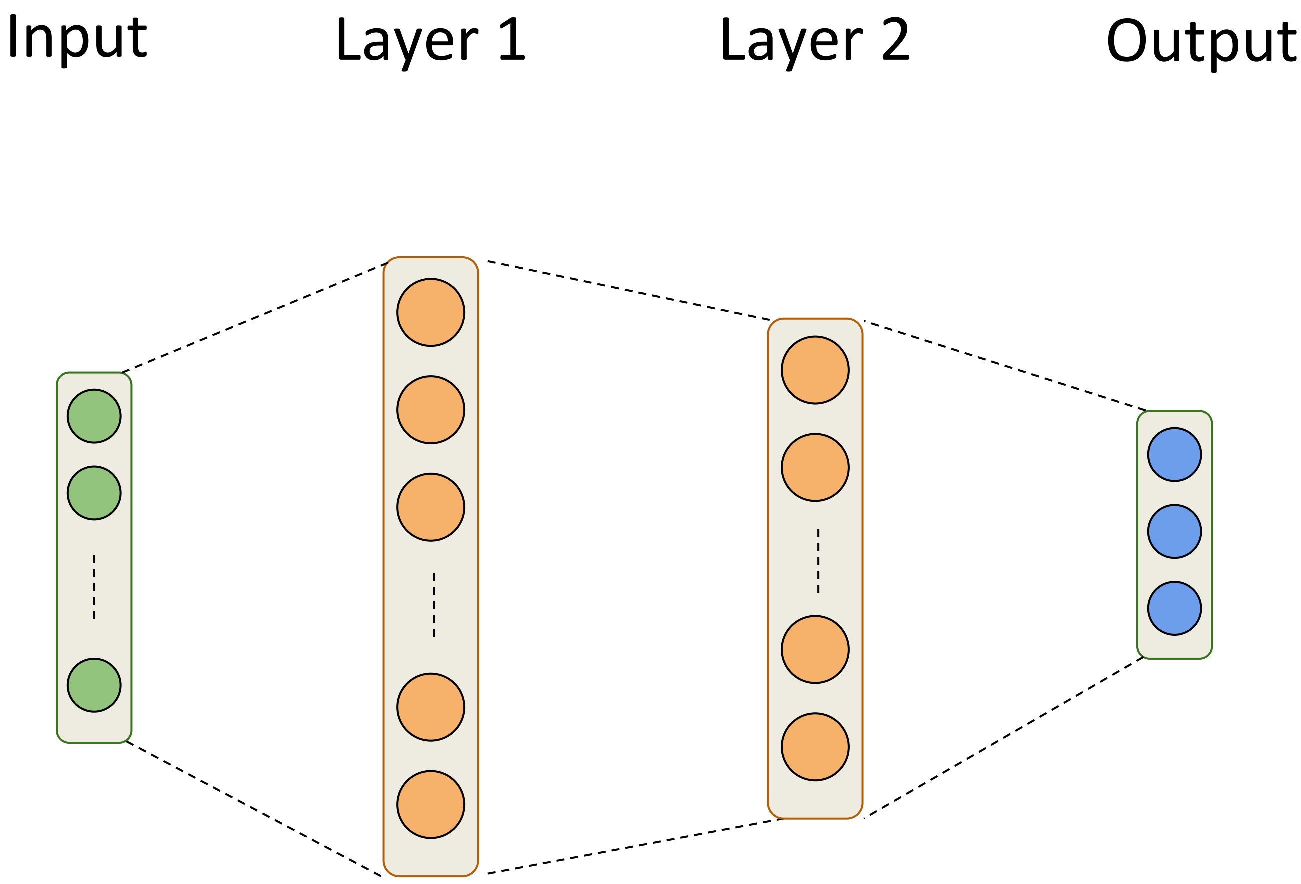
Language Model
As a Classification Problem

Words represent classes that we want to predict!

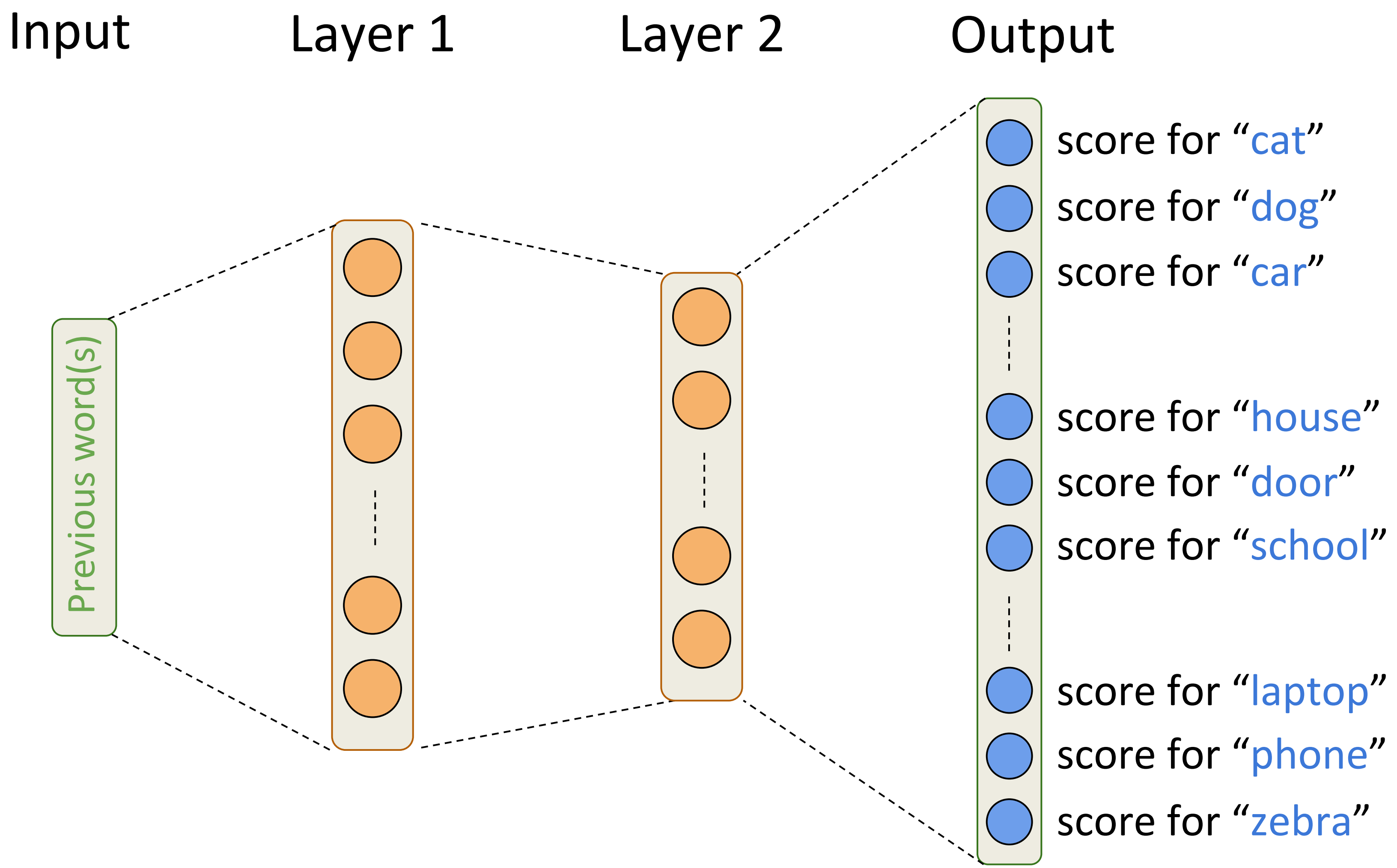
Input to the classifier: previous words i.e. context

Output: probability distribution over all possible words, i.e. our vocabulary

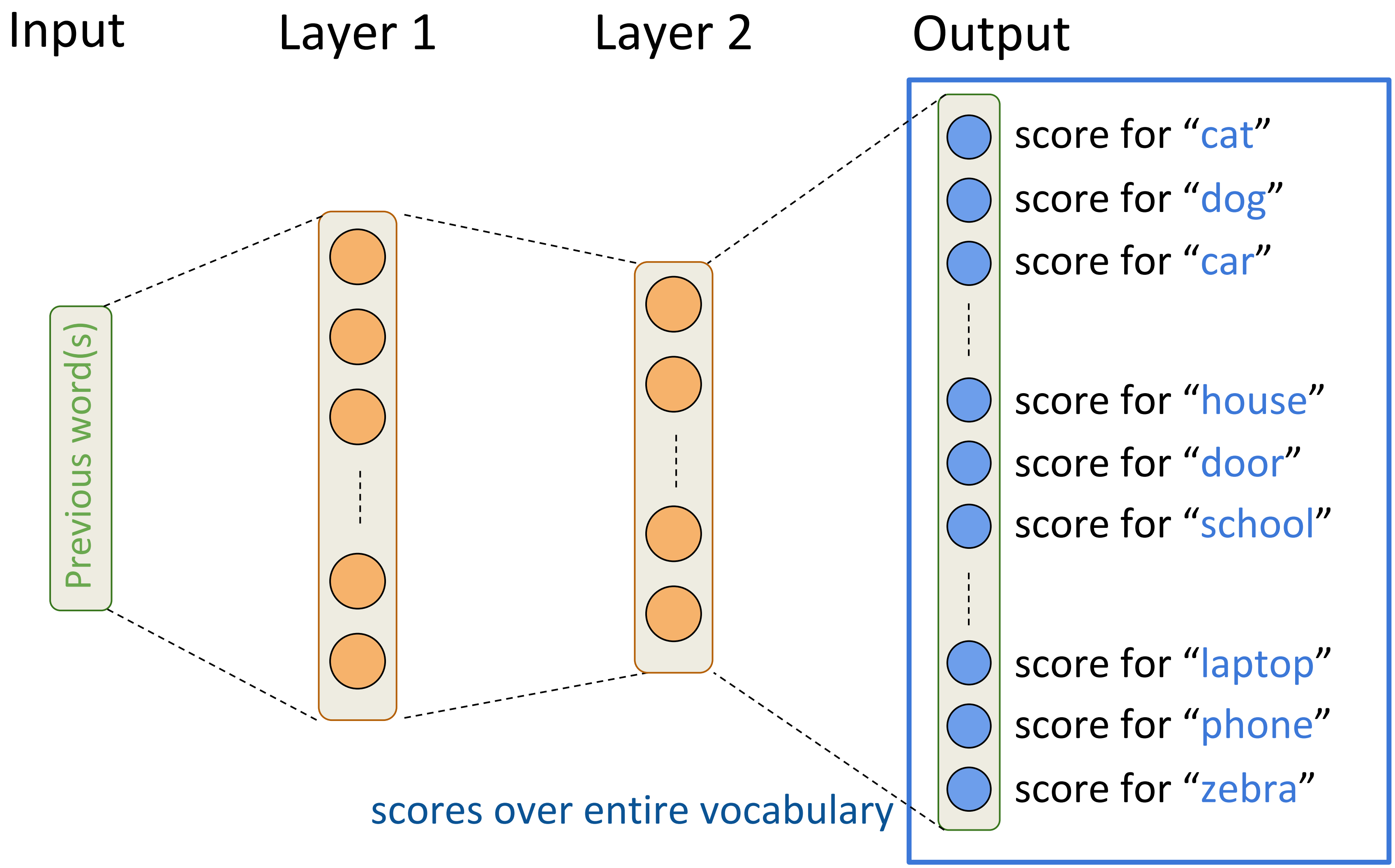
Neural Network Language Model



Neural Network Language Model



Neural Network Language Model



Input Representation

Input

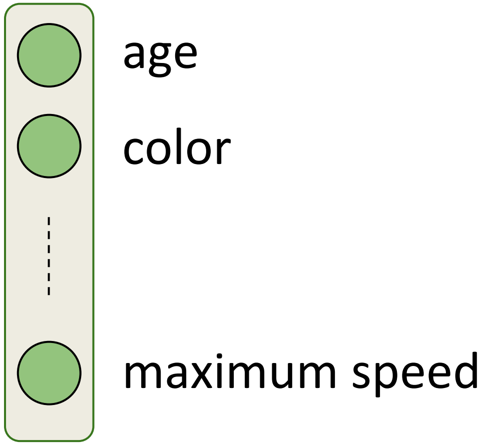
Previous word(s)

Previously we've used a vector as input, where each element of the vector represented some "feature" of the input

Input Representation

Input

Previously we've used a vector as input, where each element of the vector represented some "feature" of the input

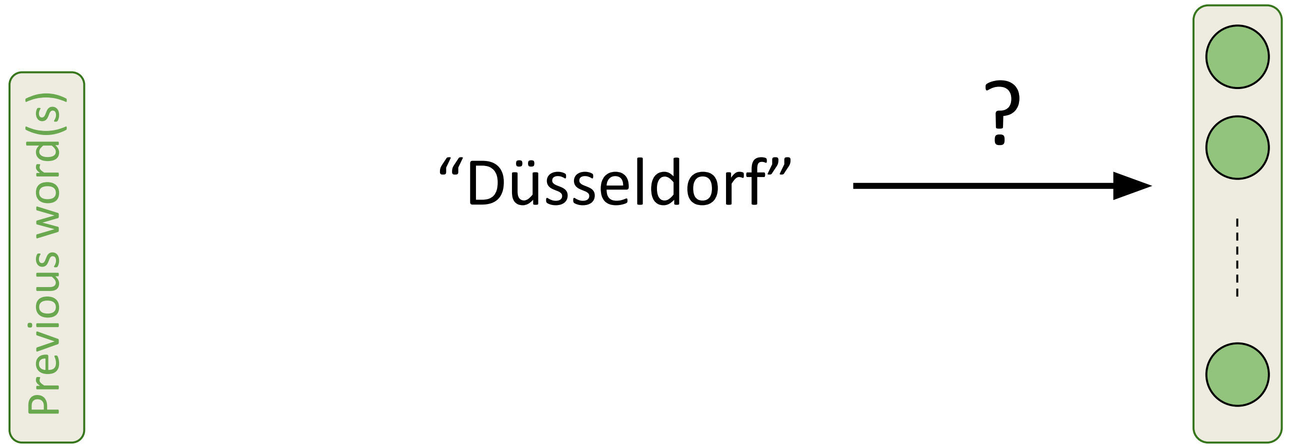


Car

Input Representation

Input

Can we represent a word as a feature vector?



One Hot Vector Representation

- Every word can be represented as a ***one hot vector***

One Hot Vector Representation

- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000

One Hot Vector Representation

- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000
- Assign each word a unique index:

```
Düsseldorf: 1
cat: 2
house: 3
car: 4
    ⋮
apple: 10,000
```

One Hot Vector Representation

- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000
- Assign each word a unique index:

```
Düsseldorf: 1
cat: 2
house: 3
car: 4
    ⋮
apple: 10,000
```

Dictionary

$$cat = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

One-hot representation

One Hot Vector Representation

- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000
- Assign each word a unique index:

Only index that represents the input word will be one

$$cat = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

One-hot representation

One Hot Vector Representation

- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000
- Assign each word a unique index:

Only index that represents the input word will be one

$$cat = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad car = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

One-hot representation

One Hot Vector Representation

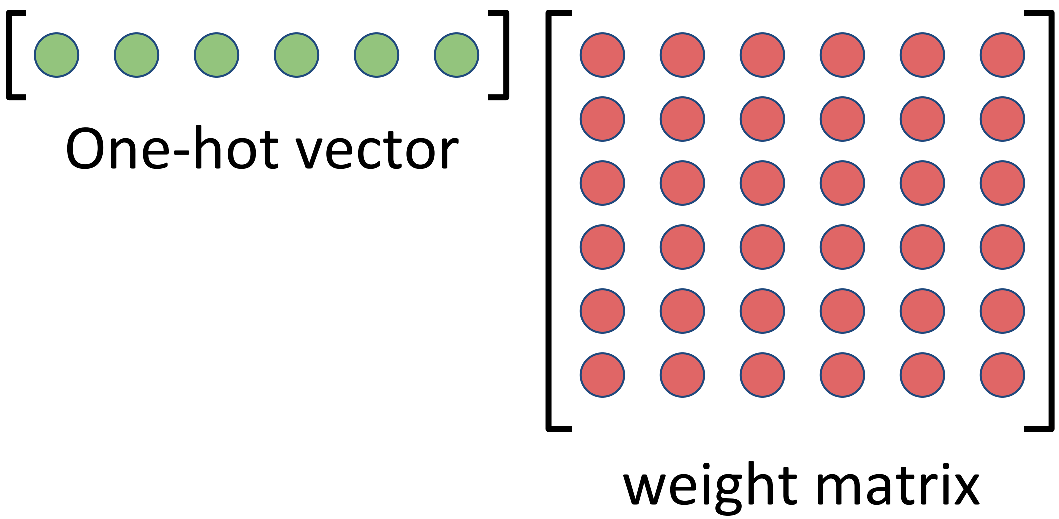
- Every word can be represented as a ***one hot vector***
- Suppose the total number of unique words in the corpus is 10,000
- Assign each word a unique index:

Vector size will be the size of the vocabulary, i.e. 10,000 in this case

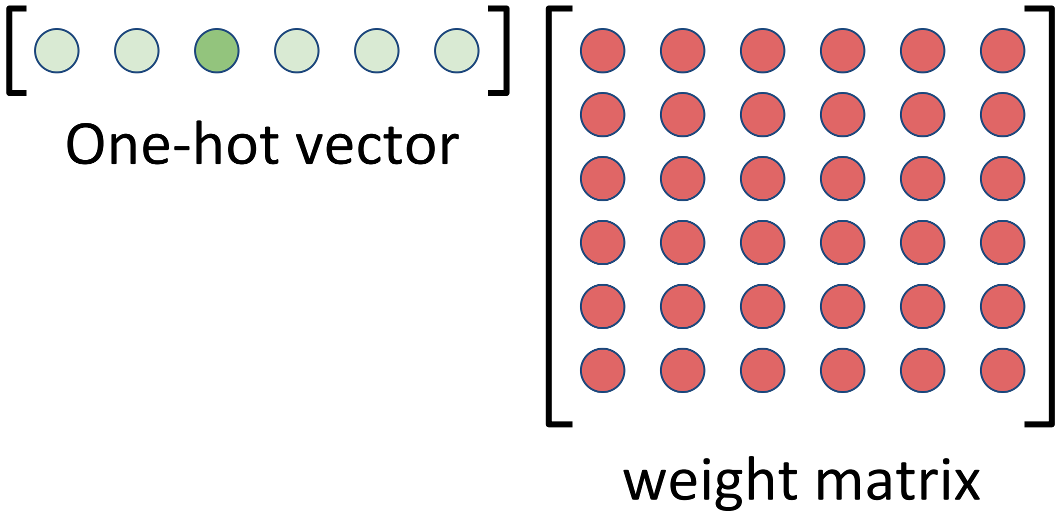
$$cat = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad car = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

One-hot representation

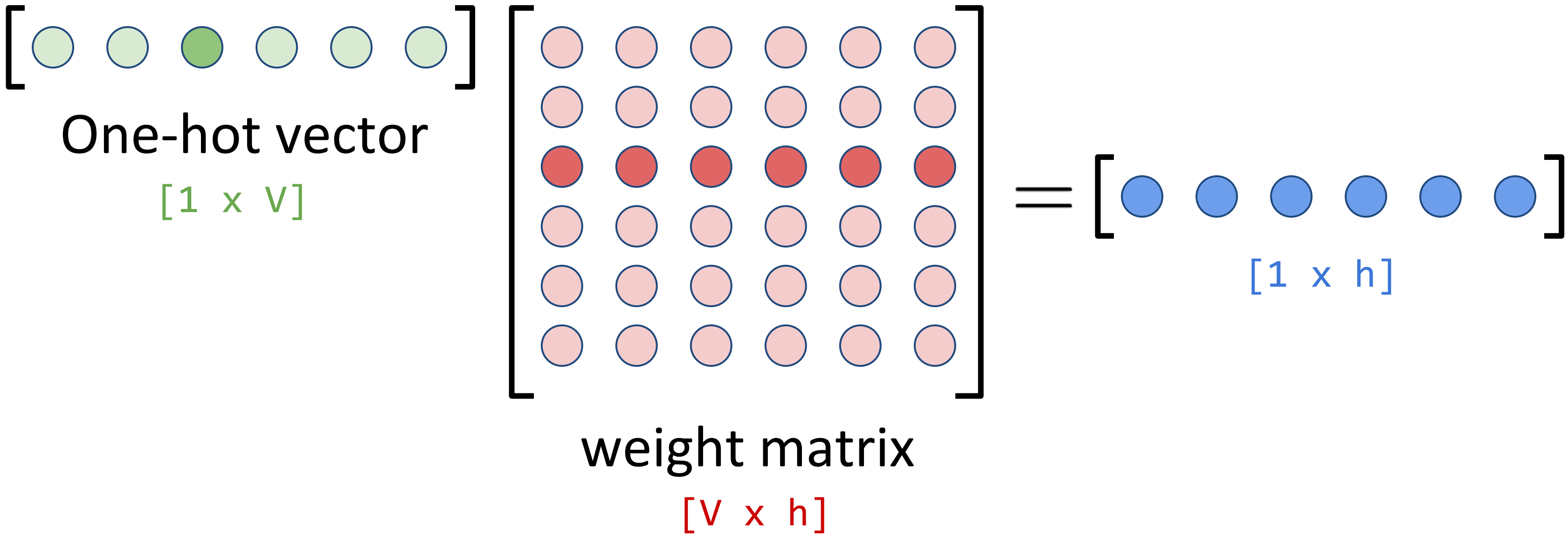
One Hot Vector Representation



One Hot Vector Representation



One Hot Vector Representation



One-hot vector will “turn on” one row of weights

Higher ngram Vector Representation

- What about representing multiple words?

Bag of words approach

Higher ngram Vector Representation

- What about representing multiple words?

Bag of words approach

Bigram: indices of the *two previous words* are 1 in the vector

$$\text{a car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \qquad \text{a cat} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

Higher ngram Vector Representation

- What about representing multiple words?

Bag of words approach

Bigram: indices of the *two previous words* are 1 in the vector

$$\text{a car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \boxed{1} \\ 0 \\ 0 \\ \vdots \\ \boxed{1} \end{bmatrix} \qquad \text{a cat} = \begin{bmatrix} 0 \\ \boxed{1} \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ \boxed{1} \end{bmatrix}$$

Higher ngram Vector Representation

- What about representing multiple words?

Bag of words approach

Trigram: indices of the *three previous words* are 1 in the vector

$$\text{a fast car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \boxed{1} \\ 0 \\ \boxed{1} \\ \vdots \\ \boxed{1} \end{bmatrix} \quad \text{a fast cat} = \begin{bmatrix} 0 \\ \boxed{1} \\ 0 \\ 0 \\ 0 \\ \boxed{1} \\ \vdots \\ \boxed{1} \end{bmatrix}$$

Higher ngram Vector Representation

- What about representing multiple words?

Context-aware approach

In the **bag of words** approach, order information is lost!

Higher ngram Vector Representation

- What about representing multiple words?

Context-aware approach

In the **bag of words** approach, order information is lost!

Solution: for *N* words, concatenate one-hot vectors for each of the words in the correct order

Higher ngram Vector Representation

Context-aware approach

$$
\text{a} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \quad
\text{car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad
\longrightarrow \quad
\text{a car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}
$$

Higher ngram Vector Representation

Context-aware approach

$$\text{a} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \quad \text{car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \longrightarrow \text{a car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

[V x 1] [V x 1] [2V x 1]

Higher ngram Vector Representation

Context-aware approach

input vector length has increased

$$\text{a} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \qquad \text{car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \longrightarrow \text{a car} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

[V x 1] [V x 1] [2V x 1]

Higher ngram Vector Representation

Context-aware approach

input vector length has increased

- order information is available for the training

Advantages

- long vectors in case of large context size
- number of parameters increases with context size

Disadvantages

Higher ngram Vector Representation

- Bag of words vs. context-aware approach?
  - Given the disadvantages of the context-aware approach, Bag of words is more commonly used
  - Works well in practice

Input Representation

Generally, the size of the vocabulary is very large

- Results in very large one-hot vectors!

Input Representation

Generally, the size of the vocabulary is very large

- Results in very large one-hot vectors!

