Recurrent Neural Network

Sequential data handling

Insop, 5/5

Agenda

Review

Sequential data

Words vector representation & embedding
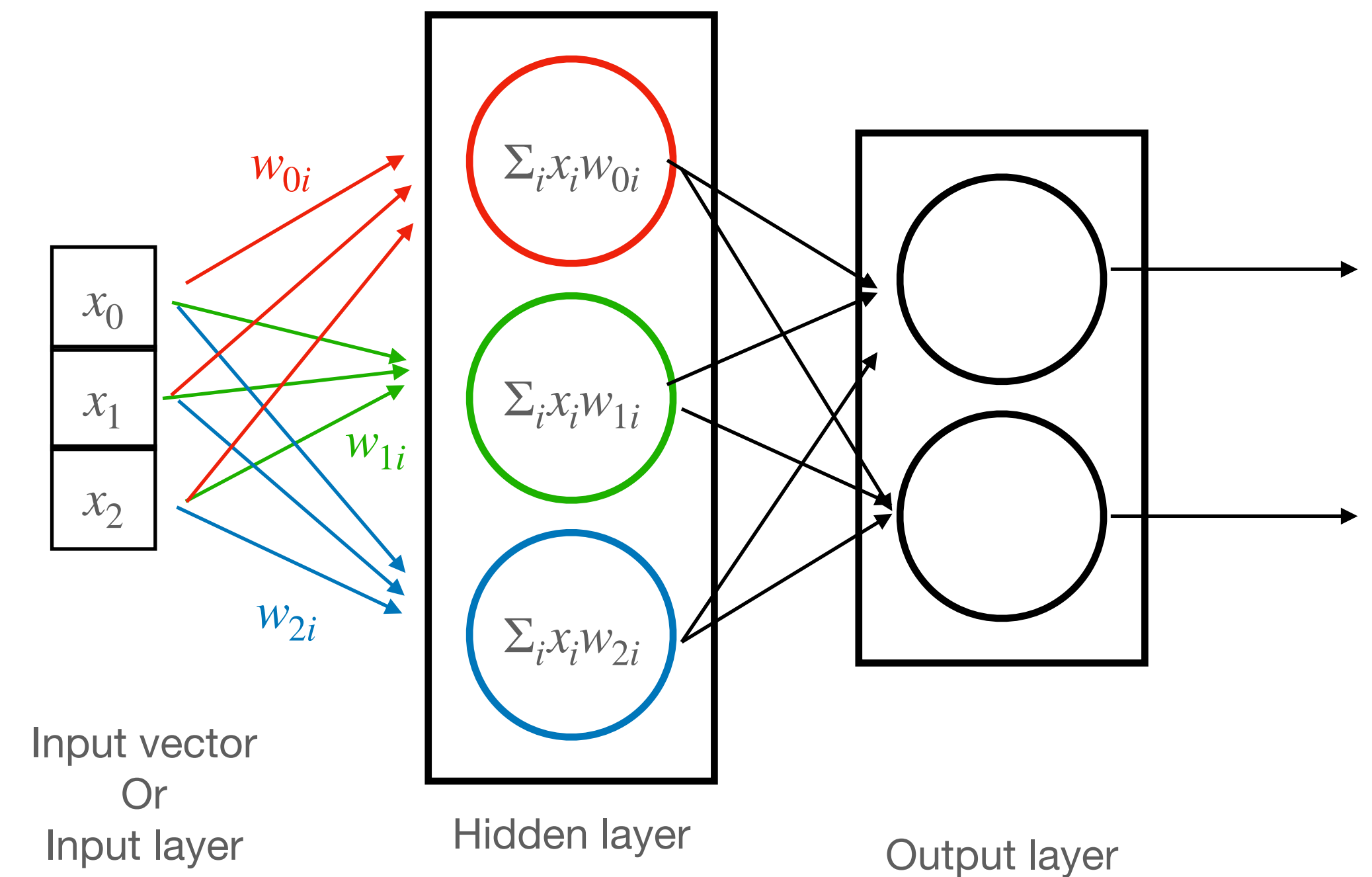
Recurrent neural network (RNN)