Some tricks to reduce vocabulary size:

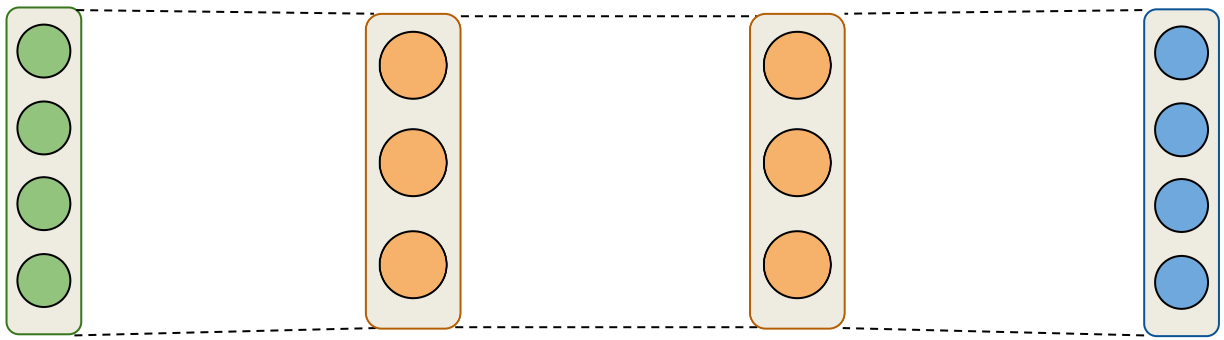
1) Take most frequent top words. For example, consider only 10,000 most frequent words and map the rest to a unique token `<UNK>`
2) Cluster words
   a) based on context
   b) based on linguistic properties

Neural Network Language Model

Let us look at a complete example:

Dataset: `{“how”, “you”, “hello”, “are”}`

Network Architecture: 2 hidden layers of size 3 each

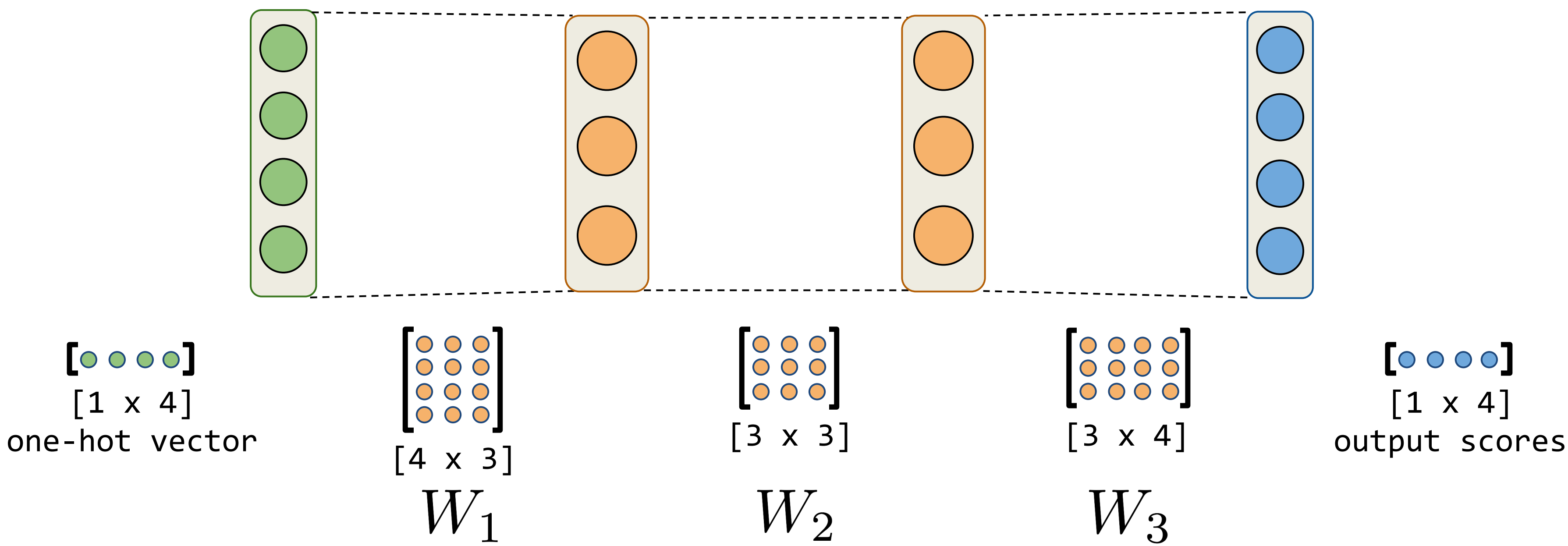


Neural Network Language Model

Let us look at a complete example:

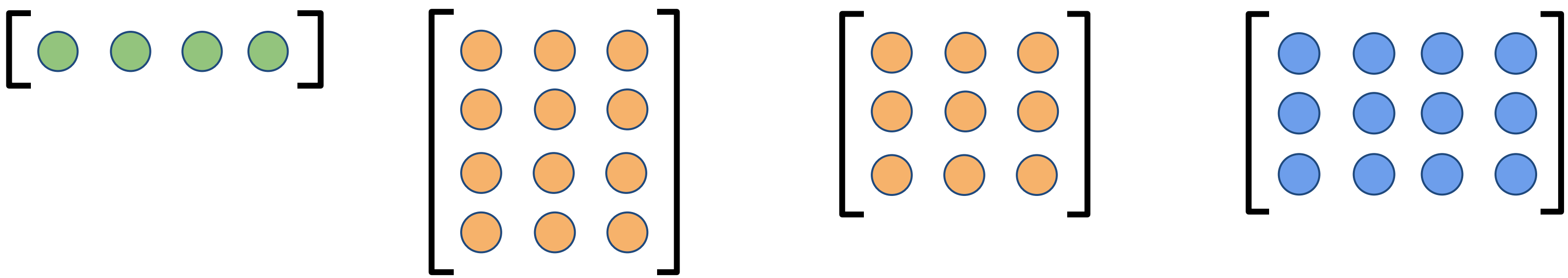
Dataset: {“how”, “you”, “hello”, “are”}

Network Architecture: 2 hidden layers of size 3 each



Neural Network Language Model

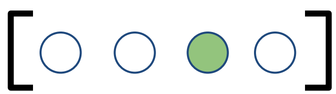
Dataset: {“how”, “you”, “hello”, “are”}



W_1 W_2 W_3

Neural Network Language Model

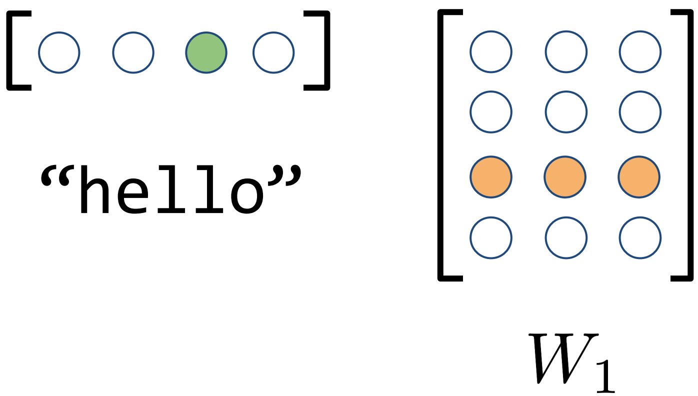
Dataset: {“how”, “you”, “hello”, “are”}



“hello”

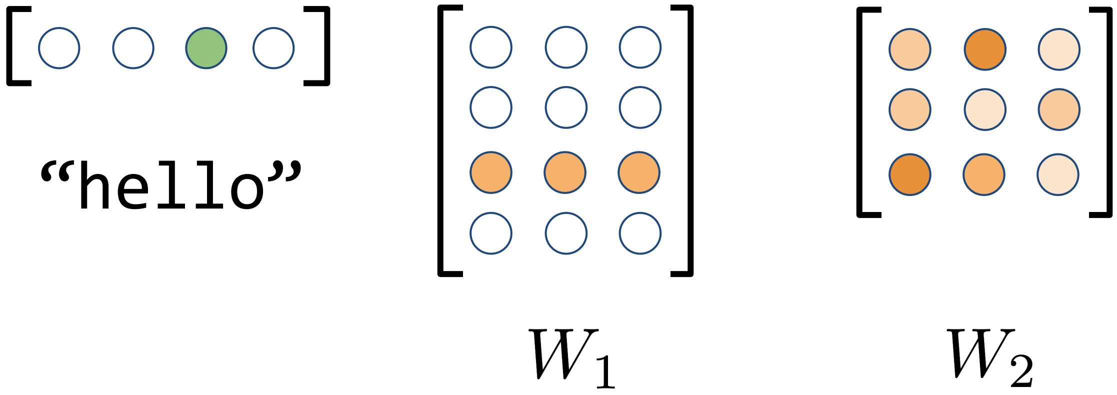
Neural Network Language Model

Dataset: {“how”, “you”, “hello”, “are”}



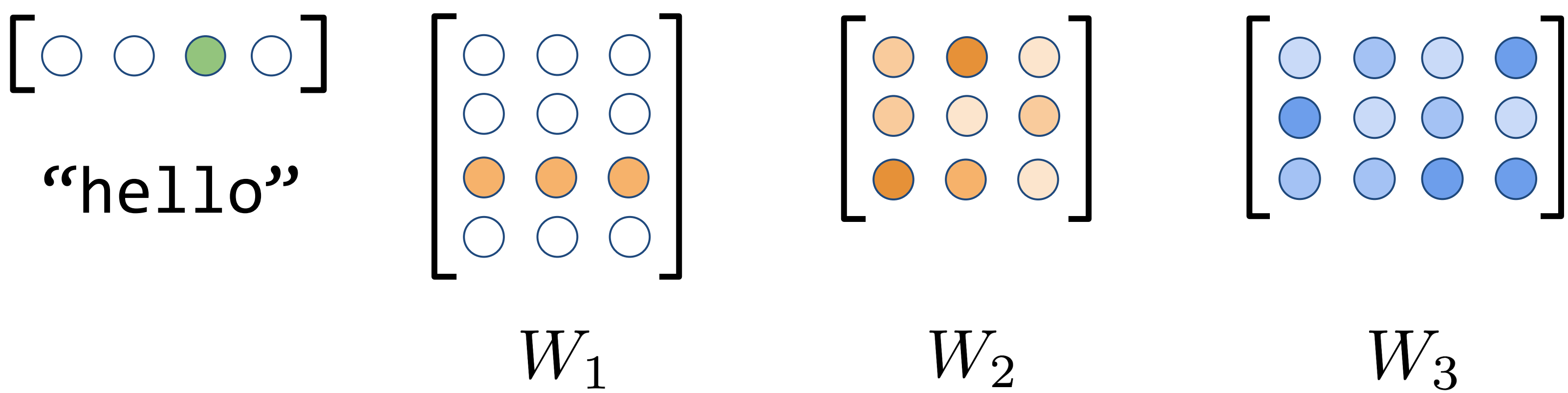
Neural Network Language Model

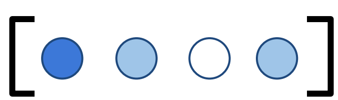
Dataset: {“how”, “you”, “hello”, “are”}



Neural Network Language Model

Dataset: {“how”, “you”, “hello”, “are”}

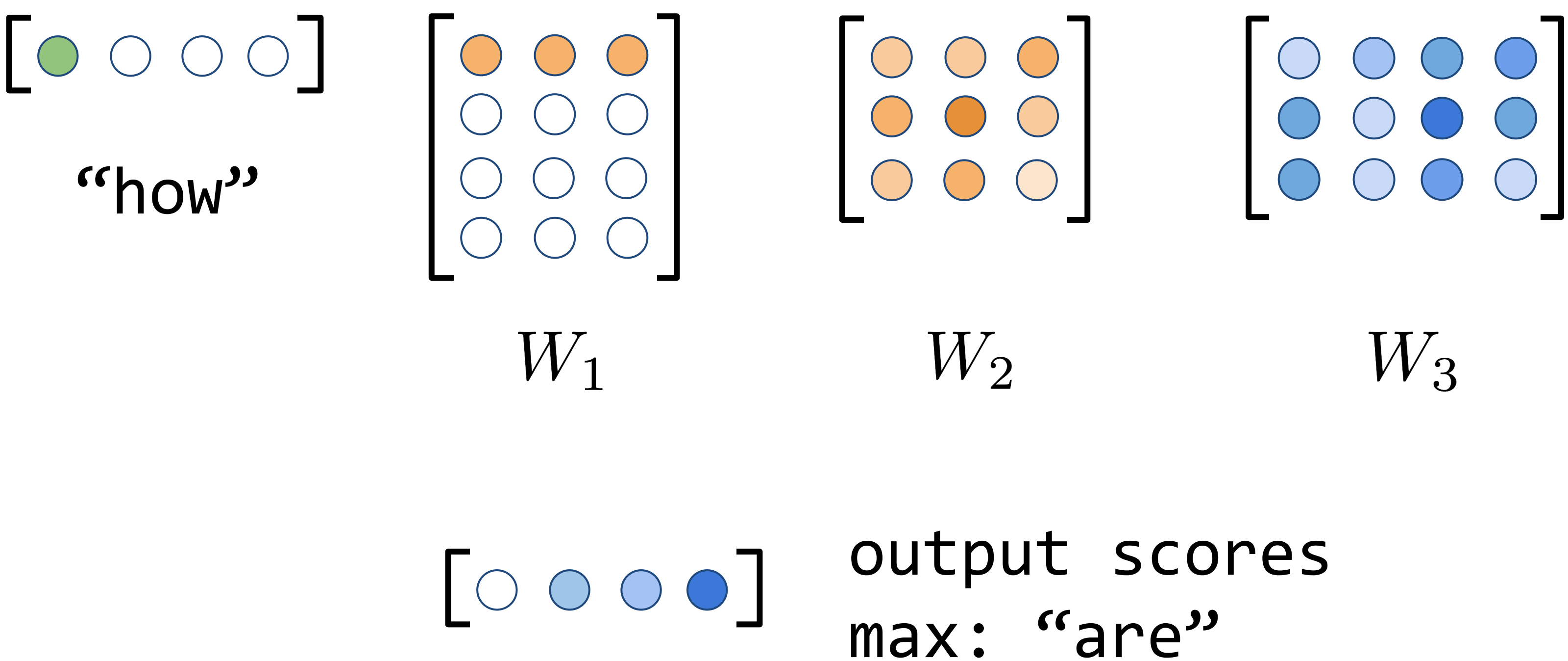




output scores
max: “how”

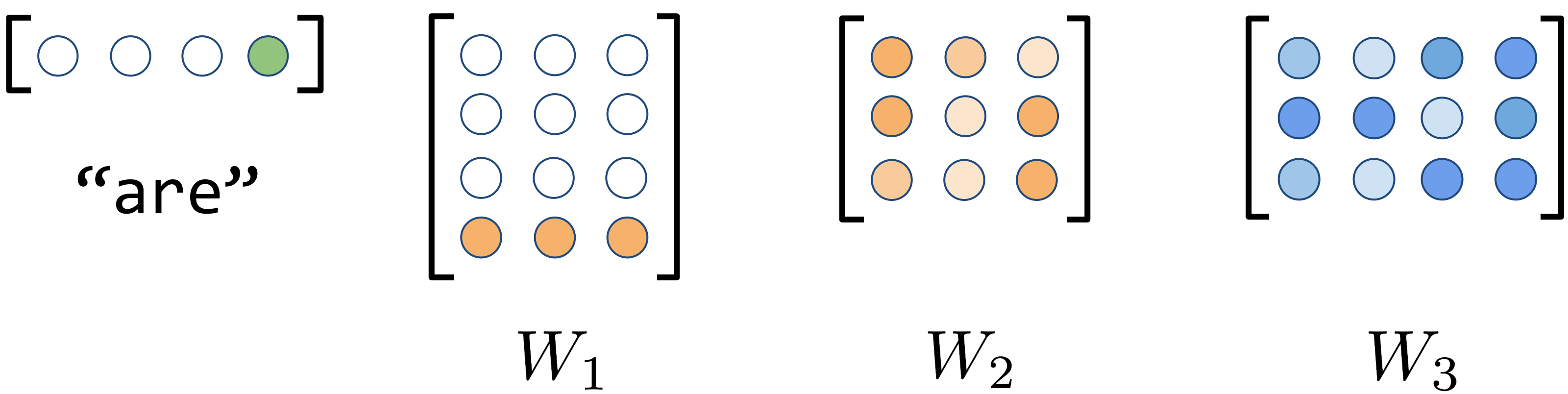
Neural Network Language Model

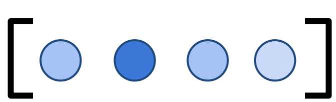
Dataset: `{“how”, “you”, “hello”, “are”}`



Neural Network Language Model

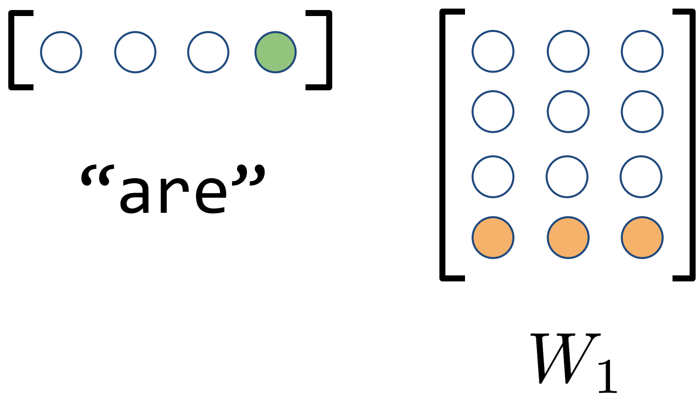
Dataset: {“how”, “you”, “hello”, “are”}





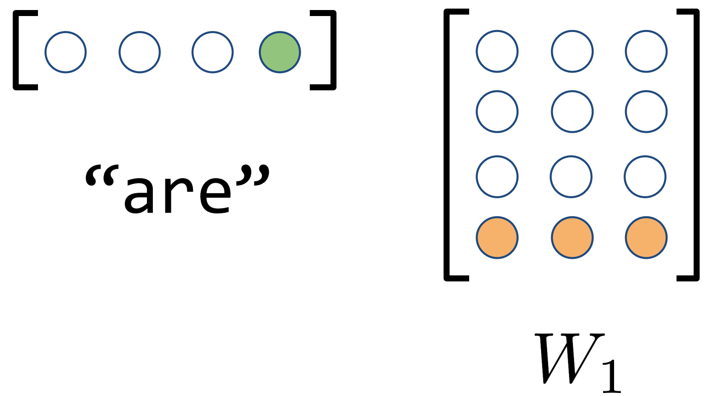
```
output scores
max: “you”
```

Neural Network Language Model



Each one-hot vector turns on one row in weight matrix and results in `[1 x 3]` vector

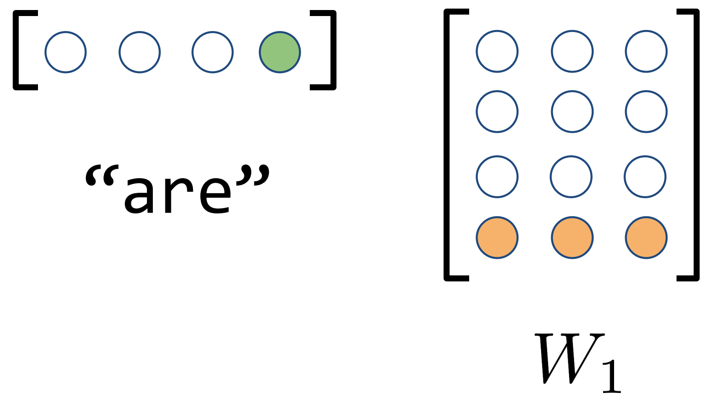
Neural Network Language Model



Each one-hot vector turns on one row in weight matrix and results in `[1 x 3]` vector

Can we say that the `[1 x 3]` vector represents the input word?

Neural Network Language Model



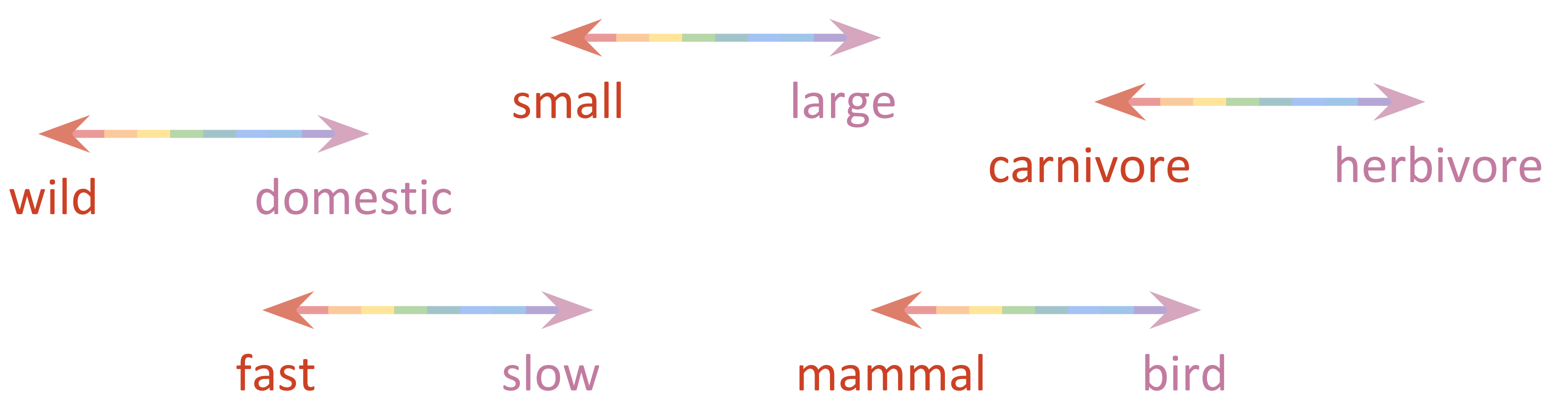
Each one-hot vector turns on one row in weight matrix and results in `[1 x 3]` vector

Can we say that the `[1 x 3]` vector represents the input word? **Yes**

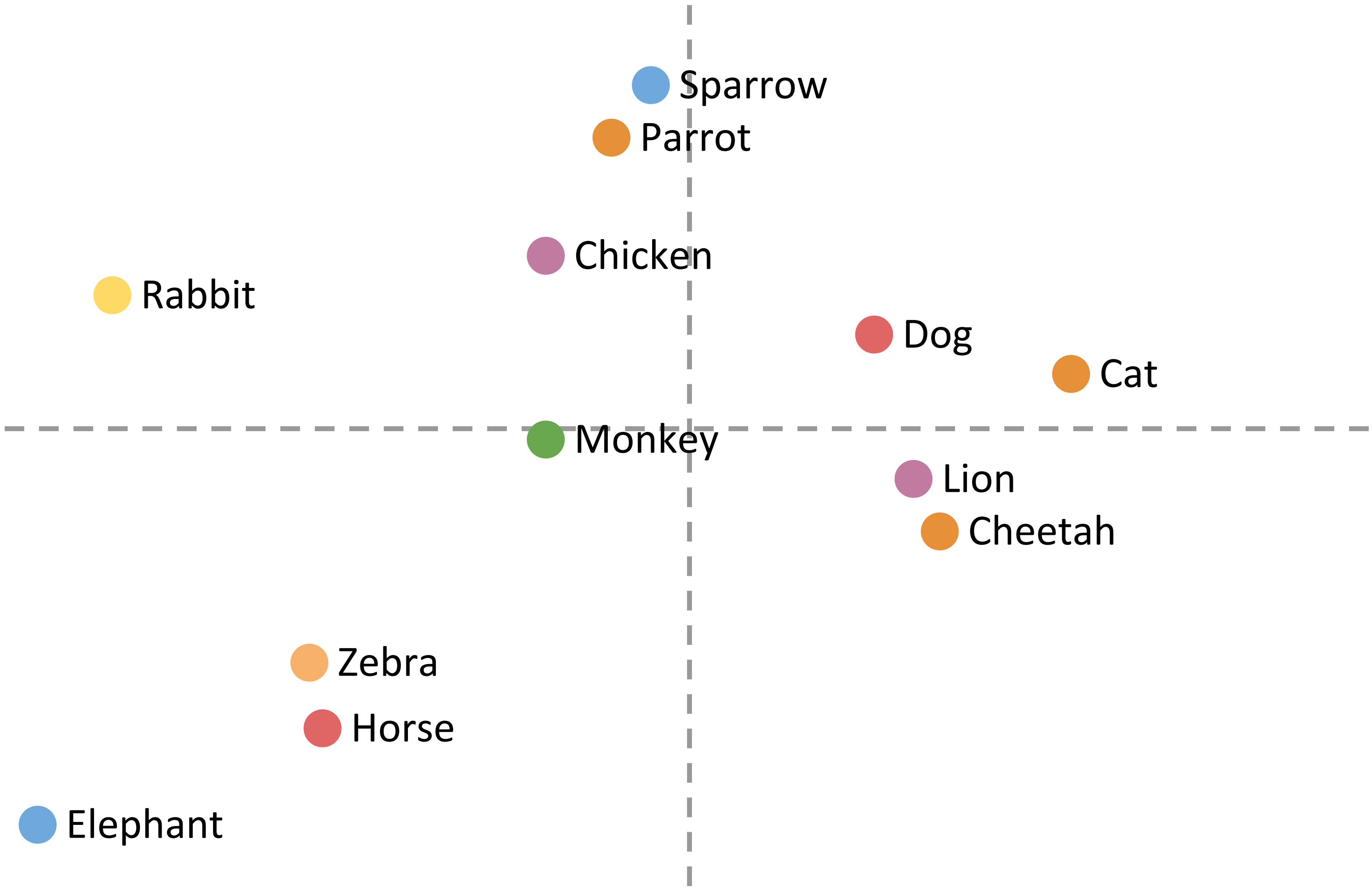
Exercise!

Create a 2D vector space representation of the following words:

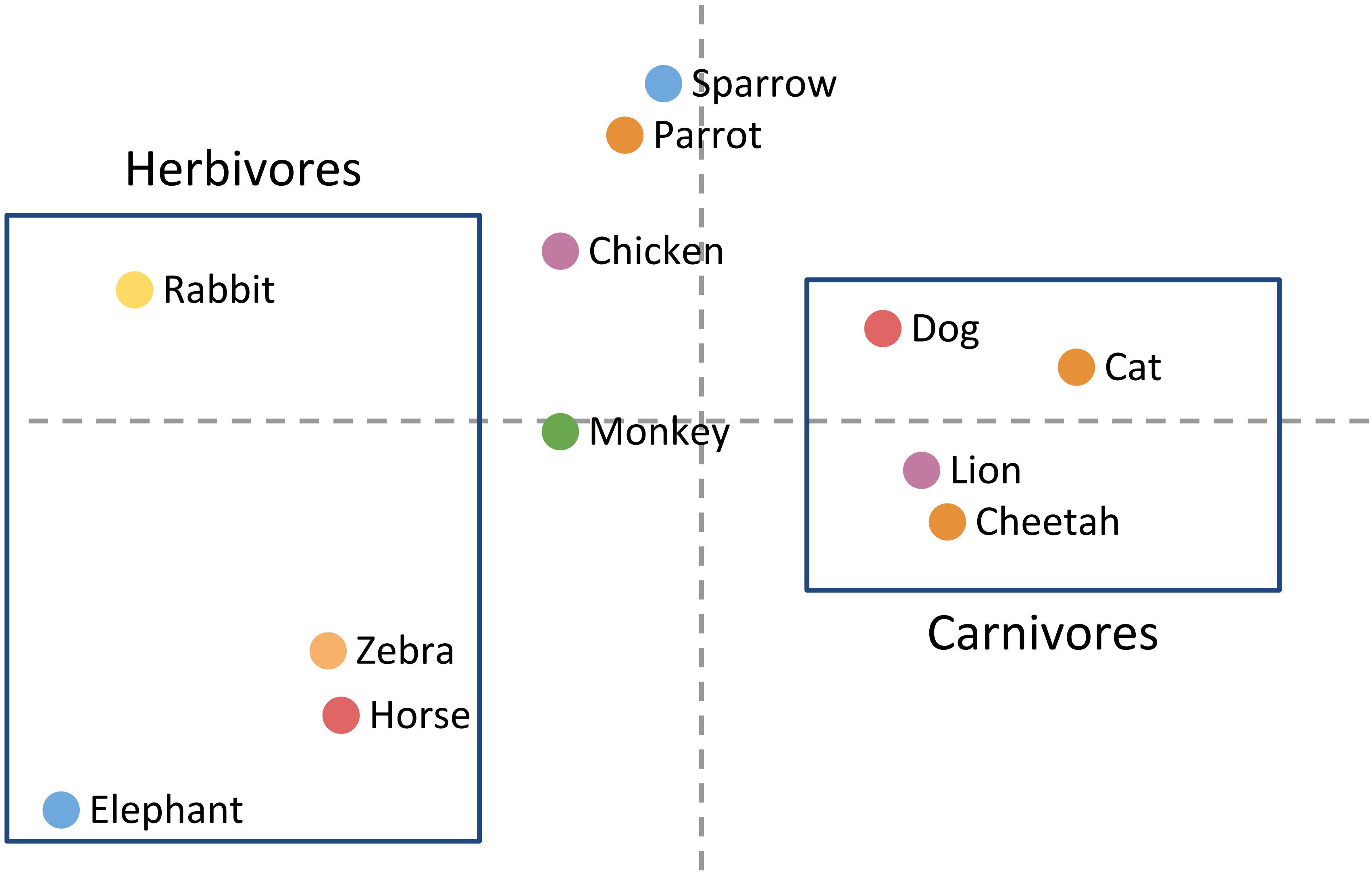
`dog, lion, cat, rabbit, horse, zebra, cheetah, parrot, sparrow, elephant, chicken, monkey`



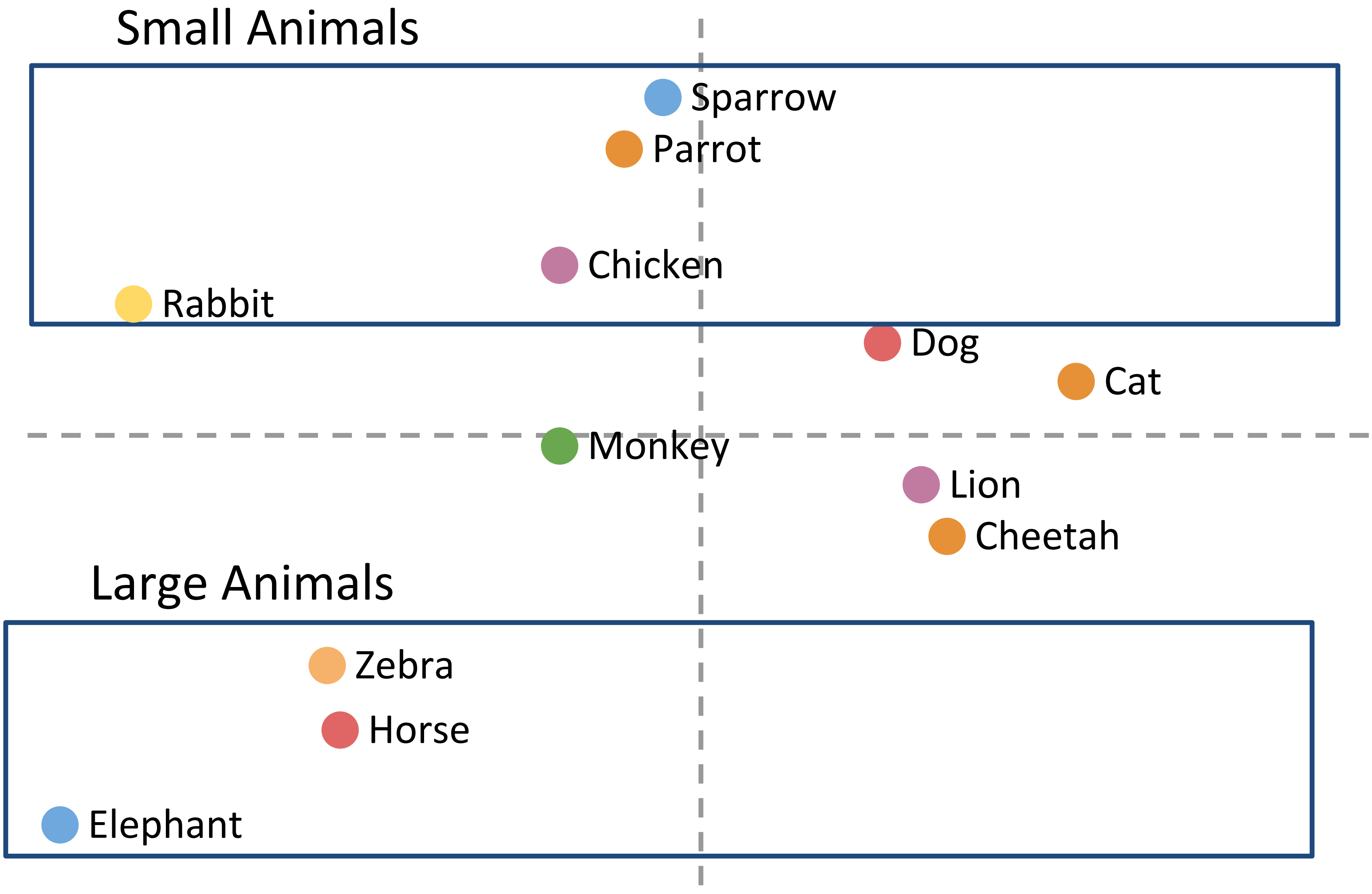
Exercise


Sparrow
Parrot
Chicken
Rabbit
Dog
Cat
Monkey
Lion
Cheetah
Zebra
Horse
Elephant

Exercise

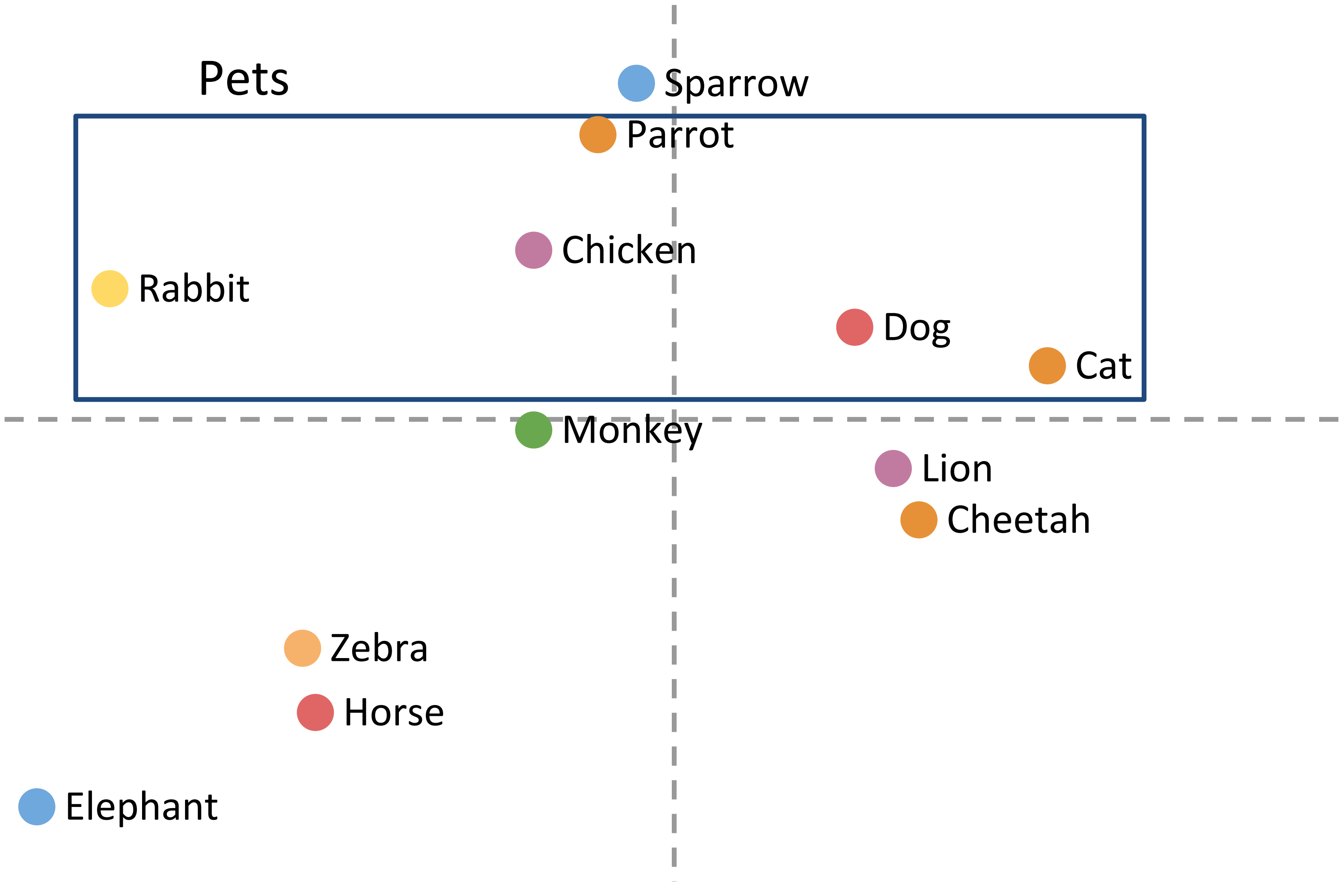

Herbivores
Sparrow
Parrot
Chicken
Rabbit
Dog
Cat
Monkey
Lion
Cheetah
Zebra
Horse
Elephant
Carnivores

Exercise



Small Animals
Sparrow
Parrot
Chicken
Rabbit
Dog
Cat
Monkey
Lion
Cheetah
Large Animals
Zebra
Horse
Elephant

Exercise



Pets
Sparrow
Parrot
Chicken
Rabbit
Dog
Cat
Monkey
Lion
Cheetah
Zebra
Horse
Elephant

Word Embeddings

How did you decide which animals need to be closer?

How did you handle conflicts between animals that belong to multiple groups?

How does having this kind of vector space representation help us?

Word Embeddings

- In one-hot vector representation, a word is represented as one large ***sparse*** vector

only one element is 1 in the entire vector

vectors of different words do not give us any information about the potential relations between the words!

Word Embeddings

- In one-hot vector representation, a word is represented as one large ***sparse*** vector
- Instead, **word embeddings** are ***dense*** vectors in some vector space

Word Embeddings

- In one-hot vector representation, a word is represented as one large ***sparse*** vector
- Instead, **word embeddings** are ***dense*** vectors in some vector space

word vectors are ***continuous*** representations of words

vectors of different words give us information about the potential relations between the words - words closer together in meaning have vectors closer to each other

Word Embeddings

$$\text{cat} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad \longrightarrow \qquad \text{cat} = \begin{bmatrix} 0.523 \\ -0.813 \\ 0.417 \\ 1.738 \\ -0.145 \\ -3.451 \\ \vdots \\ 0.451 \end{bmatrix}$$

one-hot vector

word embedding

Word Embeddings

"Representation of words in continuous space"

Inherit benefits

- Reduce dimensionality
- Semantic relatedness
- Increase expressiveness
  - one word is represented in the form of several features (numbers)

Word Embeddings

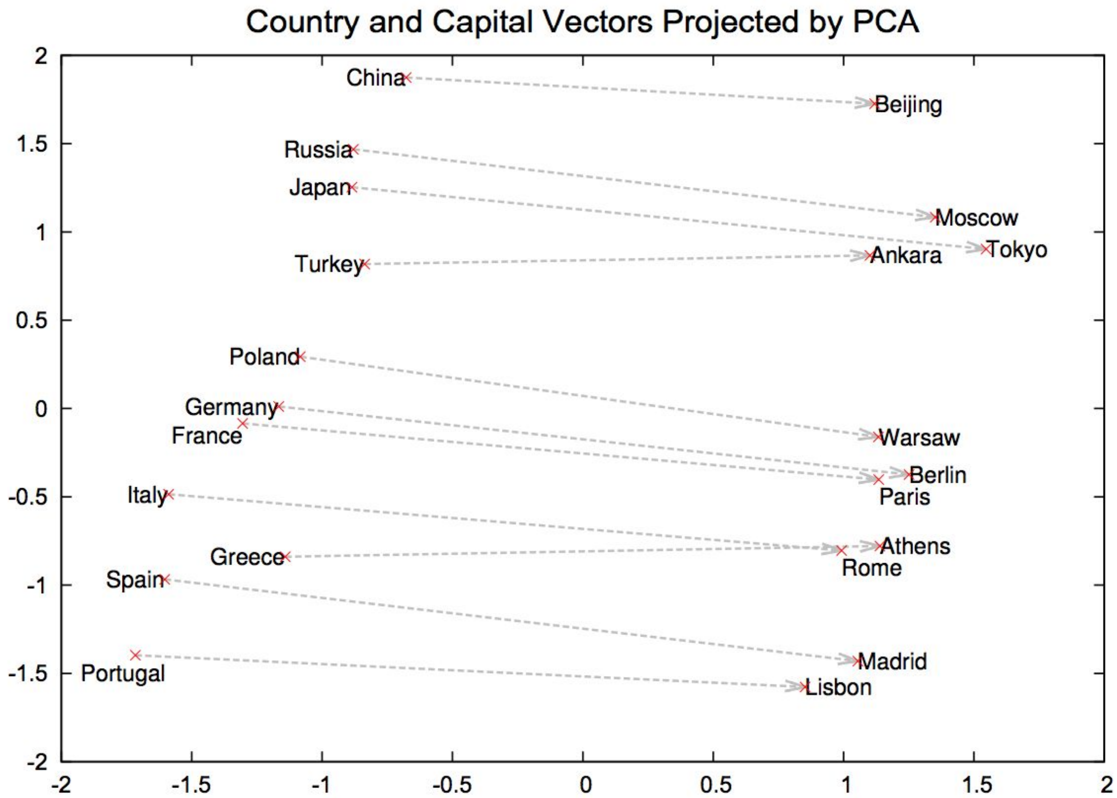
Let’s play with embeddings!

https://rare-technologies.com/word2vec-tutorial/#bonus_app

Try various relationships...

Word Embeddings

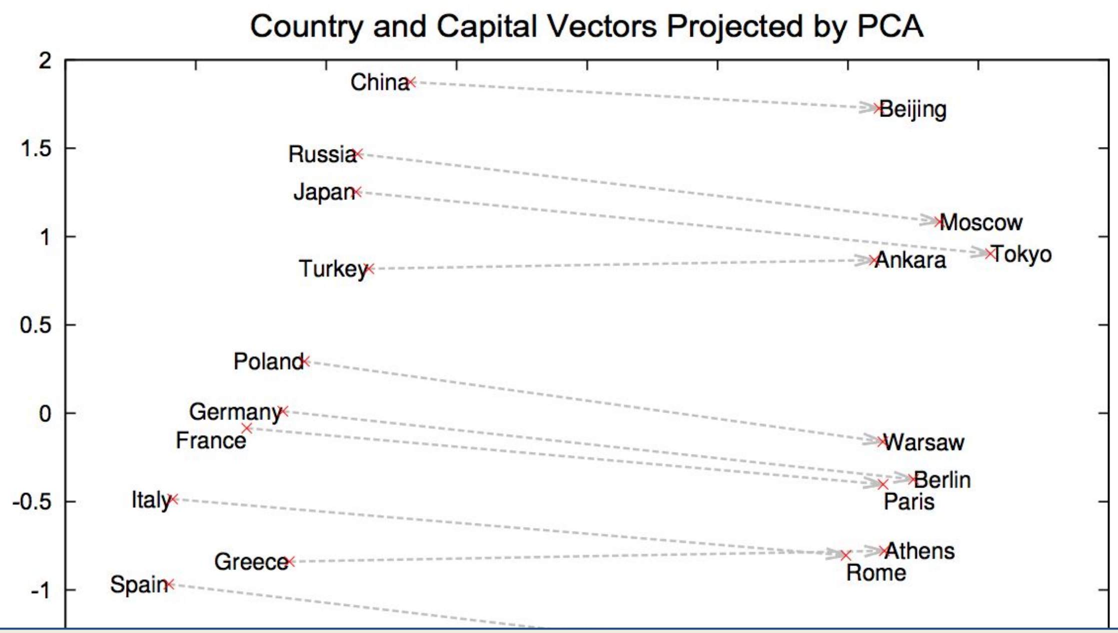
- Plot the embedding vectors



Mikolov et al. 2013

Word Embeddings

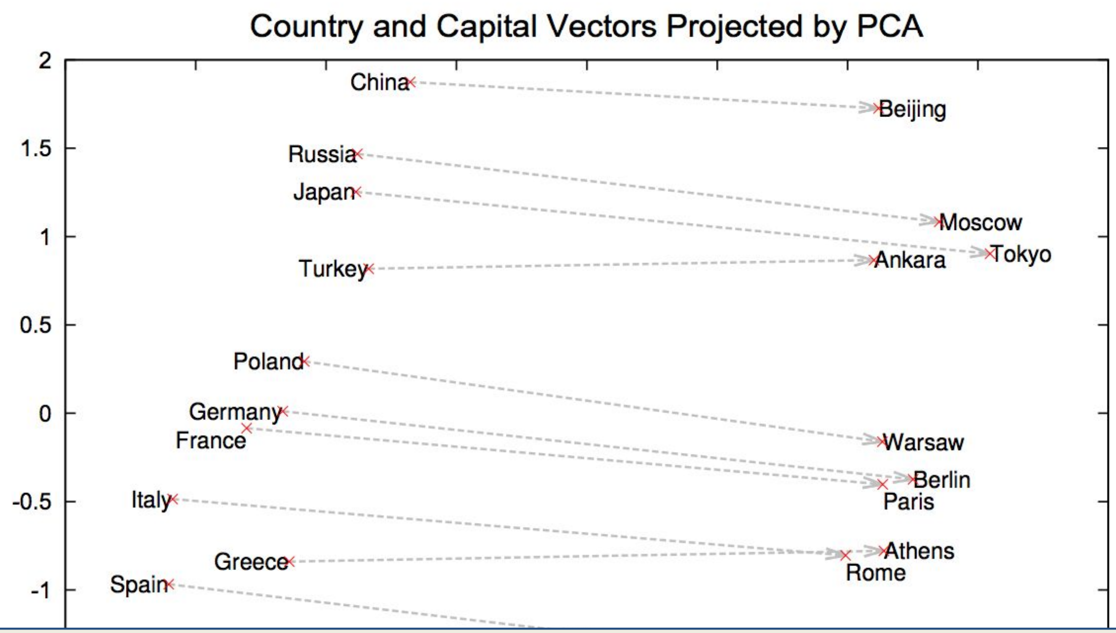
- Plot the embedding vectors



Plot shows the relationship between vectors representing related concepts

Word Embeddings

- Plot the embedding vectors

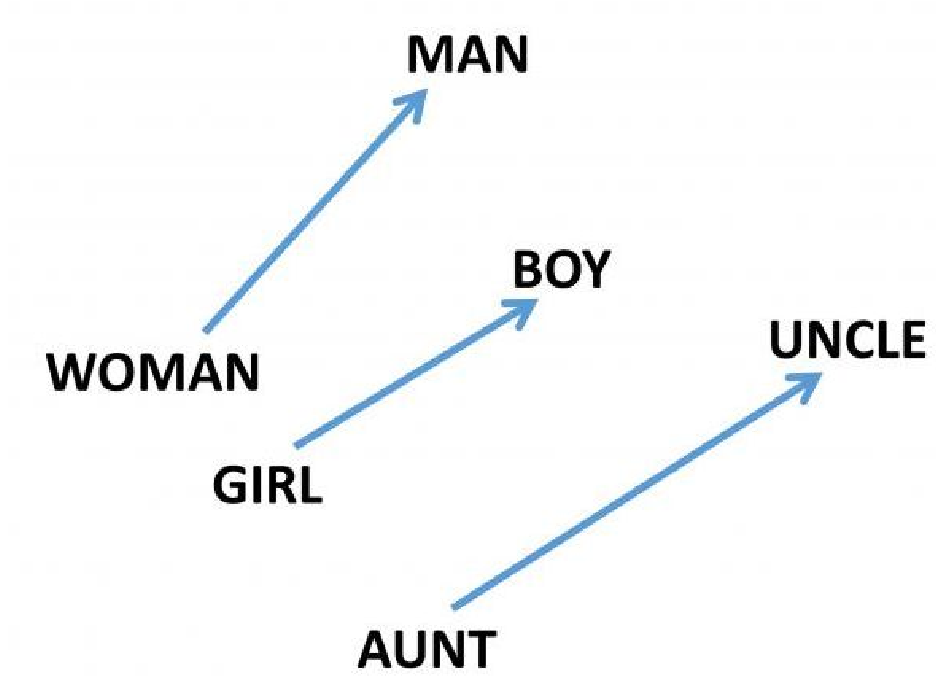


The vectors from countries to capitals point roughly in the same direction

Mikolov et al. 2013

Word Embeddings

- Similarly, learning the gender relationship



Word Embeddings

Q: How can we learn these embeddings automatically?

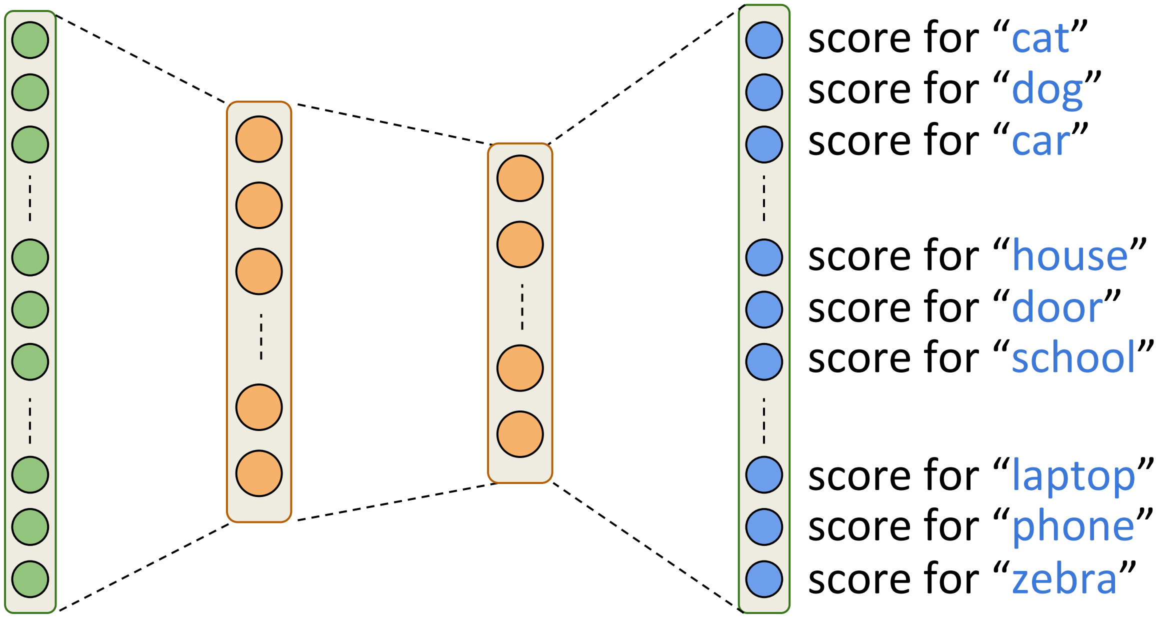
Word Embeddings

Q: How can we learn these embeddings automatically?