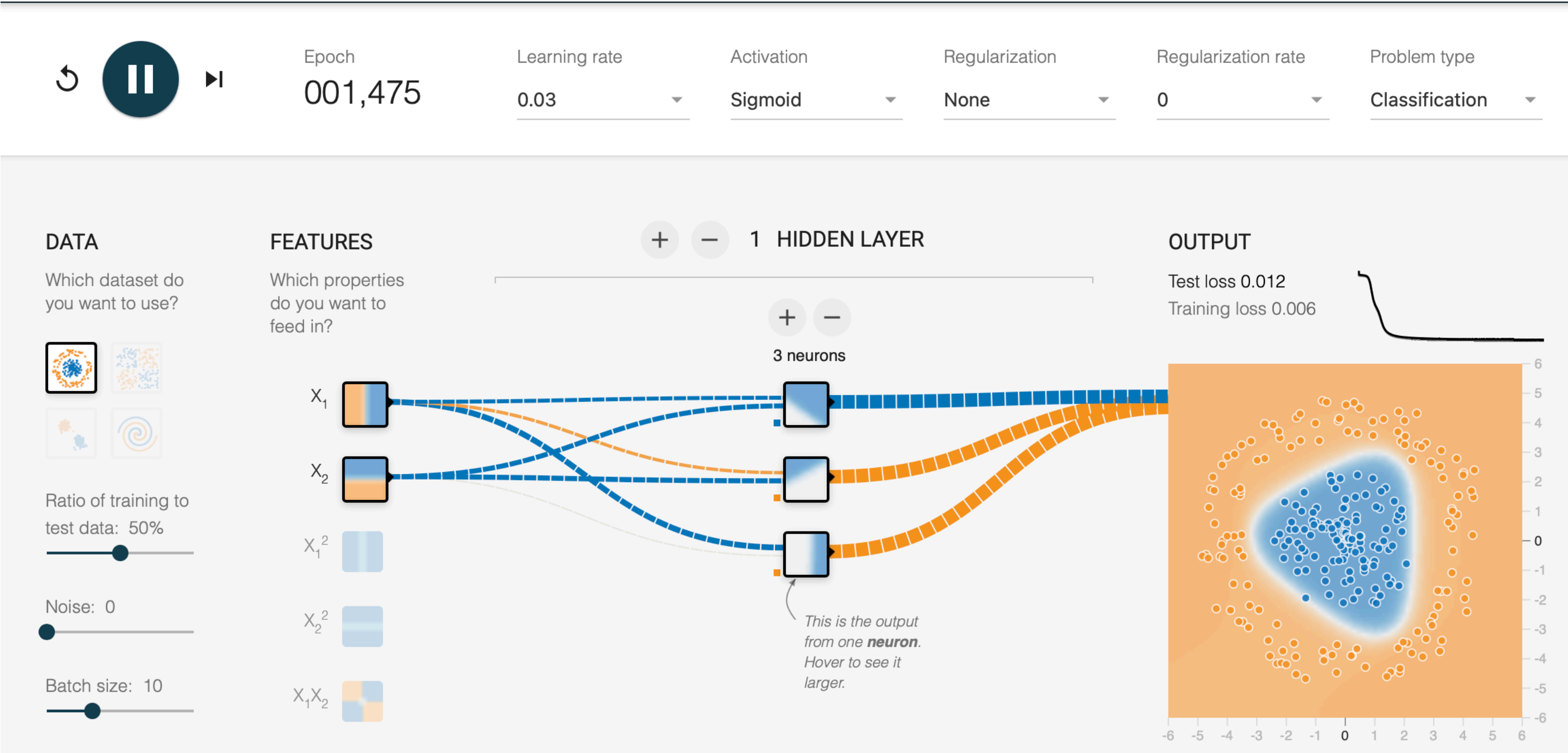