A: Neural Networks are a step ahead - embeddings are already learned as “richer” features

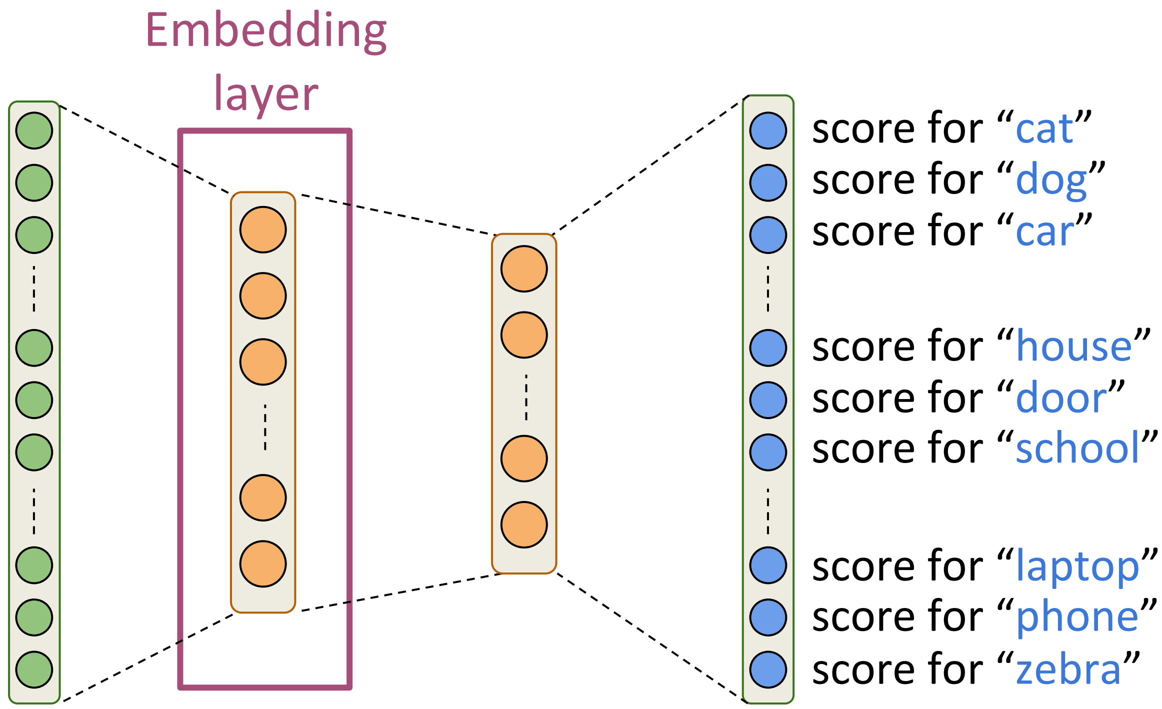
Word Embeddings

Neural Networks are a step ahead - embeddings are already learned as “richer” features



Word Embeddings

Neural Networks are a step ahead - embeddings are already learned as "richer" features



Word Embeddings

The overall **training task** defines the relationships which will be learned by the model

For example:

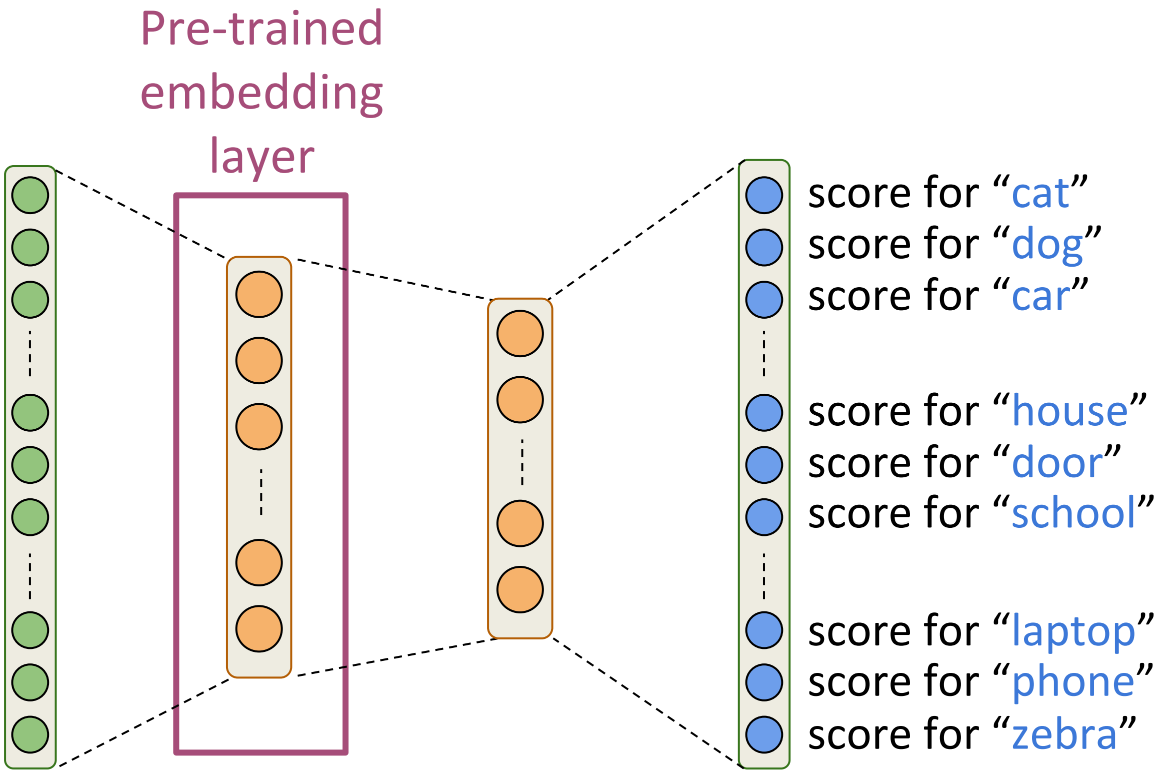
- In language modeling, the model uses neighboring context thus bringing words with similar context closer
- In doing POS tagging task, words with similar POS tags will come close to each other
- If our network is doing Machine Translation, the embeddings will be tuned for translation

Word Embeddings

- Generally, task specific embeddings are better than generic embeddings
- In case of small amount of training data, generic embeddings learned on large amount of data works better
- Generic embeddings can also be used as a starting point

Word Embeddings

We can use pre-trained embeddings as well - just initialize the weights in the first layer with some learned embeddings



Word Embedding Tools

Some tools to learn word embeddings:

- Word2Vec (from Google)
- FastText (from Facebook)
- GloVe (from Stanford)

Word Embeddings

A few pre-trained word embeddings

- GloVe: Wikipedia plus Gigaword https://goo.gl/1XYZhc
- FastText: Wikipedia of 294 languages https://goo.gl/1v423g
- Dependency-based https://goo.gl/tpgw4R

Using pretrained embeddings in keras:

https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html

Neural Network Language Model

Let’s implement a bigram neural network language model in Keras!

Neural Network Language Model

Why does NNLM work better than an ngram based LM like we saw in lecture 2?

Ngram: one word is represented as *one feature*

House vs. Home (two different words)

Neural Network Language Model

Why does NNLM work better than an ngram based LM like we saw in lecture 2?

Ngram: one word is represented as *one feature*

House vs. Home (two different words)

NNLM: one word is represented as *500 fine grained features*

House vs. Home (same concept in space)

Semantic relatedness is learned

Shortcomings

Q: What are some shortcomings of the language modeling algorithms we have seen so far?

Shortcomings

Q: What are some shortcomings of the language modeling algorithms we have seen so far?

A: Independence assumption: We have a “hard” limit on the amount of context we see - bigram, trigram or some ngram.

It is not uncommon to have longer range dependencies in language!

Recurrent Neural Network

Recurrent Neural Network

A sentence consists of a sequence of words

John is driving a car

Recurrent Neural Network

A sentence consists of a sequence of words

John	is	driving	a	car

Recurrent Neural Network

A sentence consists of a sequence of words

John | is | driving | a | car

Think of the sequence as steps in time

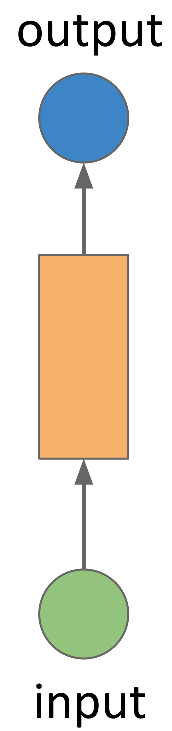
Recurrent Neural Network

A sentence consists of a sequence of words

John	is	driving	a	car
t_1	t_2	t_3	t_4	t_5

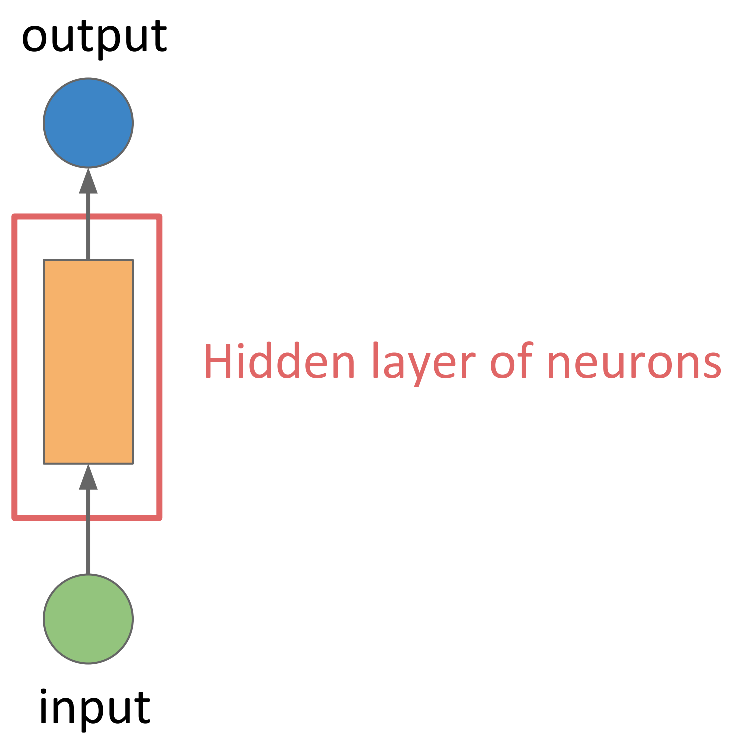
Think of the sequence as steps in time

Recurrent Neural Network



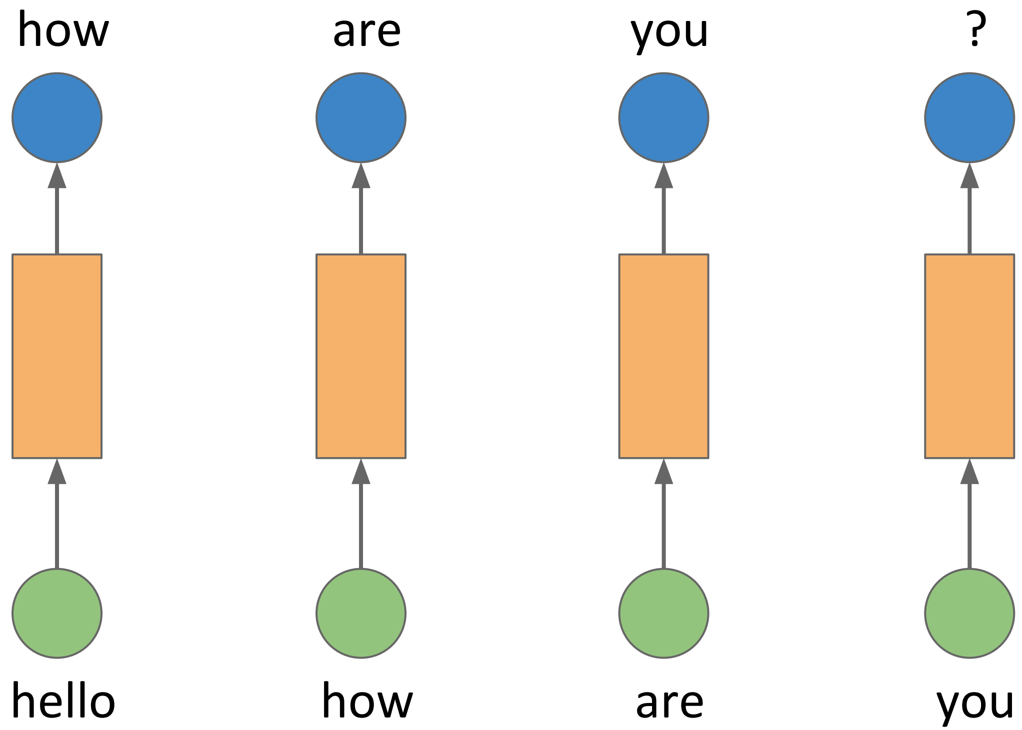
1-layer Feedforward Neural network

Recurrent Neural Network

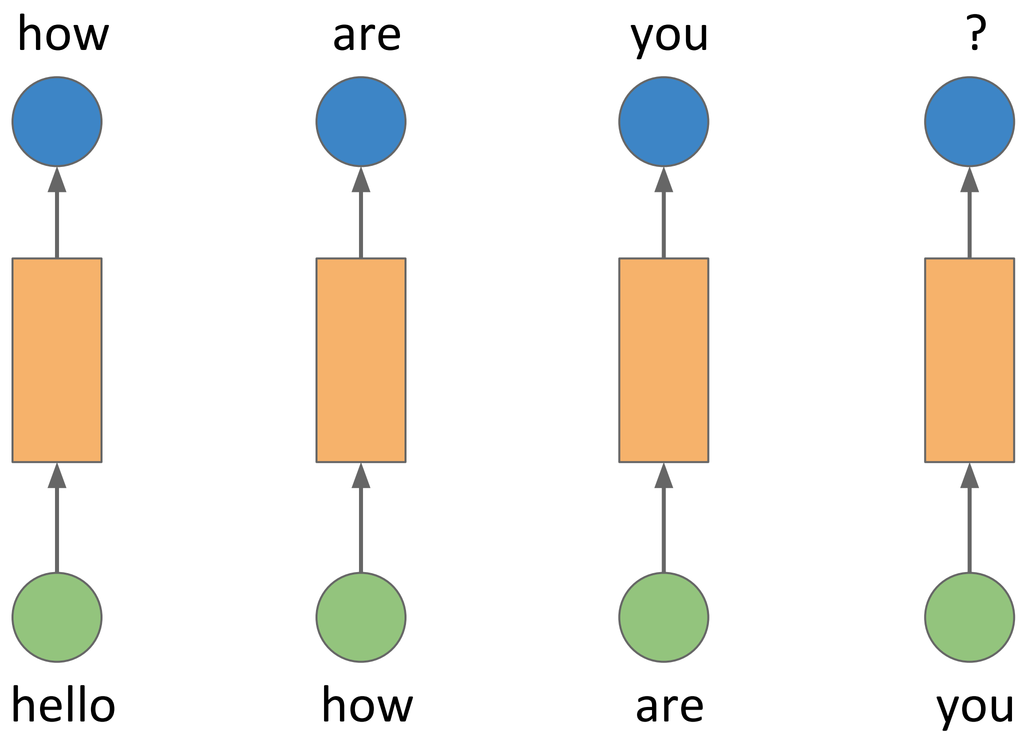


1-layer Feedforward Neural network

Recurrent Neural Network

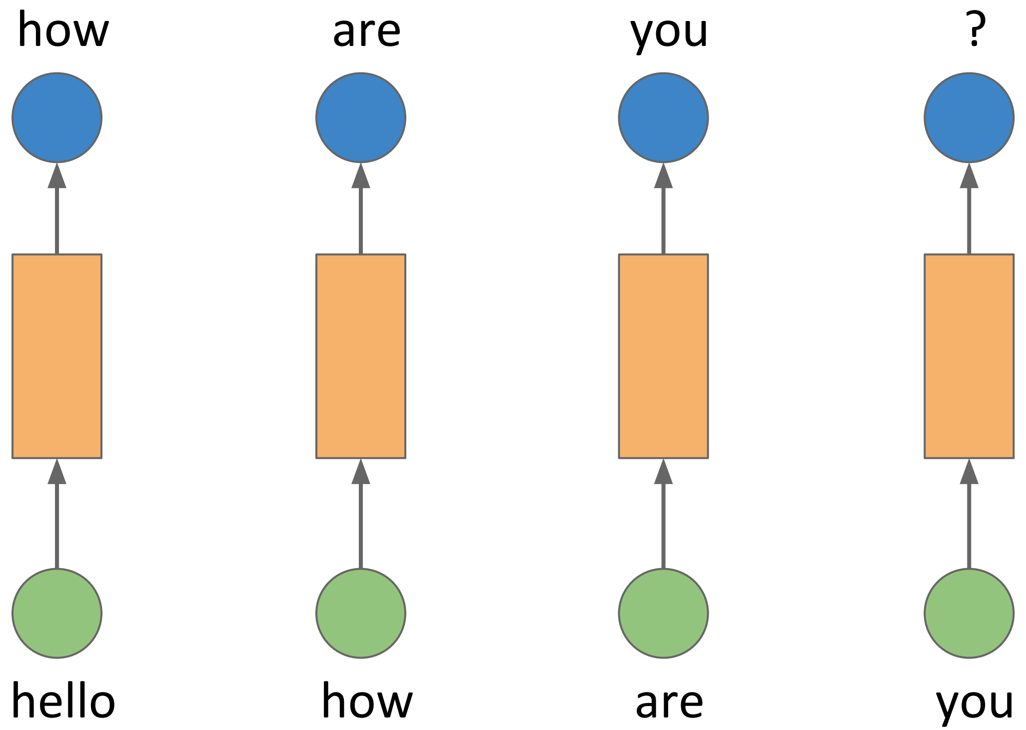


Recurrent Neural Network



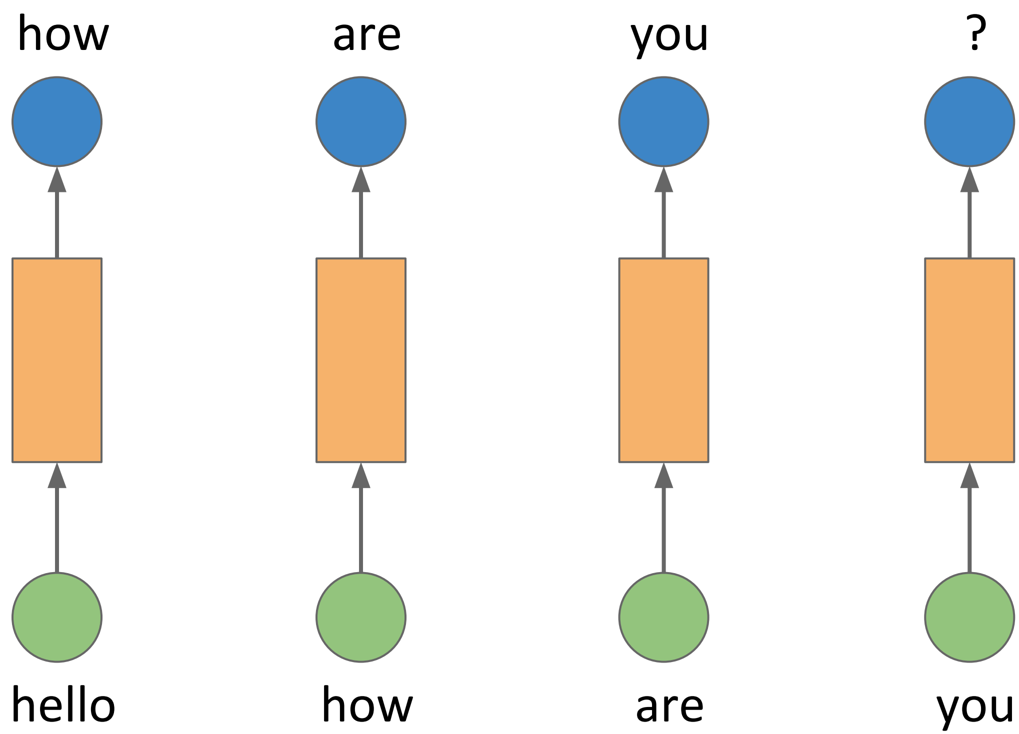
In the real world, we remember some history of previous words

Recurrent Neural Network



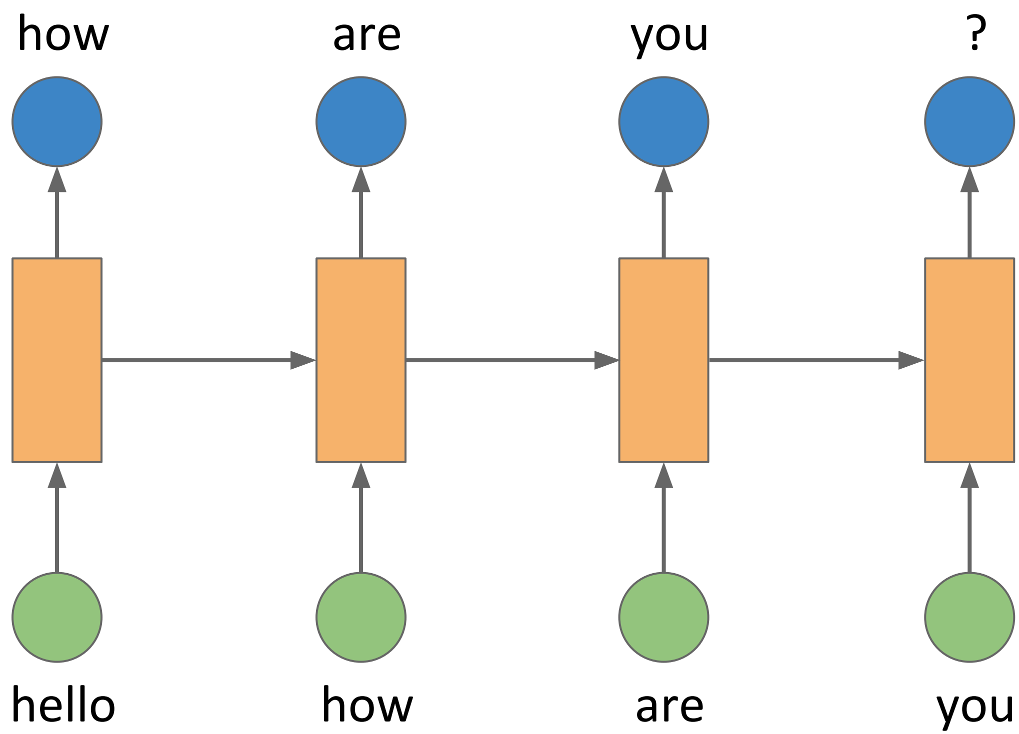
In our network here, each prediction is independent of the previous predictions

Recurrent Neural Network



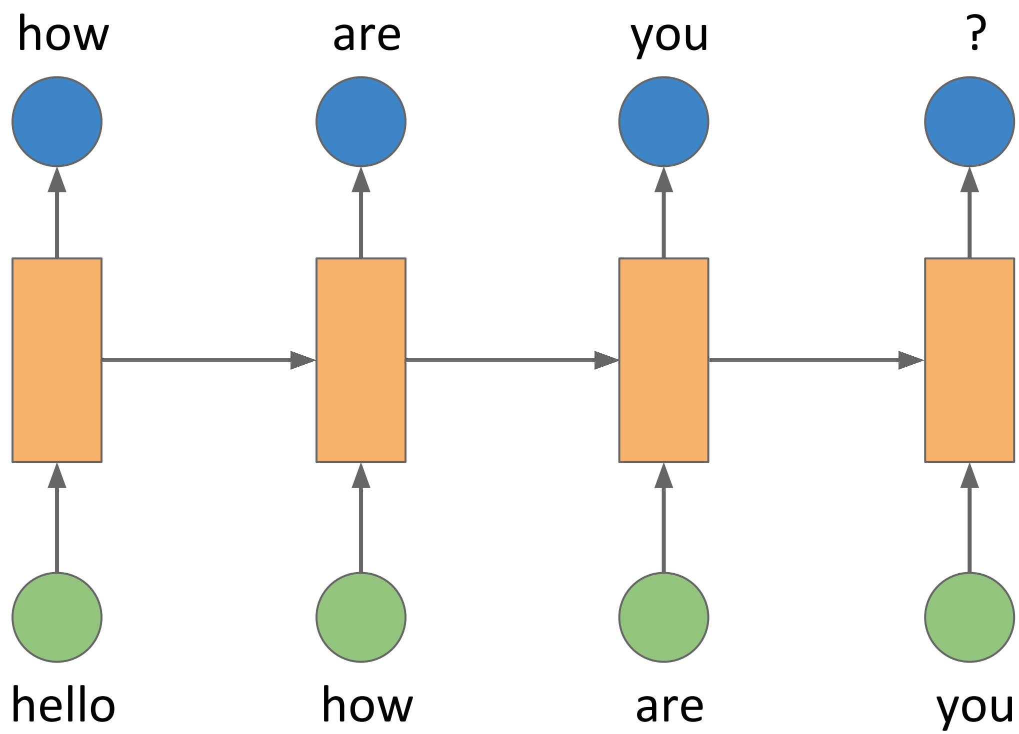
Why not connect these networks?

Recurrent Neural Network



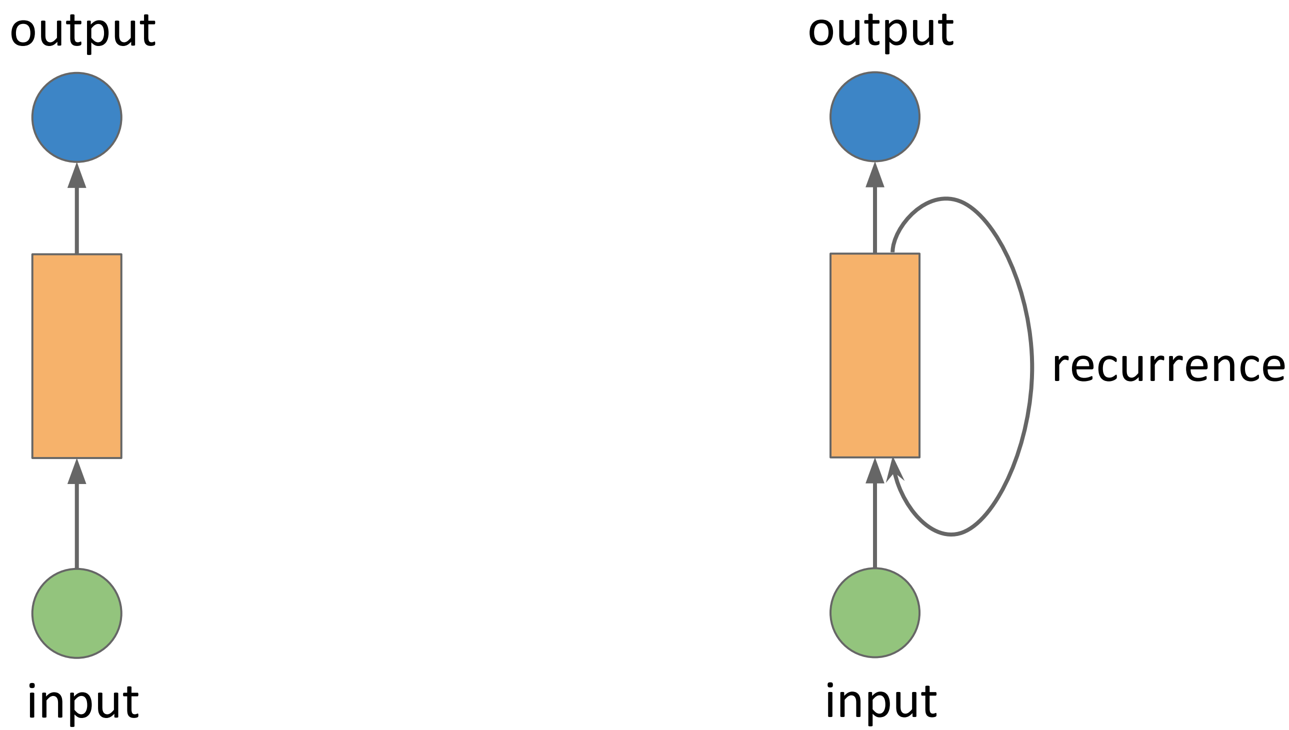
Why not connect these networks?

Recurrent Neural Network



This is what recurrent neural networks do

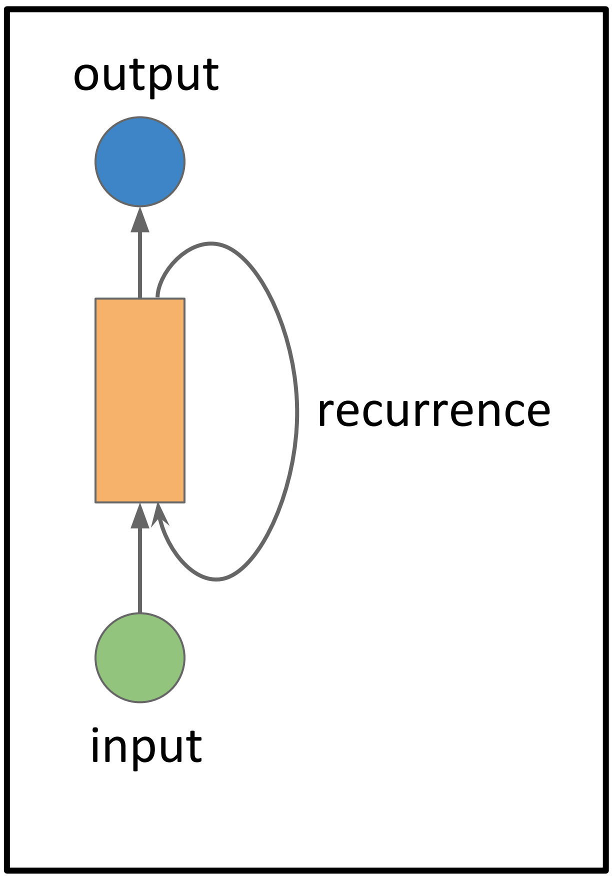
Recurrent Neural Network



1-layer Feedforward
Neural network

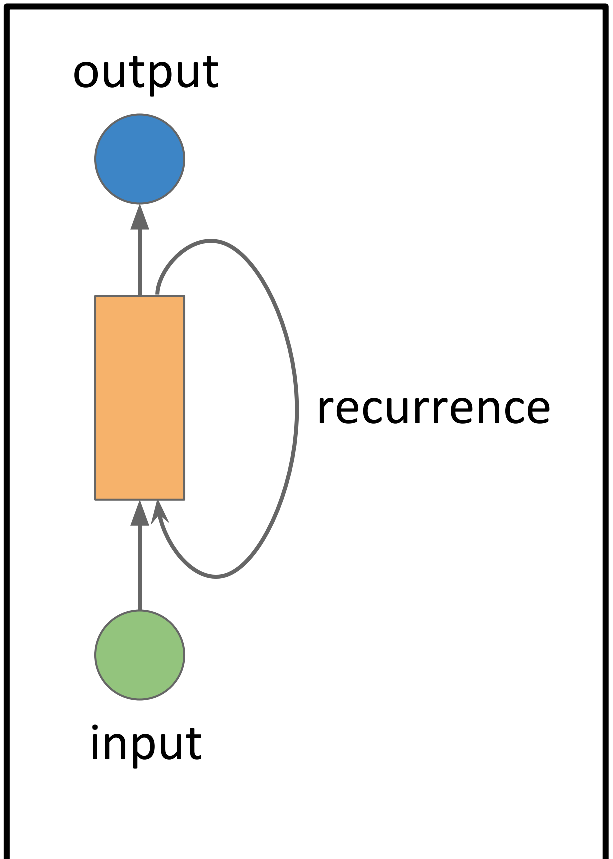
1-layer Recurrent
Neural network

Recurrent Neural Network



Recurrent units work very well for sequential information like a series of words, or knowledge across timesteps

Recurrent Neural Network



Recurrent units work very well for sequential information like a series of words, or knowledge across timesteps

The recurrence unit has two inputs:

1) x_i (input at time i)
2) h_{i-1} (input from previous state)

Recurrent Neural Network

Mathematically,

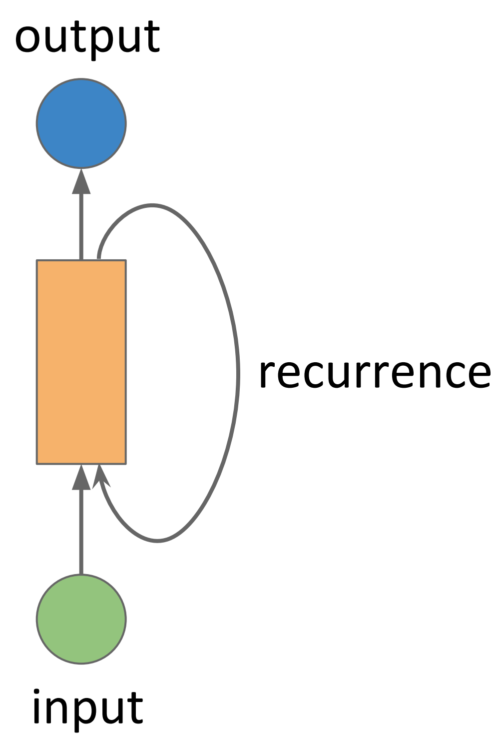
$$h = Wx + b \longrightarrow h_t = Wx + W_h h_{t-1} + b$$

Linear Recurrent

We have one additional set of parameters: W_h, which deals with the information transferred from the previous timestep

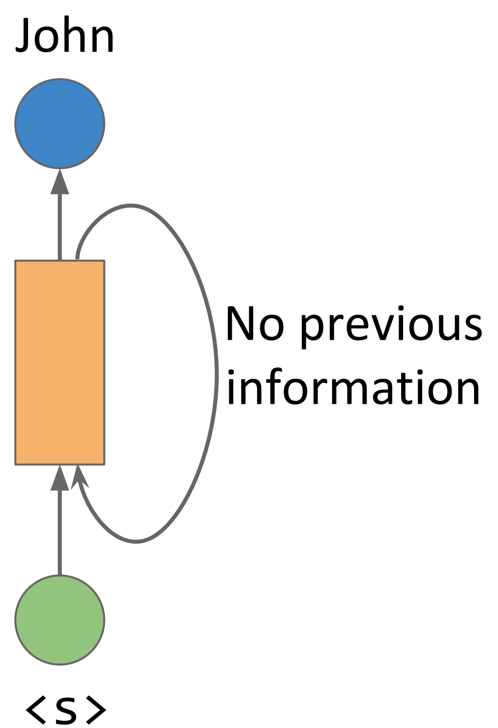
Recurrent Neural Network

Consider an example: <s> John is driving a car </s>



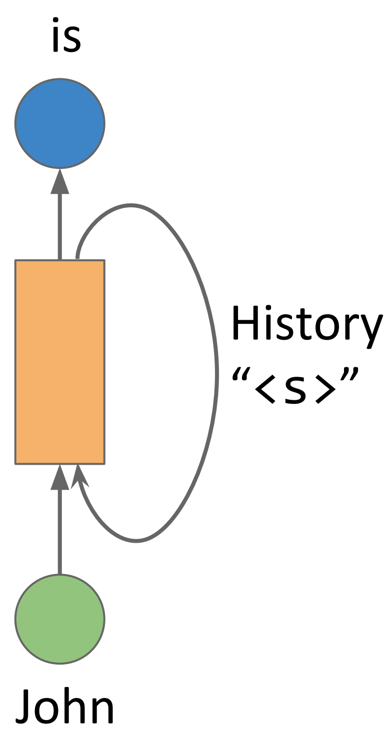
Recurrent Neural Network

Consider an example: <s> John is driving a car </s>



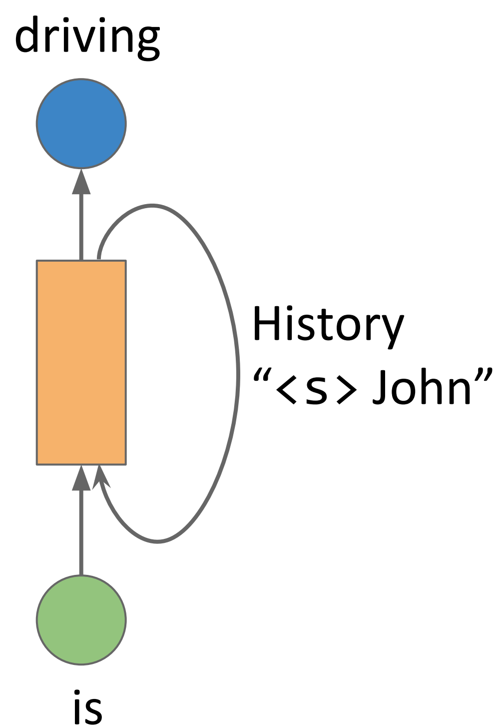
Recurrent Neural Network

Consider an example: <s> John is driving a car </s>



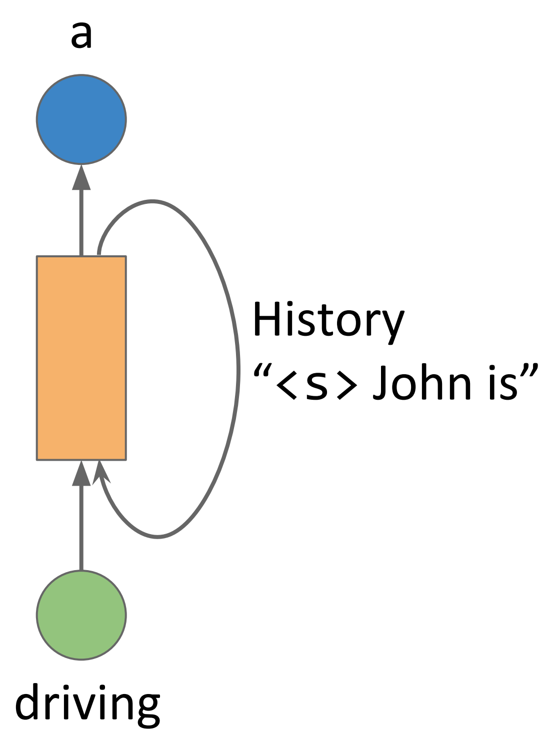
Recurrent Neural Network

Consider an example: <s> John is driving a car </s>



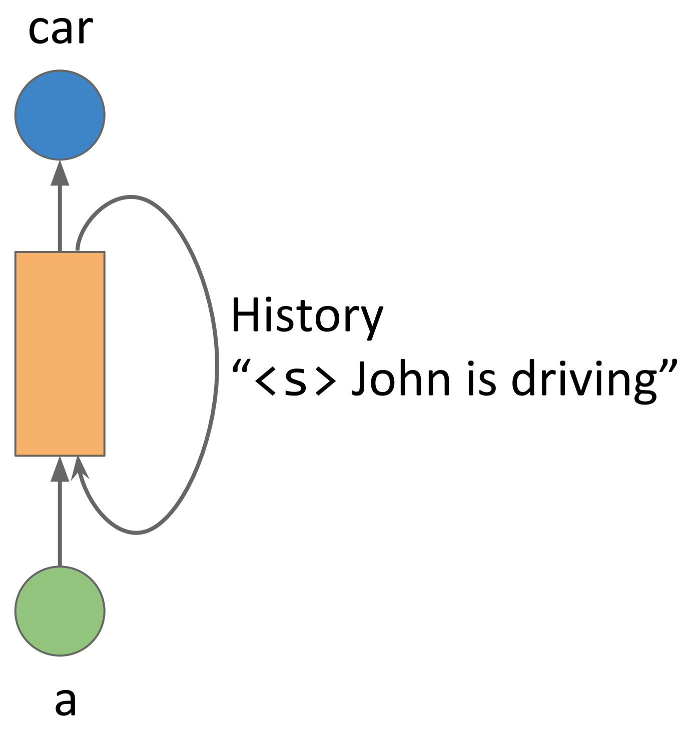
Recurrent Neural Network

Consider an example: <s> John is driving a car </s>



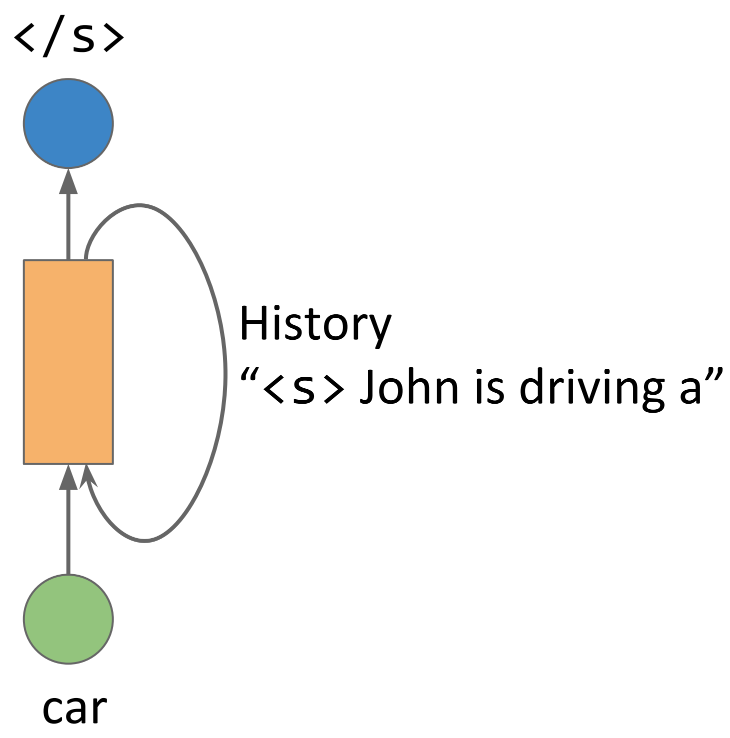
Recurrent Neural Network

Consider an example: <s> John is driving a car </s>



Recurrent Neural Network

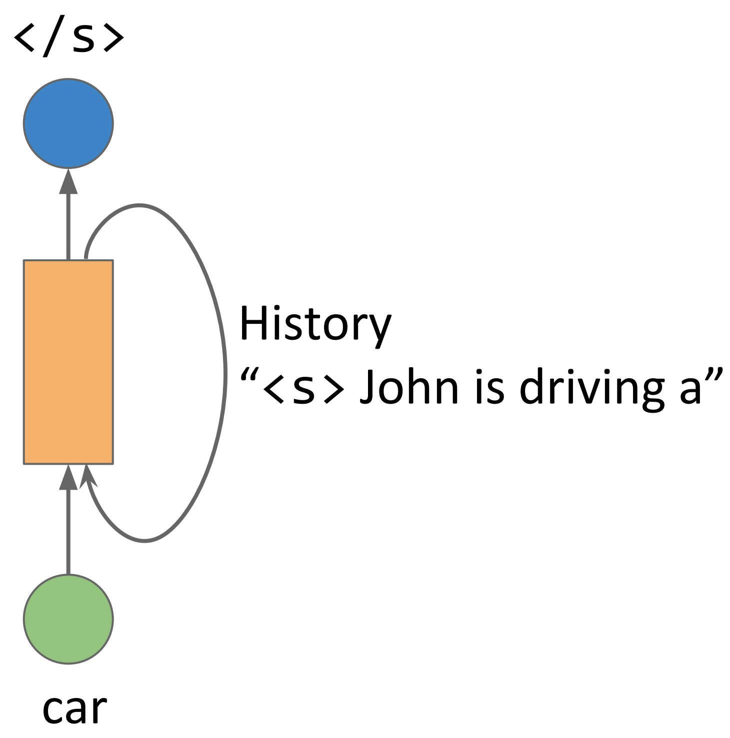
Consider an example: <s> John is driving a car </s>



Recurrent Neural Network

Consider an example: <s> John is driving a car </s>

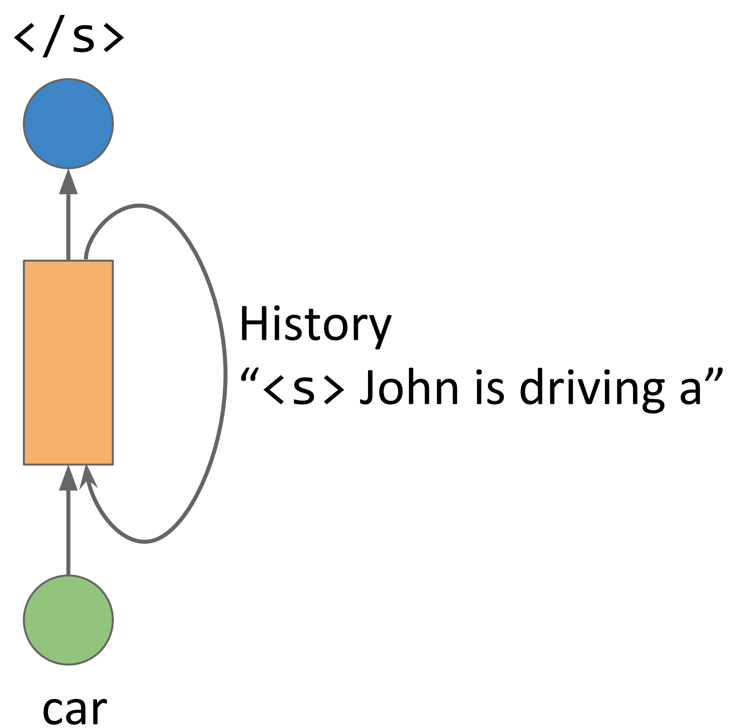
At the last timestep, the hidden state will have information about the entire sentence: **“John is driving a”** from history and **“car”** from the input



Recurrent Neural Network

Consider an example: <s> John is driving a car </s>

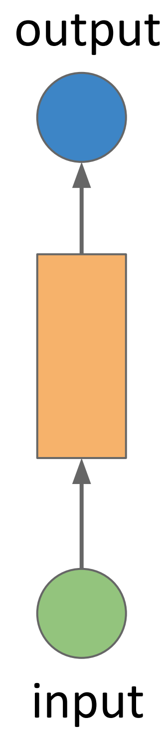
This hidden state can be considered as a "summary" of the entire sentence represented as a vector



Backpropagation through time

for recurrent neural networks

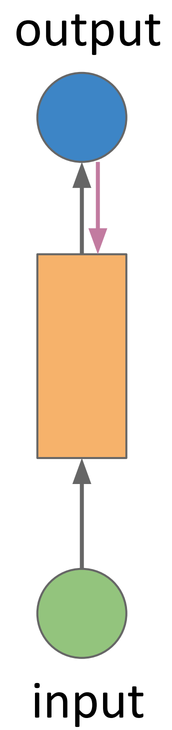
Backpropagation through time



Recall backpropagation in

Feedforward Neural network

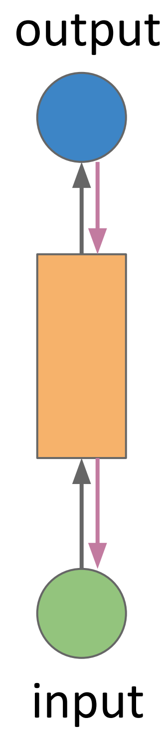
Backpropagation through time



Recall backpropagation in

Feedforward Neural network

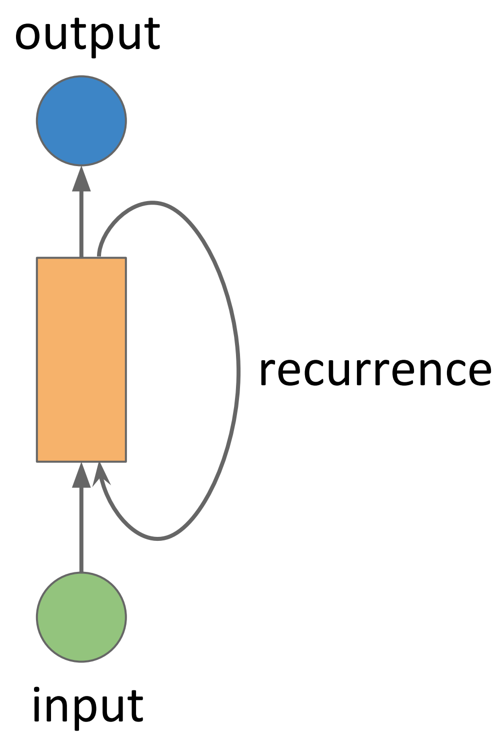
Backpropagation through time



Recall backpropagation in

Feedforward Neural network

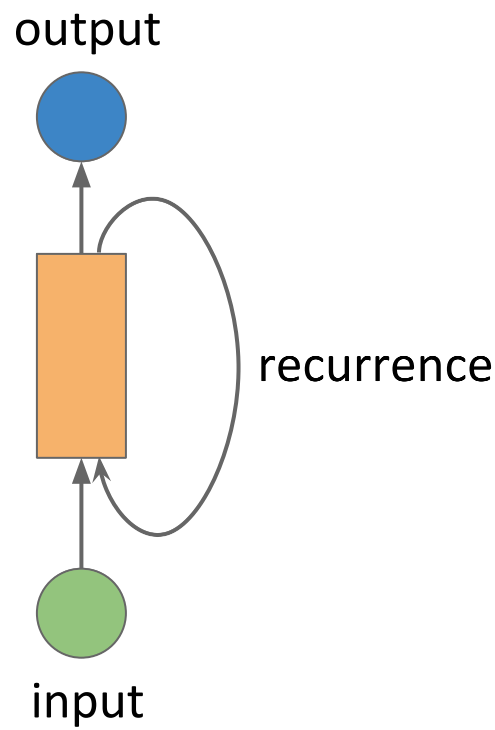
Backpropagation through time



What about backpropagation in recurrent neural networks?