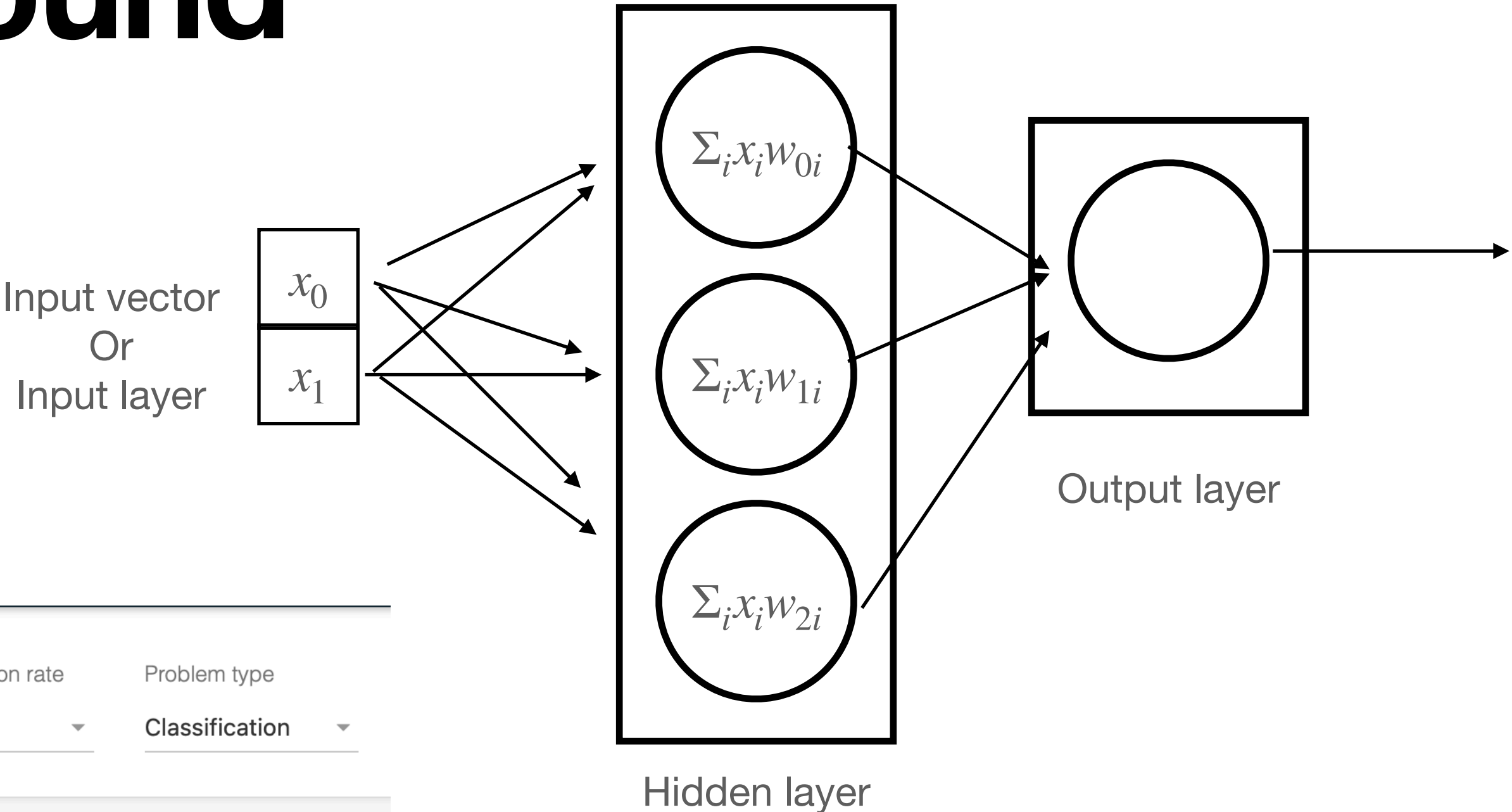Applications