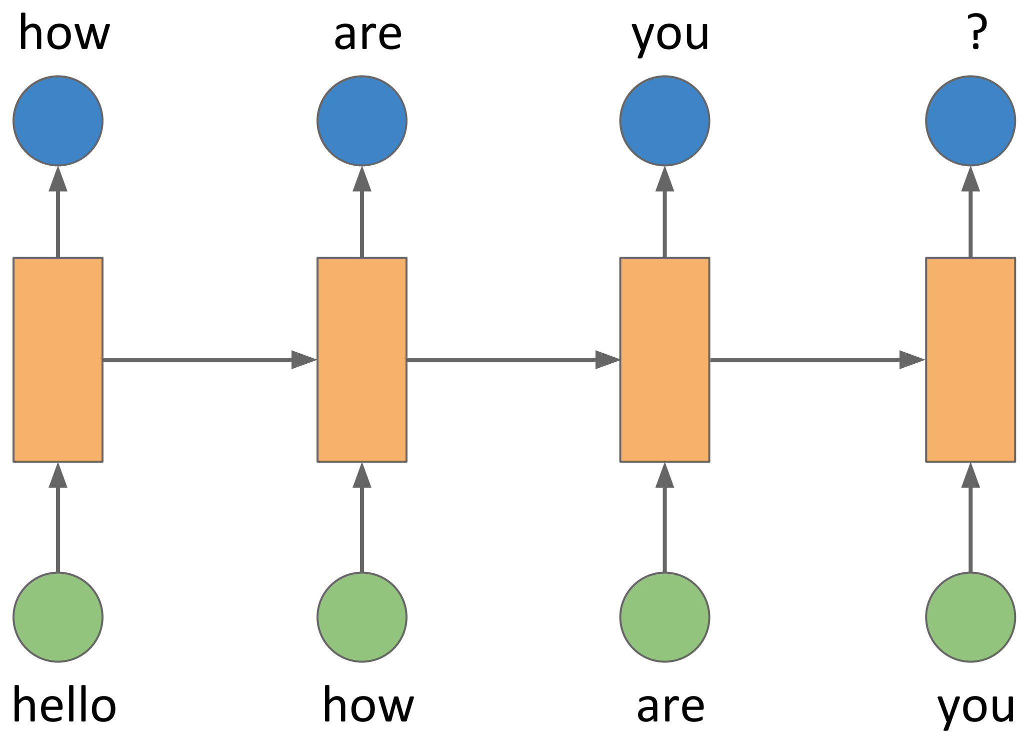
We now have an additional dimension of **time**

Backpropagation through time



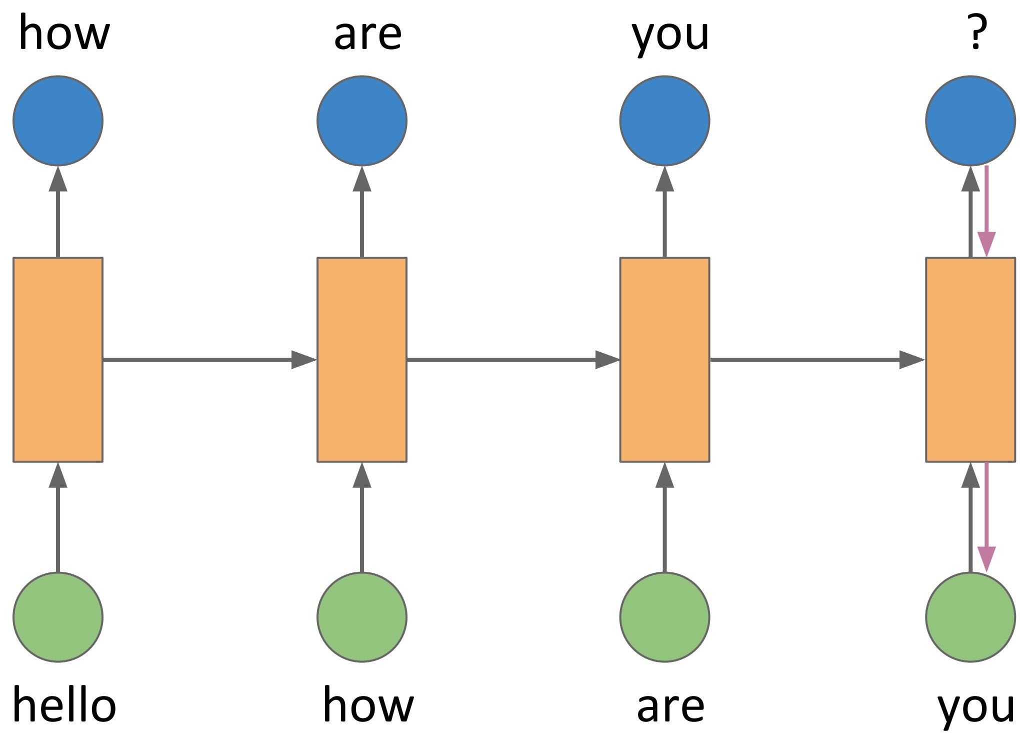
Easier to see when we have ***unrolled*** the RNN

Backpropagation through time



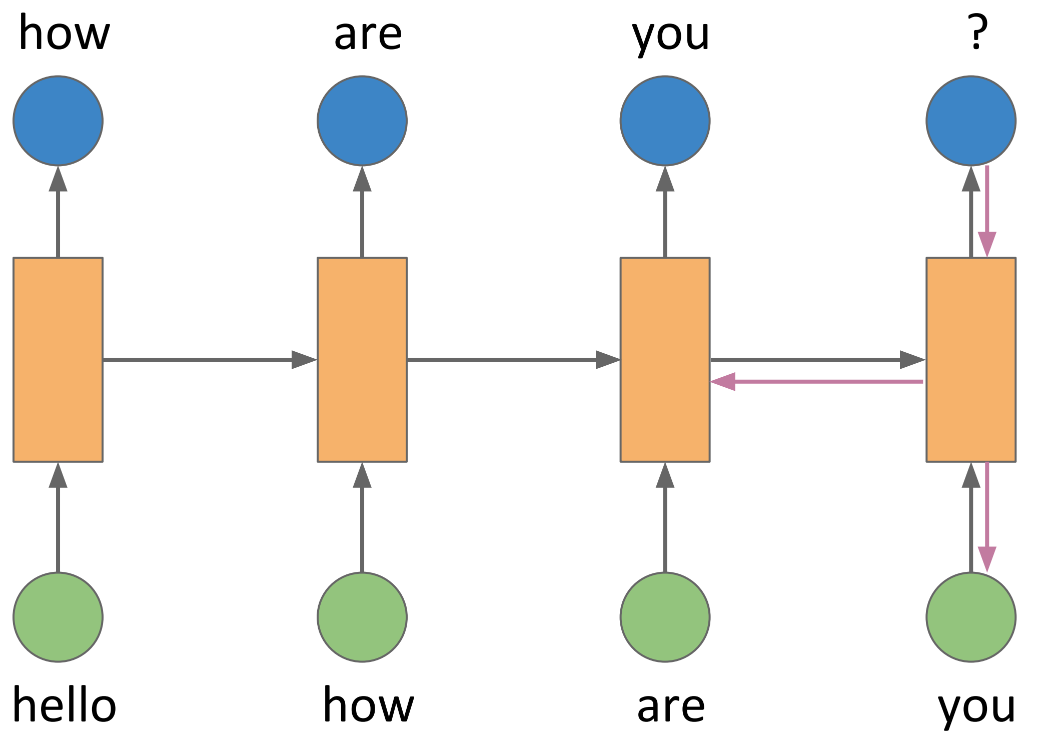
Easier to see when we have ***unrolled*** the RNN

Backpropagation through time



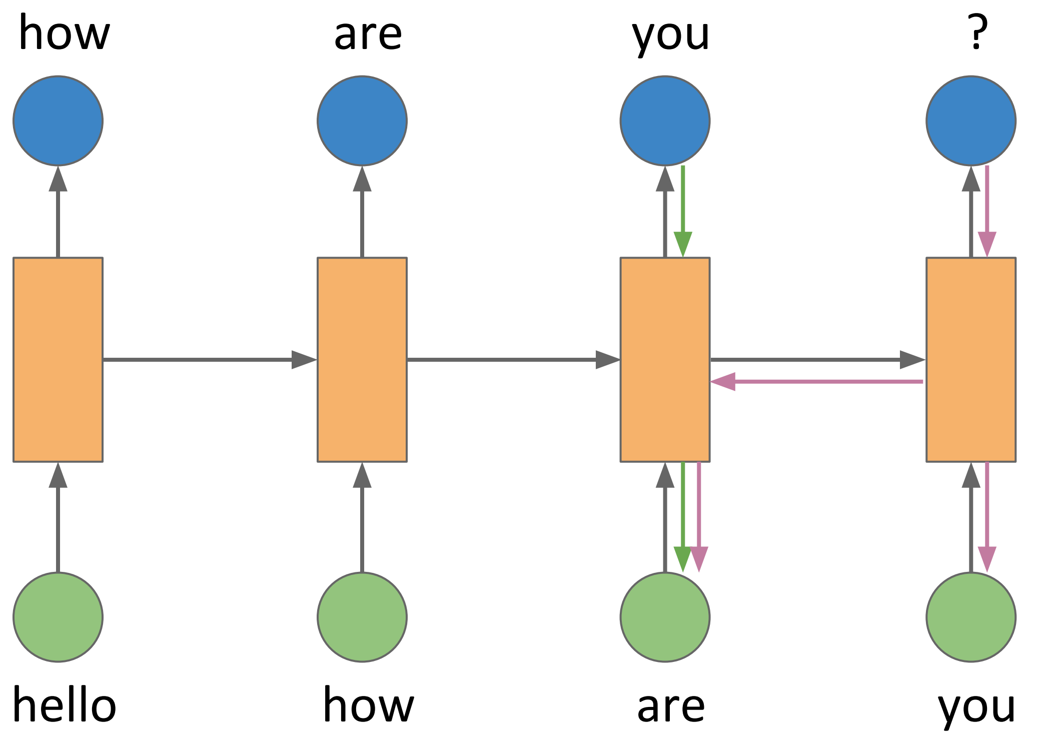
The last timestep propagates its gradient as usual

Backpropagation through time



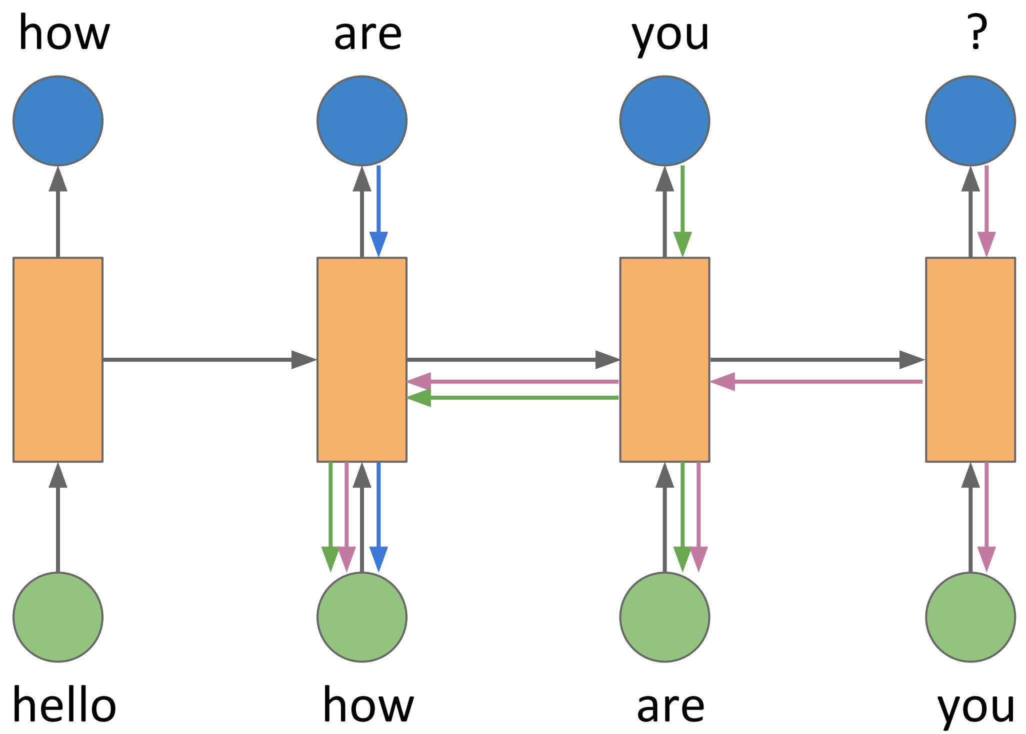
This time, we also propagate the gradient of the last timestep to timestep `t-1`

Backpropagation through time



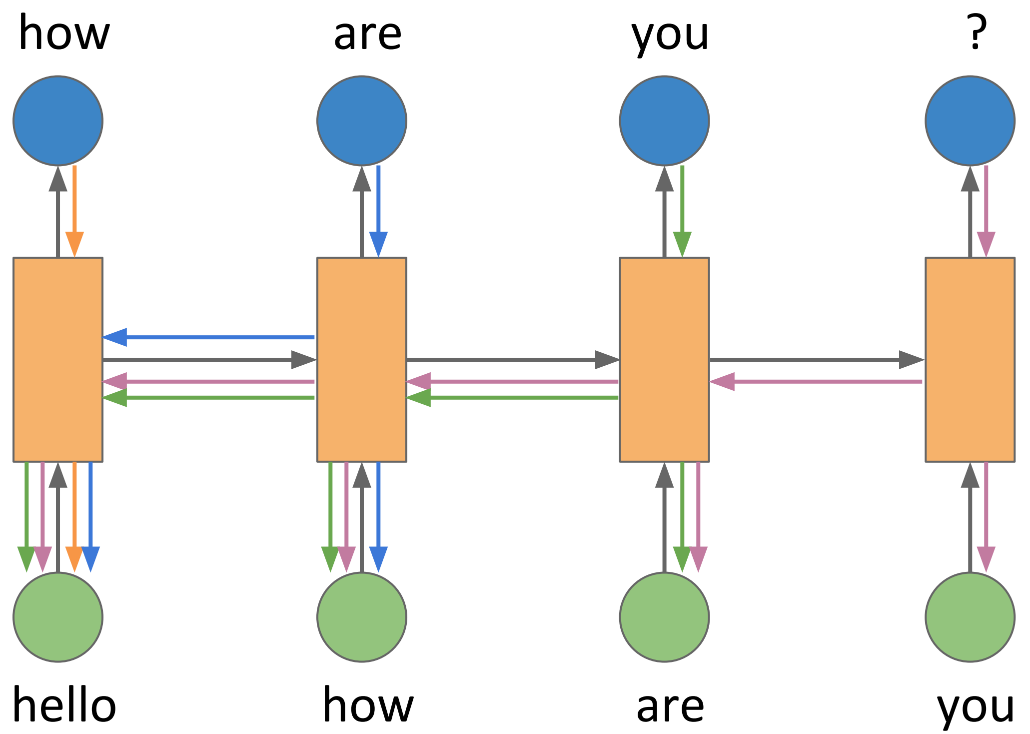
Timestep `t-1` gets gradients from both the output of timestep `t-1` and `t`!

Backpropagation through time



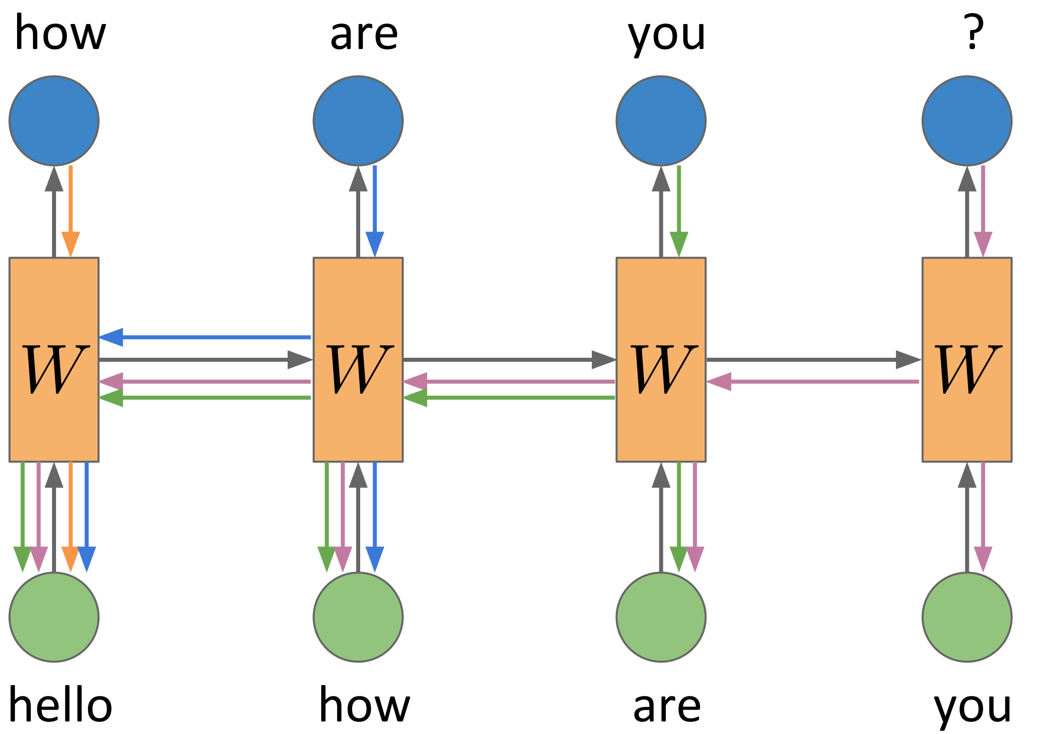
Timestep `t-2` gets gradients from all future timesteps

Backpropagation through time



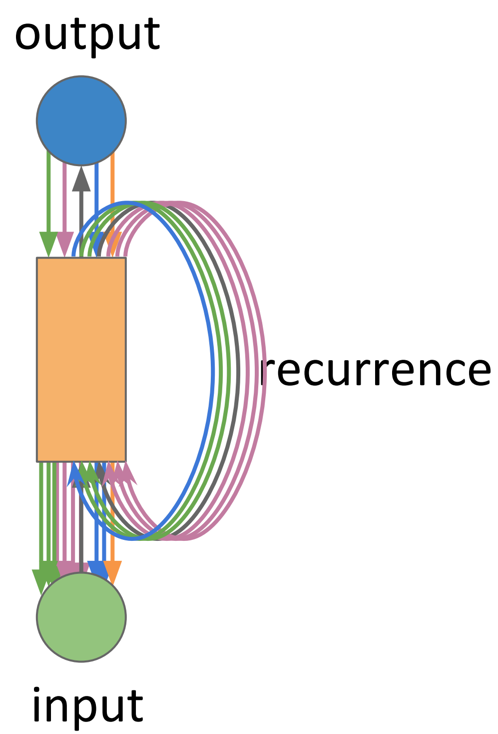
Timestep 1 gets gradients from all future timesteps

Backpropagation through time



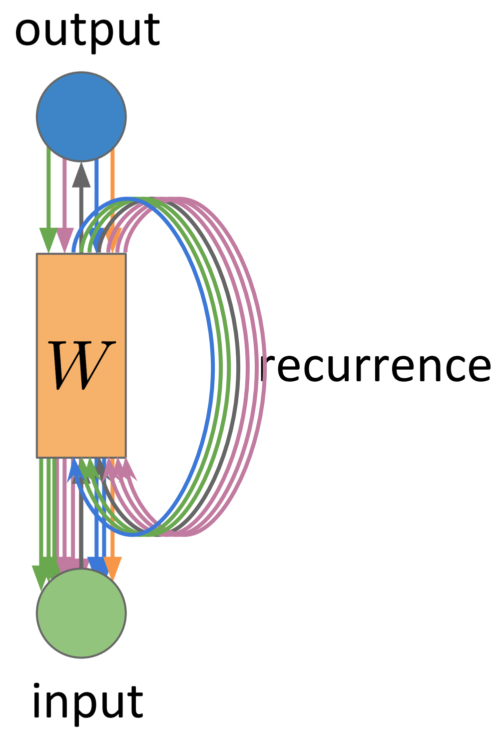
Remember, this is an unrolled network - so the parameters are the same in each of the hidden units!

Backpropagation through time



A bit difficult to see in the ***rolled*** RNN...

Backpropagation through time



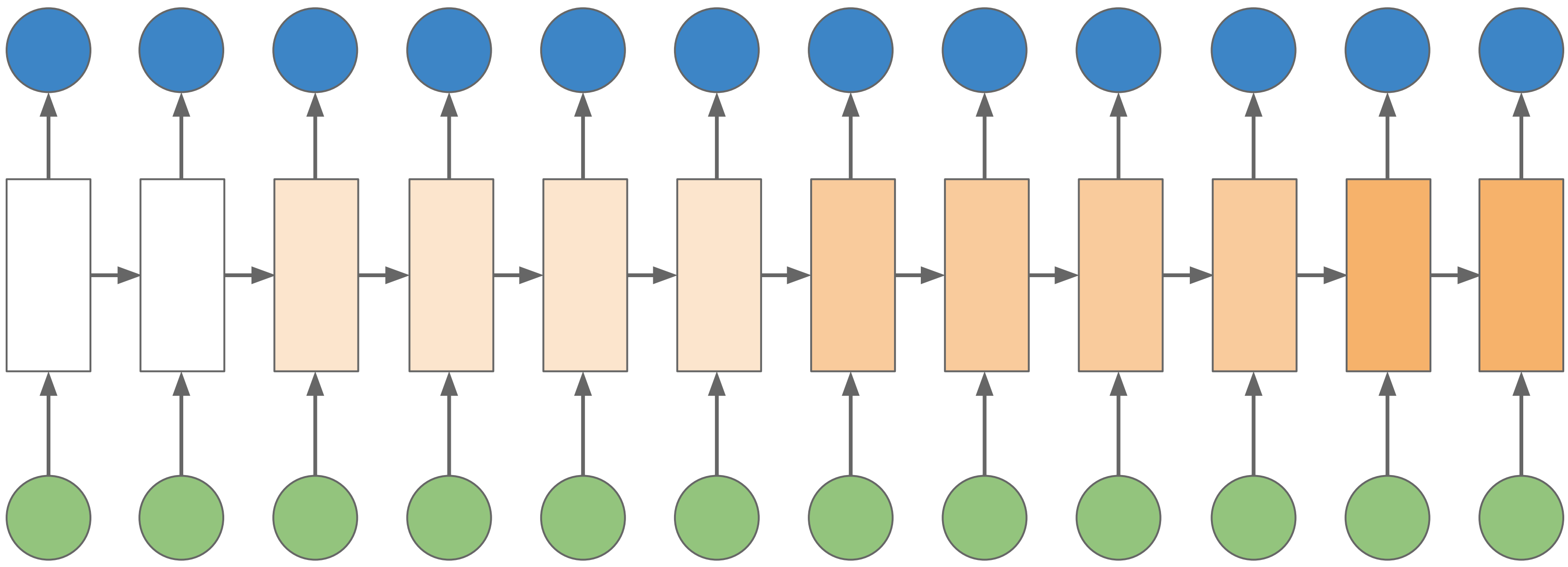
A bit difficult to see in the ***rolled*** RNN...

Issues with standard recurrent neural networks

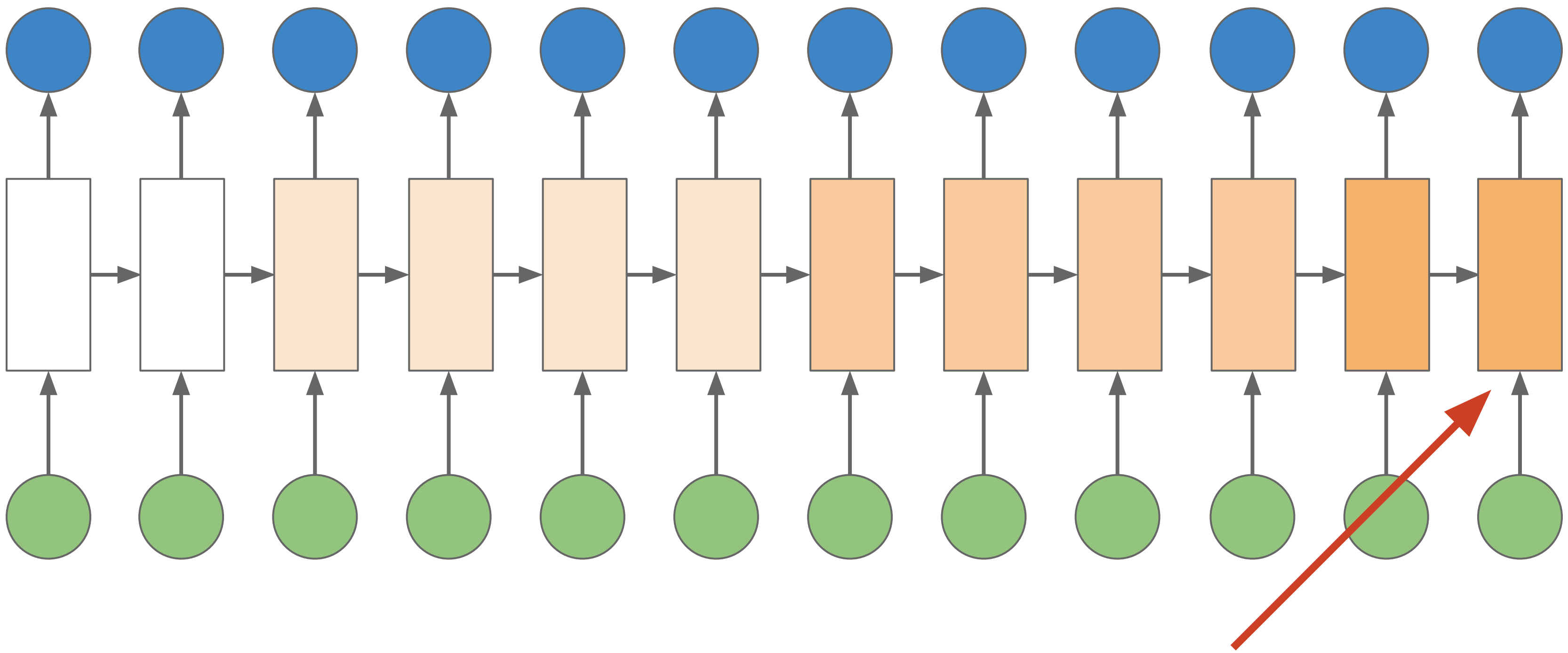
Issues with standard RNN’s

Information Decay

Information decay

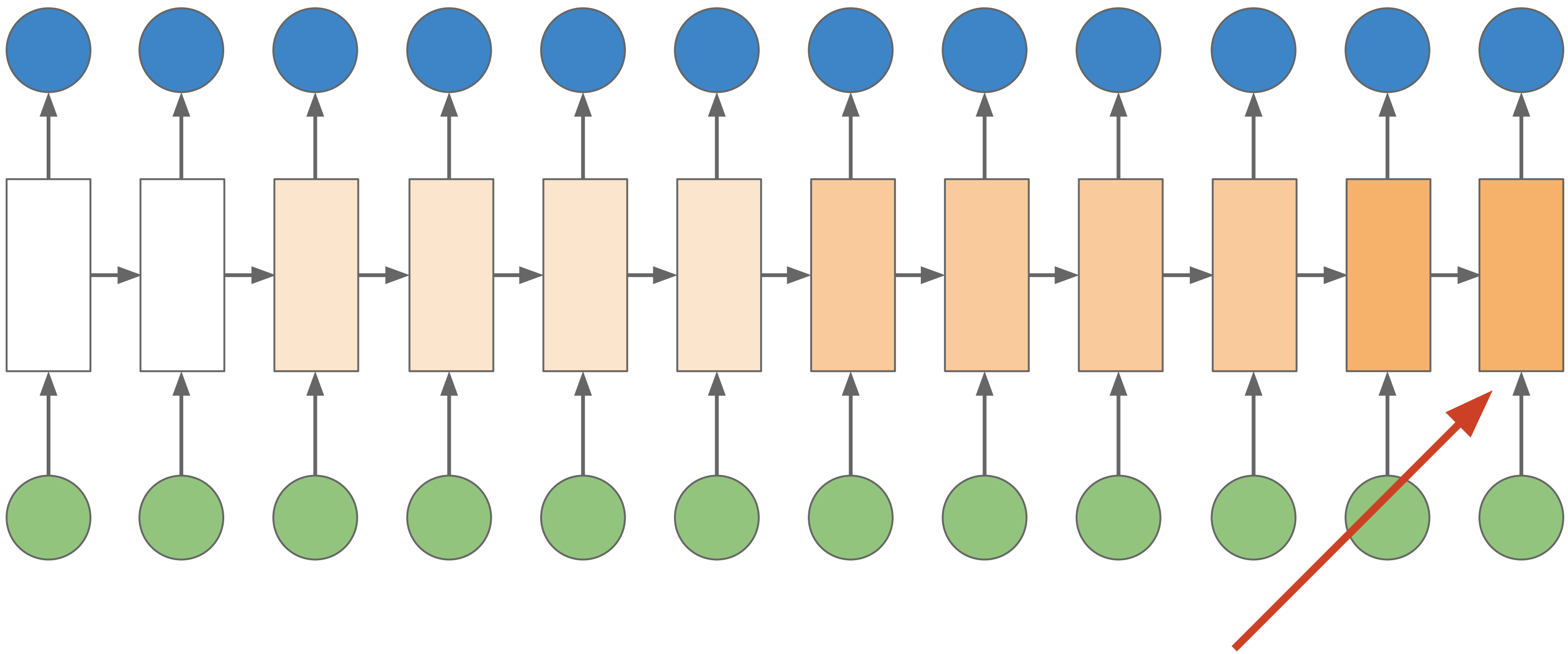


Information decay



The last timestep remembers very little about older timesteps, since it needs to remember information from recent history and the current timestep

Information decay



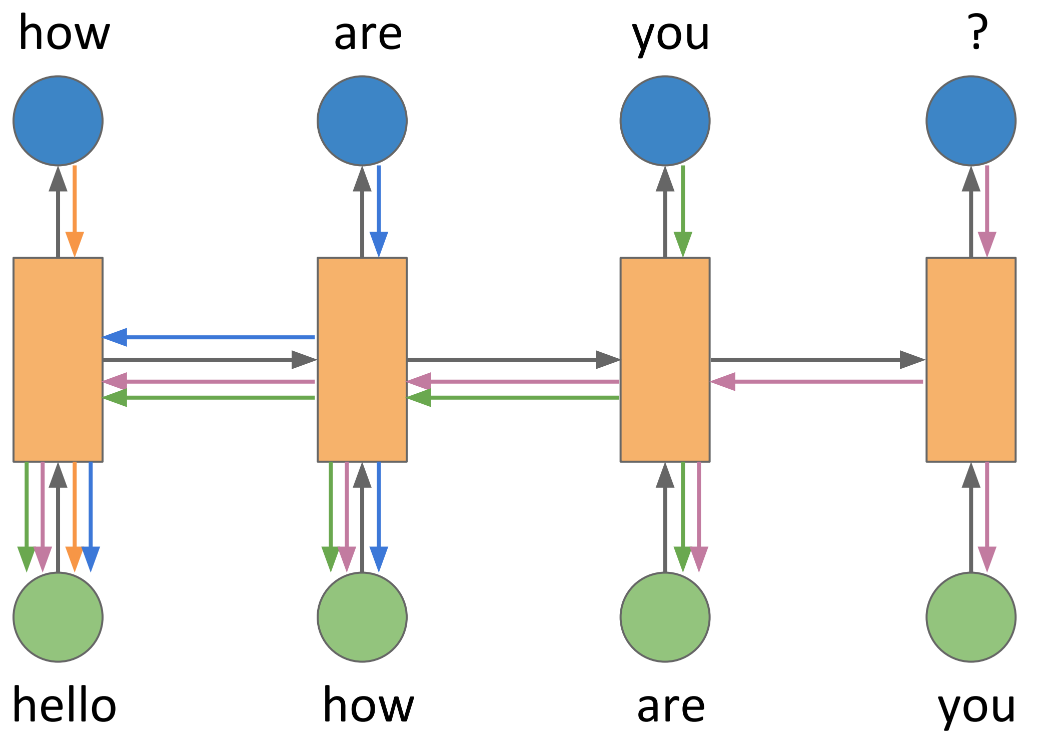
Remember, the output of the hidden layer is a **fixed length vector**

The network can only remember a limited amount of total information!

Issues with standard RNN’s

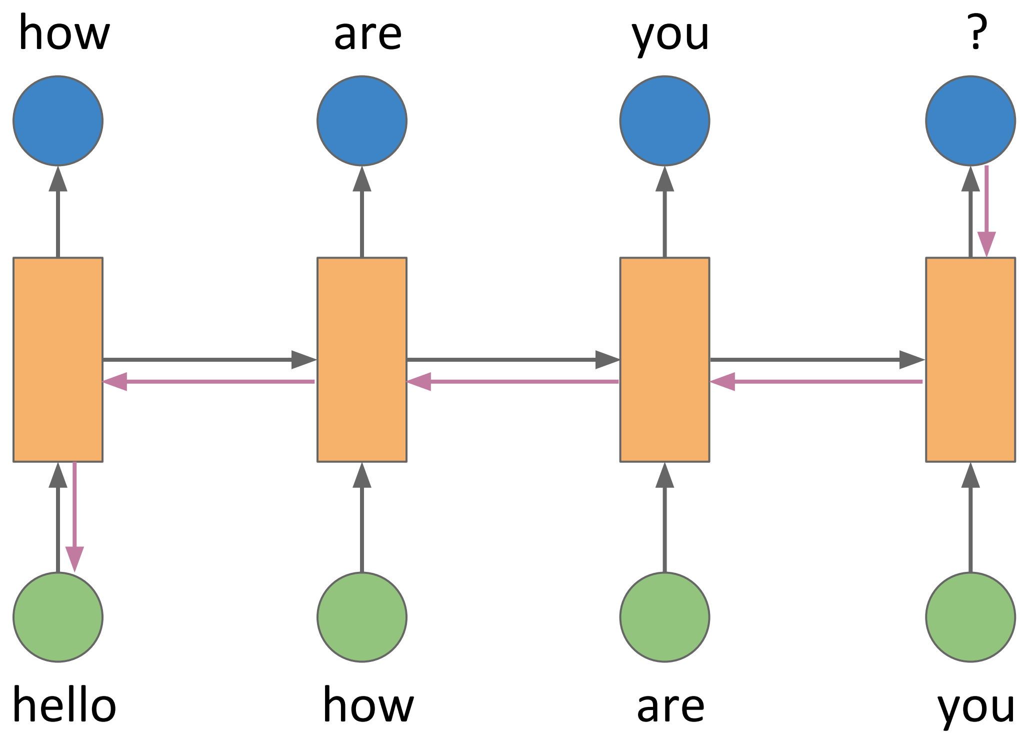
Vanishing Gradients

Vanishing Gradients



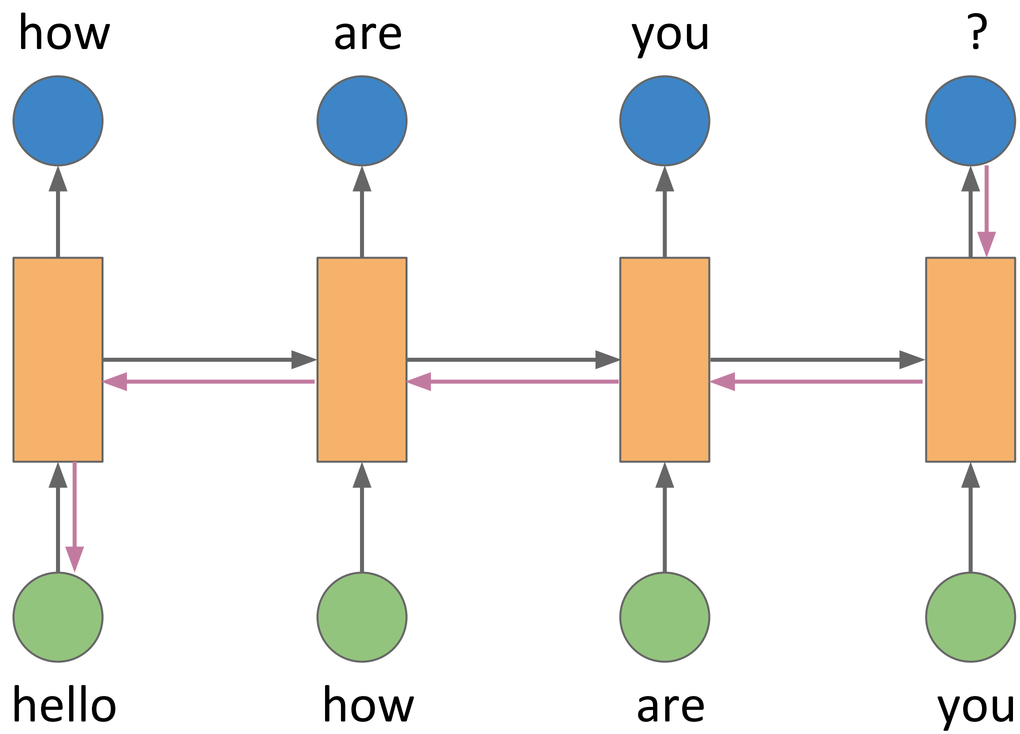
Consider the backpropagation through time

Vanishing Gradients



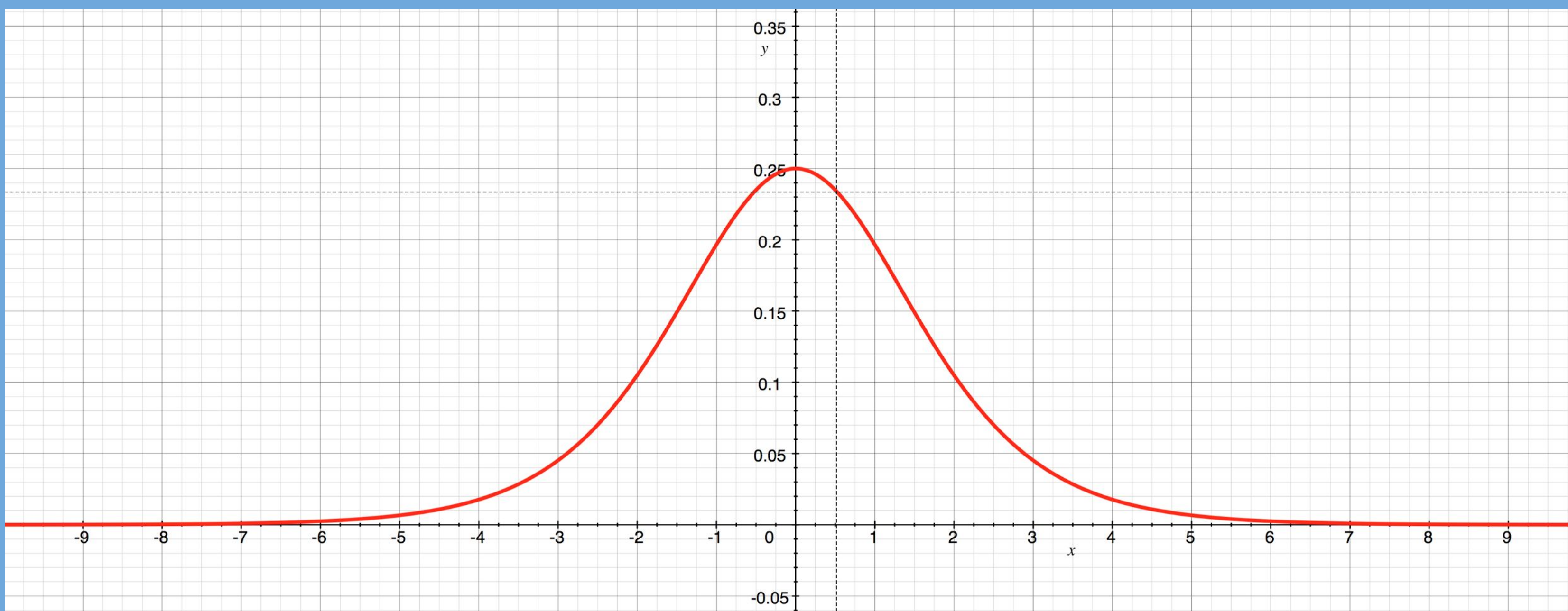
Consider the backpropagation through time

Vanishing Gradients



When the gradients passes through each time step, we have to multiply it with the derivative of the activation function

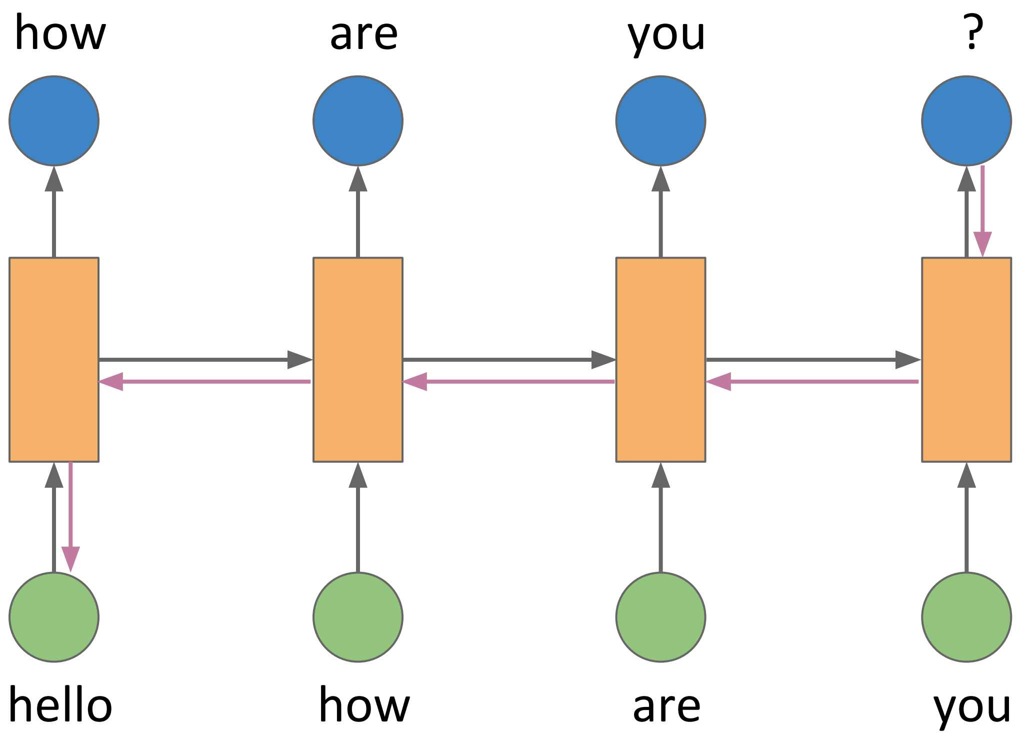
Vanishing Gradients



Gradient of sigmoid function

When the gradients passes through each time step, we have to multiply it with the derivative of the activation function

Vanishing Gradients



Since the gradient of the sigmoid activation is atmost `0.25`, we will be multiplying the gradient of the final timestep repeatedly by `0.25`

Vanishing Gradients

Since the gradient of the sigmoid activation is atmost `0.25`, we will be multiplying the gradient of the final timestep repeatedly by `0.25`

Vanishing Gradients

Since the gradient of the sigmoid activation is atmost `0.25`, we will be multiplying the gradient of the final timestep repeatedly by `0.25`

Multiplying several small numbers would result in even smaller numbers!

`0.25 x 0.25 x 0.25 = 0.0156`

Vanishing Gradients

Since the gradient of the sigmoid activation is atmost `0.25`, we will be multiplying the gradient of the final timestep repeatedly by `0.25`

Multiplying several small numbers would result in even smaller numbers!

`0.25 x 0.25 x 0.25 = 0.0156`

Similarly, our gradient from the final timestep becomes very small by the time it reaches a few steps in the beginning

Vanishing Gradients

Since the gradient of the sigmoid activation is atmost `0.25`, we will be multiplying the gradient of the final timestep repeatedly by `0.25`

Multiplying several small numbers would result in even smaller numbers!

`0.25 x 0.25 x 0.25 = 0.0156`

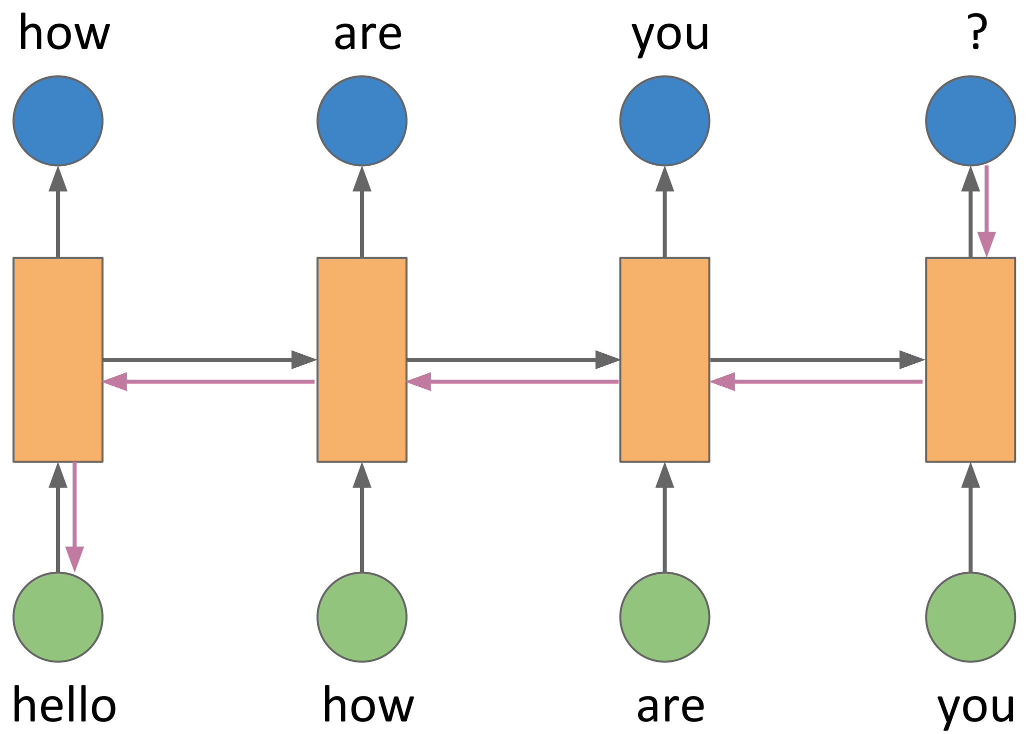
Similarly, our gradient from the final timestep becomes very small by the time it reaches a few steps in the beginning

Hence, our parameters do not change over long distances - but language has a lot of long range dependencies!

Issues with standard RNN’s

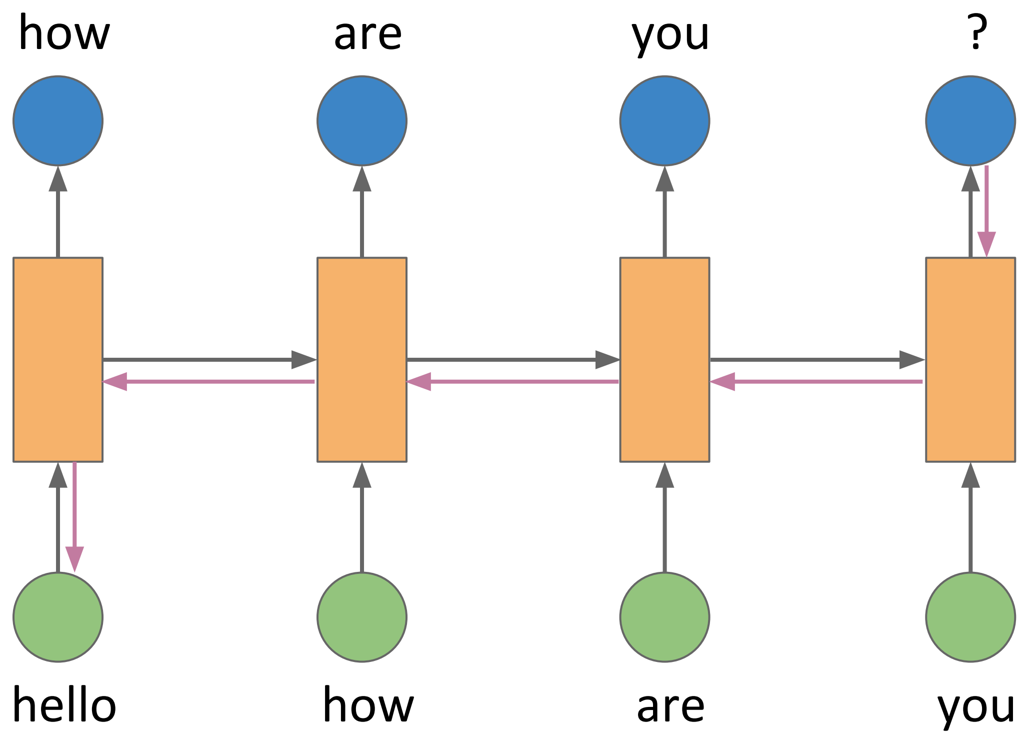
Exploding Gradients

Exploding Gradients



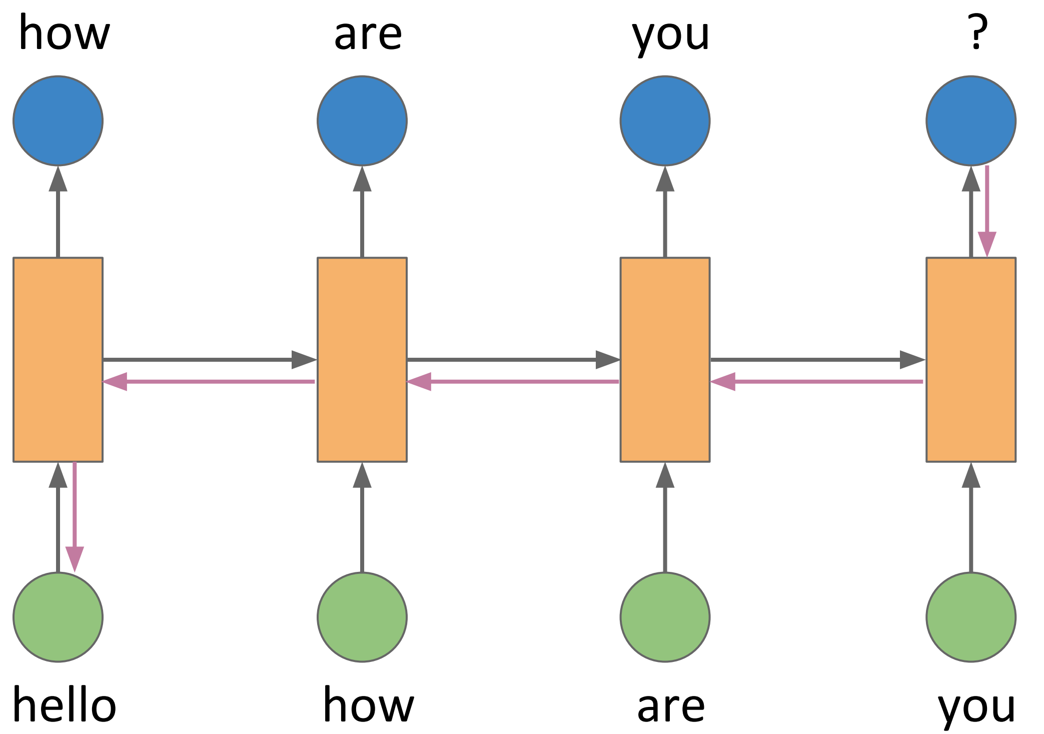
Similar to vanishing gradients, we can also have the problem of an **exploding gradient**

Exploding Gradients



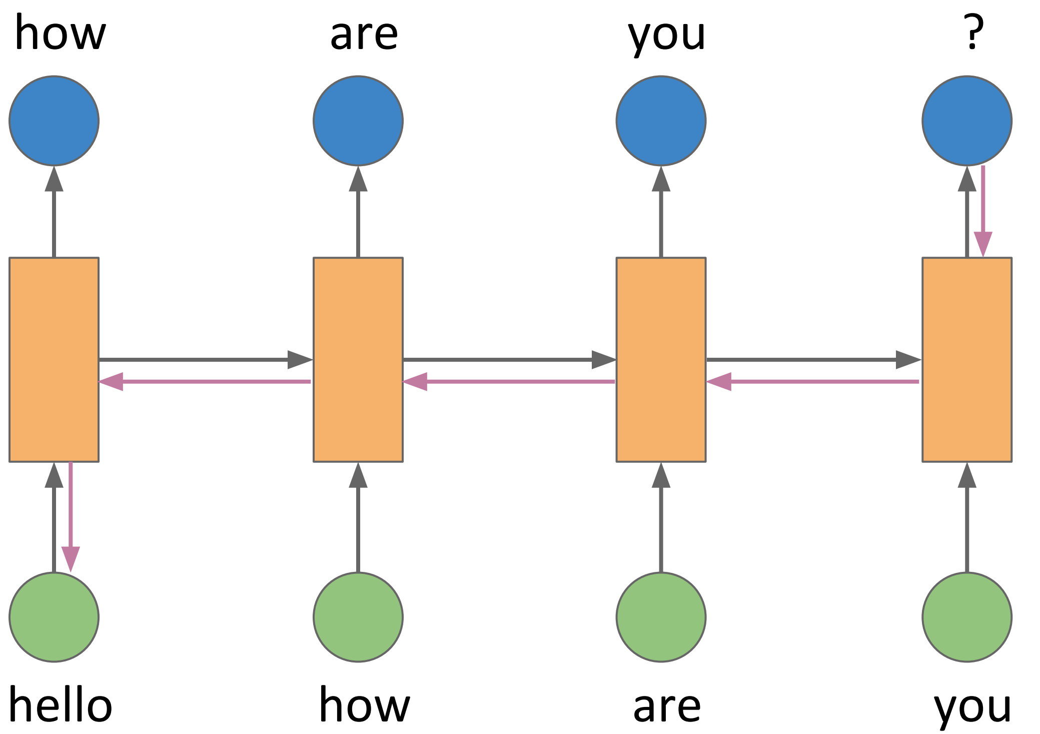
We have activation functions where the gradient can be greater than 1. Our weights themselves can also be greater than 1

Exploding Gradients



In very long sequences, multiplying a lot of large numbers can result in our gradient becoming too large very quickly!

Exploding Gradients



A lot of the time, the problem surfaces as our gradient becomes NaN, and so does our loss!

Issues with standard RNN’s

Long range dependency handling

Long-term Dependencies

In theory, RNNs are capable of remembering long distance information

Practically, they start forgetting information over long distances as we have seen with the information decay problem

Long-term Dependencies

Words can have long-term dependency on previous words

Ansehen

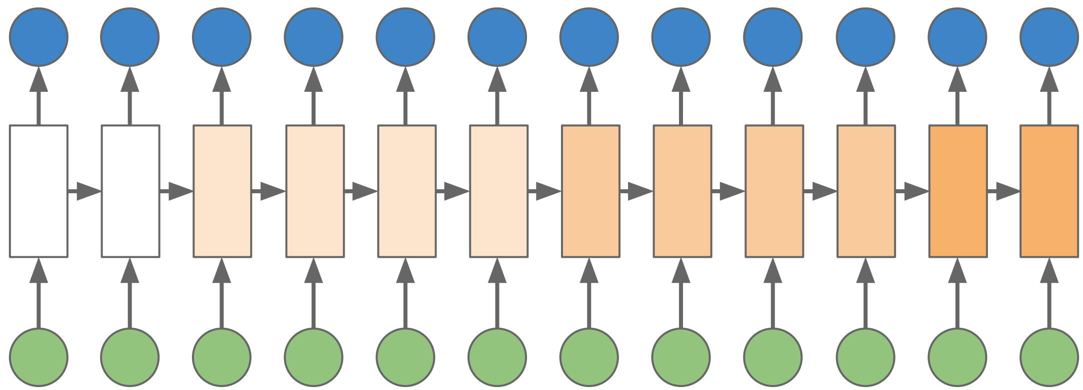
Anna ***sieht*** sich die Talkshow **an**

If the distance between "an" and "sieht" becomes long, the RNN may forget to correctly learn the relationship

Long-term Dependencies

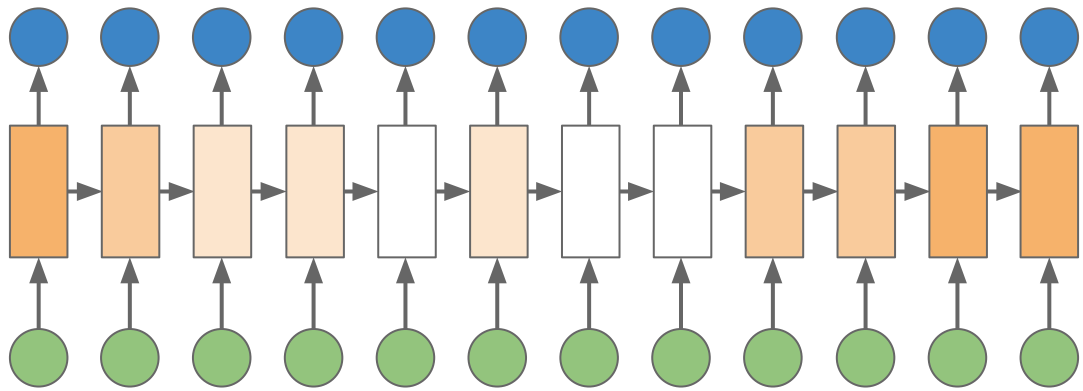
- In real world scenarios with inherited sequence properties, relevant information over long distances is vital
- For example, for an RNN to describe a movie scene, it would need to remember relevant information over longer sequences to describe the current scene correctly

Long-term Dependencies



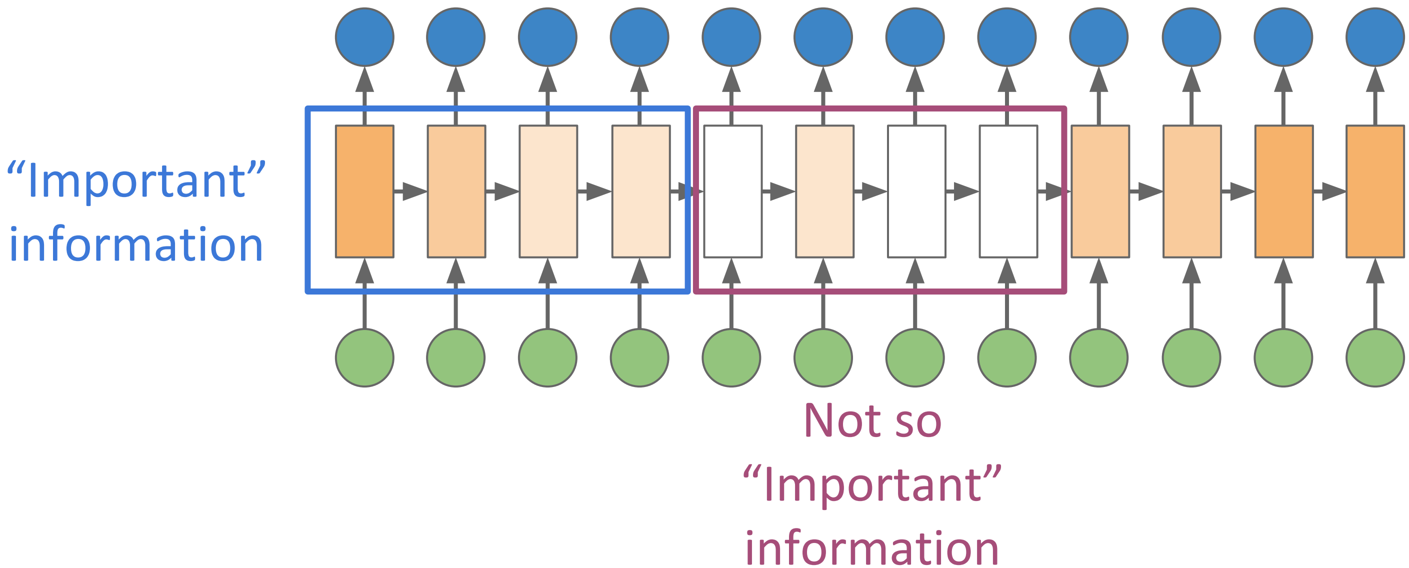
What if at each timestep, we can choose some information to be “important” and tell the network to remember it for longer?

Long-term Dependencies



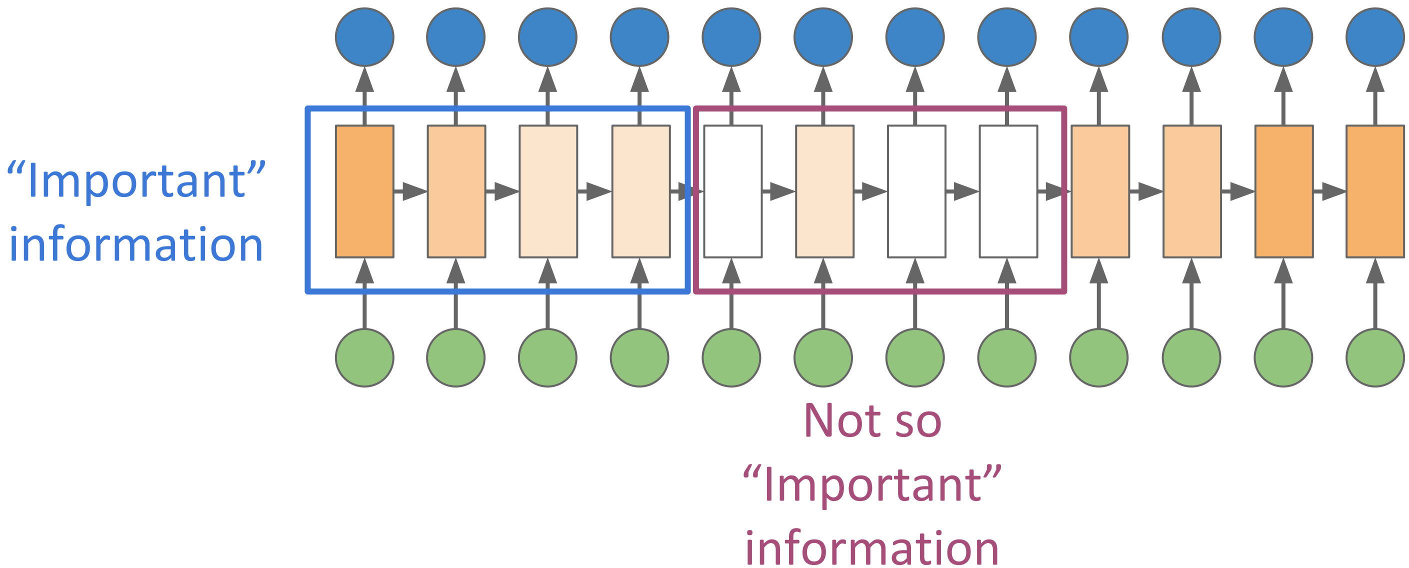
What if at each timestep, we can choose some information to be “important” and tell the network to remember it for longer?

Long-term Dependencies



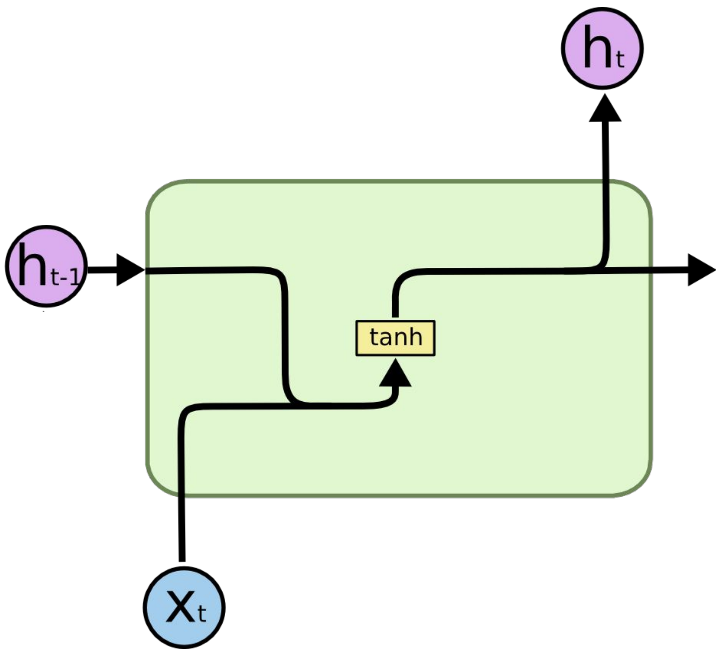
What if at each timestep, we can choose some information to be "important" and tell the network to remember it for longer?

Long-term Dependencies

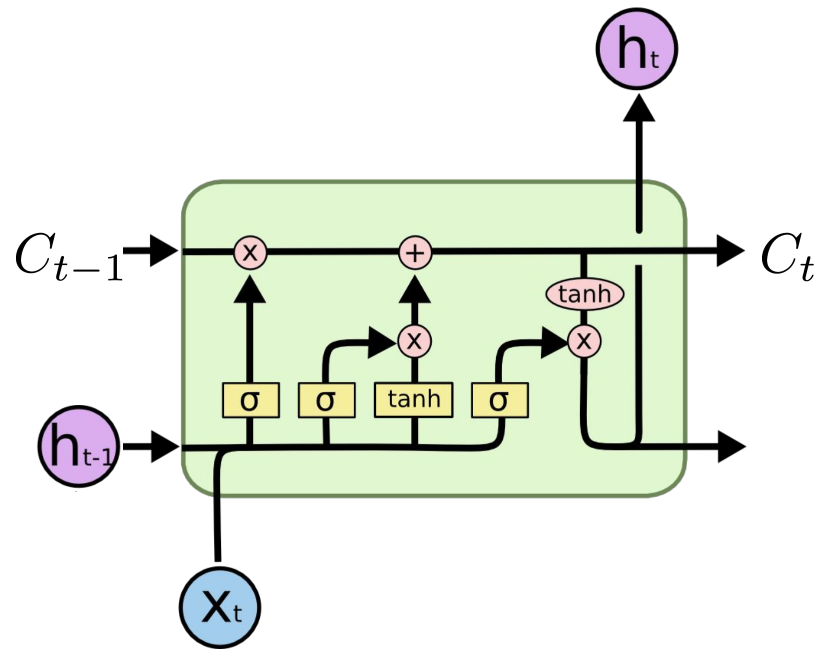


This is what **Long Short-term Memory** units do!

Long Short-term Memory



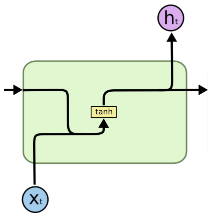
Recurrent
Unit

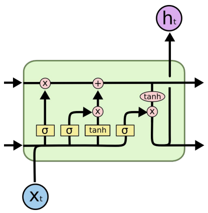


LSTM
Unit

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-term Memory





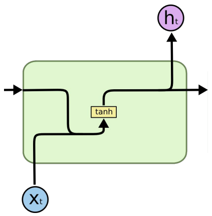
$$h_t = \tanh\left(Wx + W_h h_{t-1} + b\right)$$

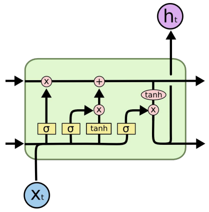
Recurrent Unit

LSTM Unit

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-term Memory





$$h_t = \tanh\left(Wx + W_h h_{t-1} + b\right)$$

Recurrent Unit

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\widetilde{C_t} = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$
$$C_t = f_t \cdot C_{t-1} + i_t \cdot \widetilde{C_t}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t \cdot \tanh(C_t)$$

LSTM Unit

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-term Memory

Intuition: We have a **“memory cell”** or **“cell state”** that is passed along the time steps. At each timestep, the unit decides to forget some information from this **cell** and add some new information from the current input!

Long Short-term Memory

Consider an example: We are building a language model over some text that has several different people.

Alice studies computational linguistics. She is currently learning about LSTM's. Bob on the other hand studies about cyber security. He is completely confused right now!

Long Short-term Memory

Consider an example: We are building a language model over some text that has several different people.

Alice studies computational linguistics. **She** is currently learning about LSTM's. Bob on the other hand studies about cyber security. He is completely confused right now!

Long Short-term Memory

Consider an example: We are building a language model over some text that has several different people.

Alice studies computational linguistics. She is currently learning about LSTM's. **Bob** on the other hand studies about cyber security. **He** is completely confused right now!

Long Short-term Memory

Alice studies computational linguistics. She is currently learning about LSTM's. Bob on the other hand studies about cyber security. He is completely confused right now!

Our LSTM see's the above text word-by-word, so it needs to remember who we are talking about to use the correct pronouns

Long Short-term Memory

Alice studies computational linguistics. She is currently learning about LSTM’s. Bob on the other hand studies about cyber security. He is completely confused right now!

LSTM unit sees **“Alice”** - and from the embeddings it knows that we are talking about a female person

Memory
Cell

Long Short-term Memory

Alice studies computational linguistics. She is currently learning about LSTM’s. Bob on the other hand studies about cyber security. He is completely confused right now!

LSTM unit sees **“Alice”** - and from the embeddings it knows that we are talking about a female person

Lets add this information to our **memory cell**!

P: Alice

Memory
Cell

Long Short-term Memory

Alice studies computational linguistics. She is currently learning about LSTM’s. Bob on the other hand studies about cyber security. He is completely confused right now!

LSTM unit see’s other words, and decides what information should go into the memory cell

Memory
Cell

Long Short-term Memory

Alice studies computational linguistics. She is currently learning about LSTM’s. Bob on the other hand studies about cyber security. He is completely confused right now!

LSTM unit see’s other words, and decides what information should go into the memory cell

P: Alice
⋮

Memory
Cell

Long Short-term Memory

Alice studies computational linguistics. She is currently learning about LSTM’s. Bob on the other hand studies about cyber security. He is completely confused right now!

LSTM unit now has to predict **“She”**. It does so by looking into the memory cell to identify which person we are talking about!

P: Alice

⋮

Memory
Cell

Long Short-term Memory

Alice studies computational linguistics. She is currently learning about LSTM's. Bob on the other hand studies about cyber security. He is completely confused right now!

LSTM unit see's other words, and decides what information should go into the memory cell

Memory Cell

Long Short-term Memory

Alice studies computational linguistics. She is currently learning about LSTM’s. Bob on the other hand studies about cyber security. He is completely confused right now!

LSTM unit see’s other words, and decides what information should go into the memory cell

P: Alice
⋮

Memory
Cell

Long Short-term Memory

Alice studies computational linguistics. She is currently learning about LSTM's. Bob on the other hand studies about cyber security. He is completely confused right now!

LSTM unit see's other words, and decides what information should go into the memory cell

Memory
Cell

Long Short-term Memory

Alice studies computational linguistics. She is currently learning about LSTM's. Bob on the other hand studies about cyber security. He is completely confused right now!

LSTM unit sees **"Bob"** - and from the embeddings it knows that we are talking about a male person

Lets *update* this information in our **memory cell**!

P: Bob

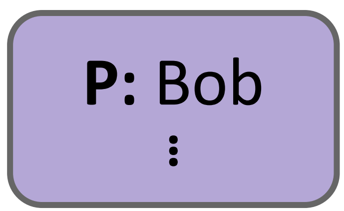
⋮

Memory
Cell

Long Short-term Memory

Alice studies computational linguistics. She is currently learning about LSTM’s. Bob on the other hand studies about cyber security. He is completely confused right now!

LSTM unit now has to predict **“He”**. Again, it does so by looking into the memory cell to identify which person we are talking about!



Memory
Cell

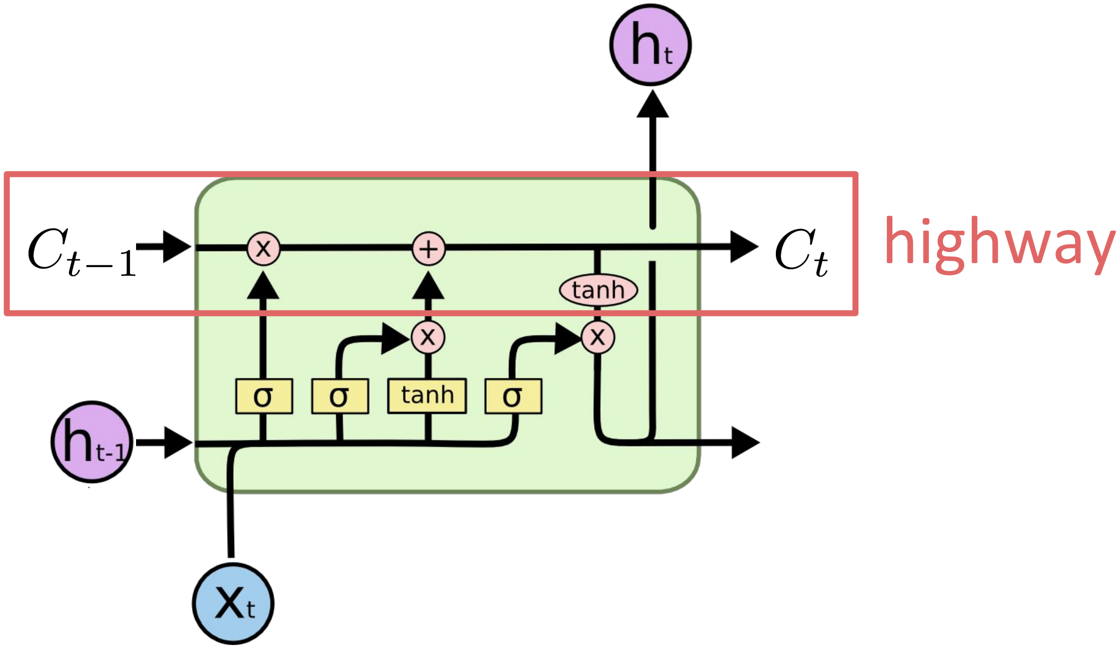
Long Short-term Memory

Intuition: We have a **"memory cell"** or **"cell state"** that is passed along the time steps. At each timestep, the unit decides to forget some information from this **cell** and add some new information from the current input!

This effectively helps us solve both the **Information decay** and the **vanishing gradient** problem

Long Short-term Memory

This effectively helps us solve both the **Information decay** and the **vanishing gradient** problem



LSTM demonstrations

Long Short-term Memory

LSTM’s are super general purpose - you can use them on any kind of sequential data that can be represented as some vectors

Long Short-term Memory

LSTM’s are super general purpose - you can use them on any kind of sequential data that can be represented as some vectors

Andrej Karpathy has some really nice demos on character-level language models, i.e. the input to the network is the **next character** instead of a word

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

LSTM evolution

Here, we will see an network using LSTM units *evolve* over time - remember, all we are doing here is asking the network to predict the next character given the history of characters

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

LSTM evolution

Here, we will see an network using LSTM units *evolve* over time - remember, all we are doing here is asking the network to predict the next character given the history of characters

qewjhhcvuh rkjghqruhgqrgnqrhlhqlrqrjlcznmcyaklm adjfadhoirqjnrm, aghouihr;qnrjnjn agyqeg cvz,cmnv;lhruhm,.nm,czbvugrablgjn,.m adnadfnalkd

Iteration 0

Initially, the output is complete garbage!

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

LSTM evolution

Here, we will see an network using LSTM units *evolve* over time - remember, all we are doing here is asking the network to predict the next character given the history of characters

> tyntd-iafhatawiaoihrdemot lytdws e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee eplia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng

Iteration 100

A few iterations later - still garbage, but it is starting to learn the concept of "words" and "spaces"

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

LSTM evolution

Here, we will see an network using LSTM units *evolve* over time - remember, all we are doing here is asking the network to predict the next character given the history of characters

"Tmont thithey" fomesscerliund Keushey. Thom here sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."

Iteration 300

The model is now learning about periods and quotes.

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

LSTM evolution

Here, we will see an network using LSTM units *evolve* over time - remember, all we are doing here is asking the network to predict the next character given the history of characters

> we counter. He stutn co des. His stanted out one ofler that concossions and was to gearang reay Jotrets and with fre colt otf paitt thin wall. Which das stimn

Iteration 500

Some simple words like "We", "He", "His" are spelt correctly!

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

LSTM evolution

Here, we will see an network using LSTM units *evolve* over time - remember, all we are doing here is asking the network to predict the next character given the history of characters

> Aftair fall unsuch that the hall for Prince Velzonski's that me of her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort how, and Gogition is so overelical and ofter.

Iteration 700

Some structure of English is starting to appear...

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

LSTM evolution

Here, we will see an network using LSTM units *evolve* over time - remember, all we are doing here is asking the network to predict the next character given the history of characters

"Kite vouch!" he repeated by her
door. "But I would be done and quarts, feeling, then, son is people...."

Iteration 1200

The model has learned longer words and some punctuation

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

LSTM evolution

Here, we will see an network using LSTM units *evolve* over time - remember, all we are doing here is asking the network to predict the next character given the history of characters

> "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.

Iteration 2000

Much better outputs than what we started with...

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

LSTM evolution

Here, we will see an network using LSTM units *evolve* over time - remember, all we are doing here is asking the network to predict the next character given the history of characters

"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.

Iteration 2000

Much better outputs than what we started with...

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

LSTM examples

A model trained on Shakespeare

PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

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

LSTM examples

A model trained on Wikipedia sources

Naturalism and decision for the majority of Arab countries' capitalide was grounded by the Irish language by **[[John Clair]]**, **[[An Imperial Japanese Revolt]]**, associated with Guangzham's sovereignty. His generals were the powerful ruler of the Portugal in the [[Protestant Immineners]], which could be said to be directly in Cantonese Communication, which followed a ceremony and set inspired prison, training. The emperor travelled back to [[Antioch, Perth, October 25|21]] to note, the Kingdom of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known in western [[Scotland]], near Italy to the conquest of India with the conflict. Many governments recognize the military housing of the [[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]], that is sympathetic to be to the [[Punjab Resolution]] (PJS) [**http://www.humah.yahoo.com/guardian.cfm/7754800786d17551963s89.htm** Official economics Adjoint for the Nazism, Montgomery was swear to advance to the resources for those Socialism's rule, was starting to signing a major tripad of aid exile.]]

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

LSTM examples

A book on Algebraic geometry!

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, **??** and the fact that any U affine, see Morphisms, Lemma **??**. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma **??** we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example **??**. It may replace S by $X_{spaces,\acute{e}tale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma **??**. Namely, by Lemma **??** we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ (by the formal open covering X and a single map $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ over U compatible with the complex

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

*When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition **??** (without element is when the closed subschemes are catenary. If T is surjective we may assume that T is connected with residue fields of S. Moreover there exists a closed subspace $Z \subset X$ of X where U in X' is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem*

(1) *f is locally of finite type. Since $S = \mathrm{Spec}(R)$ and $Y = \mathrm{Spec}(R)$.*

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{\mathcal{X},\ldots,0}$.

Lemma 0.2. *Let X be a locally Noetherian scheme over S, $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}^n \subset \mathcal{I}^n$ are nonzero over $i_0 \leq \mathfrak{p}$ is a subset of $\mathcal{J}_{n,0} \circ \overline{A}_2$ works.*

Lemma 0.3. *In Situation **??**. Hence we may assume $\mathfrak{q}' = 0$.*

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (**??**). On the other hand, by Lemma **??** we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

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

LSTM examples

A model trained on Linux source code

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    else
      ret = 1;
    goto bail;
  }
  segaddr = in_SB(in.addr);
  selector = seg / 16;
  setup_works = true;
  for (i = 0; i < blocks; i++) {
    seq = buf[i++];
    bpf = bd->bd.next + i * search;
    if (fd) {
      current = blocked;
    }
  }
  rw->name = "Getjbbregs";
  bprm_self_clearl(&iv->version);
  regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
  return segtable;
}
```

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

Exploding Gradients

Q: We now have a nice solution to take care of vanishing gradients, but what about exploding gradients?

Exploding Gradients

Q: We now have a nice solution to take care of vanishing gradients, but what about exploding gradients?

A: Just clip the gradients to some value at each step, say ± 5

Works very well in practice - intuitively we are just forcing the model to limit its updates to some maximum allowable value

Summary

- Neural Network Language models
- Word Embeddings
- Recurrent Neural Networks
  - Backpropagation through time
- Vanishing/Exploding gradients
- Long Short-term Memory units