Fancy RNNs

Summary

Review

Neural network

- Layer: multiple neurons

- Multiple layers: connecting many layers

- Neurons in each layers are connected

  - Also called FC (fully connected) layers or MLP (multi-layer perceptron)

Input vector
Or
Input layer

Hidden layer

Output layer

$$\begin{bmatrix} w_{00}x_0 + w_{01}x_1 + w_{02}x_2 \\ w_{10}x_0 + w_{11}x_1 + w_{12}x_2 \\ w_{20}x_0 + w_{21}x_1 + w_{22}x_2 \end{bmatrix} = \begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

Neural network playground

- Great way to gain intuition

- http://playground.tensorflow.org

Input vector
Or
Input layer

x_0

x_1

$\Sigma_i x_i w_{0i}$

$\Sigma_i x_i w_{1i}$

$\Sigma_i x_i w_{2i}$

Hidden layer

Output layer

Simple neural network example

- Simple 3 layers neural network

- In each layer:

  - Input: vector

  - Output: vector

  - Learned parameters: matrix

  - Operations:

    - Multiply the input vector with weight matrix

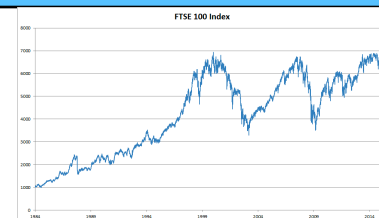   - Apply element-wise non-linear operations: ReLU *

Input vector
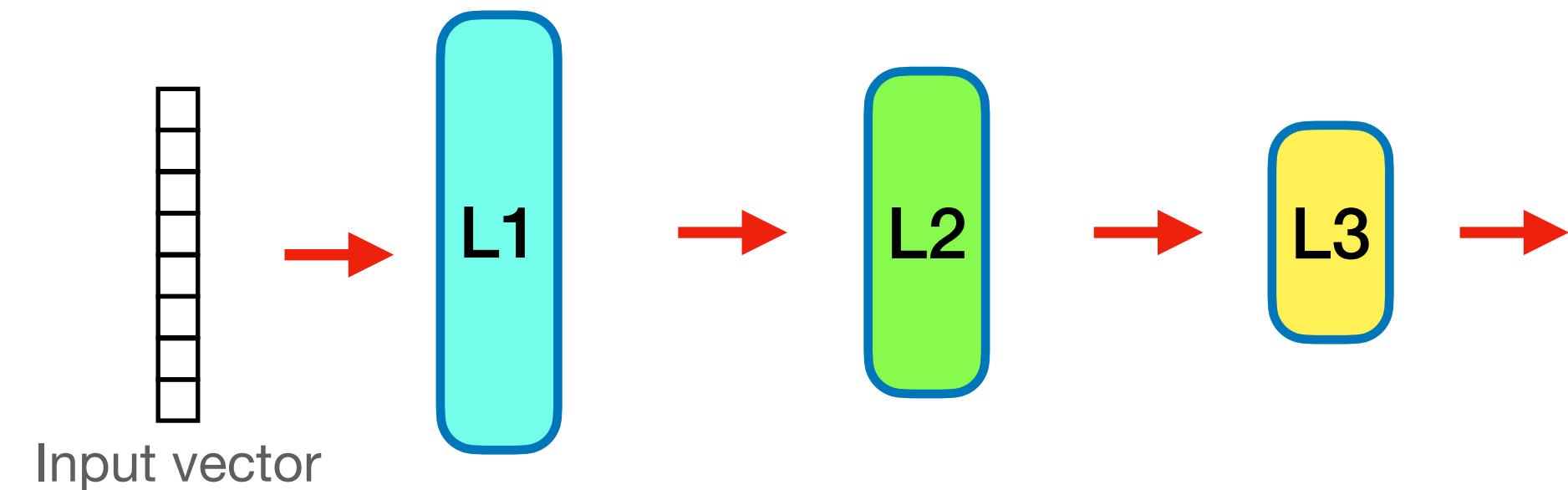
- Neural network training

  - Forward propagation

  - Backward propagation

  - Weight update

*Other types of non-linear ops: sigmoid, tanh

Sequential data

Sequential problems

- There are many applications requires sequential data processing

- Sequential data: data at time t has relationship with previous time steps

- Various types of input and outputs used in sequential applications

- A regular neural network might not use the sequential nature of the input data

- A neural network might not generate sequential outputs

Types	Input	Output
Time series forecasting	Time series	Next time step prediction
Sentiment analysis	movie review	Predicted sentiment
Language translation	English text	日本語　普通话　한국어
Speech recognition		Transcribed text
Document summarization	Document text	Summarized text
Image captioning		Describing scene, planes flying
Question answering	what is the tallest building in the world?	Shanghai Tower

Adapted from : "RNNs" section of Full Stack Deep Learning

Sequential problems

- Problem setup of sequential problems

- One to many:

  - Image captioning

- Many to one:

  - Sentiment analysis, time series forecast

- Many to many:

  - Machine translation

one to many   many to one   many to many

Input sequence vectors
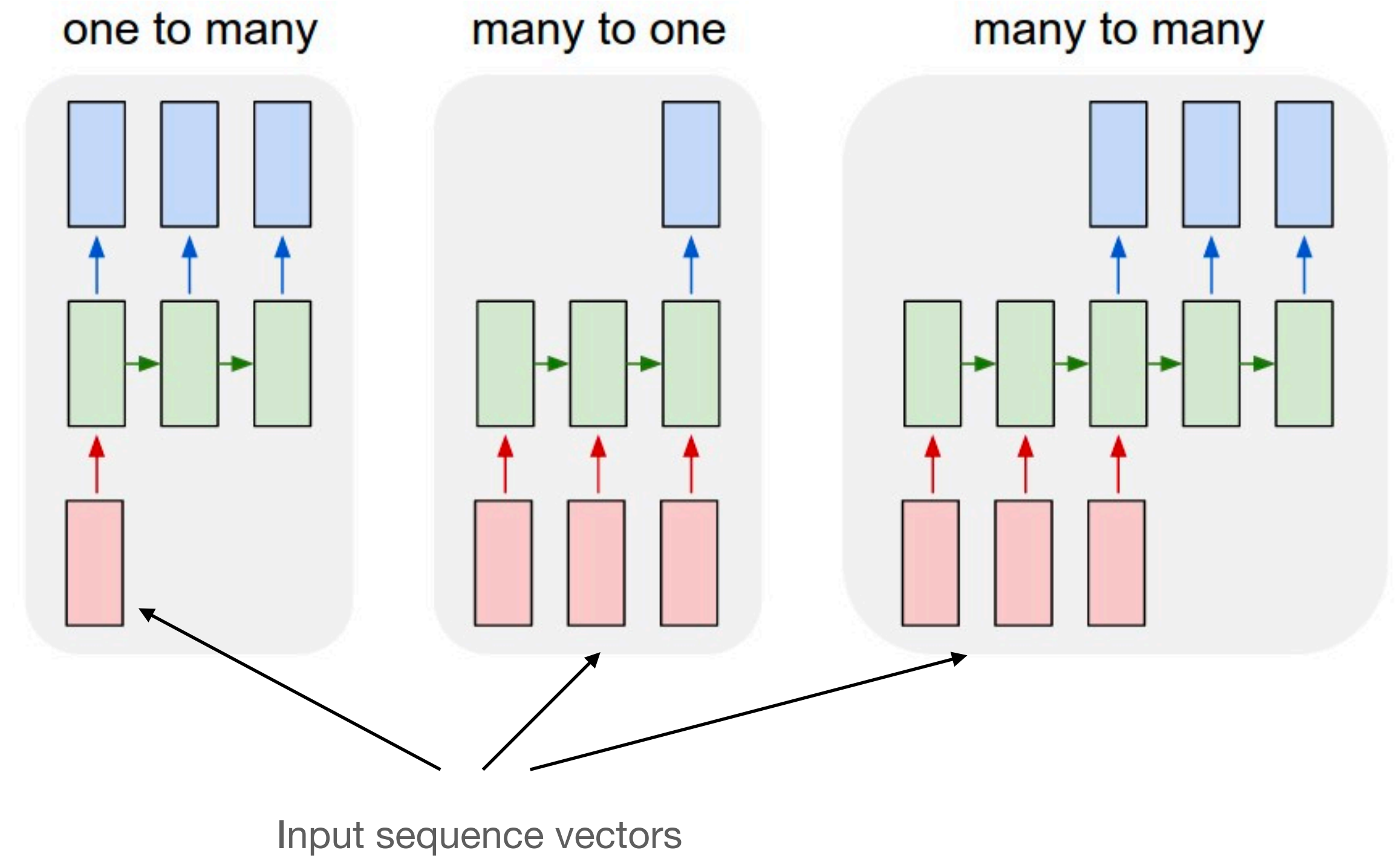
Words vector representation & embedding

Text and one-hot encoding

Input needs to be vector

- Text

- Tokenization:

- Vocabulary: fixed sized `dictionary`

- Token (word): can be converted to number using the vocabulary

- One hot encoding, all 0s, except one 1

```
the time machine by h g wells
```

```
['the', 'time', 'machine', 'by', 'h', 'g', 'wells']
```

```
[1, 19, 50, 40, 2183, 2184, 400]
```

```
Tokens example with vocab size 10 :[1, 5, 9, 3]
```

```
One-hot encodings of [1, 5, 9, 3]

[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
```

Vocabulary and one-hot vector

- Define vocabulary with index

Vocabulary

index:	word:
0	aardvark
1	able
...	...
2409	black
2410	bling
...	...
3202	candid
3203	cast
3204	cat
...	...
5281	is
5282	island
...	...
8676	the
8677	thing
...	...
9999	zombie

10,000 words with indices

Vocabulary and one-hot vector
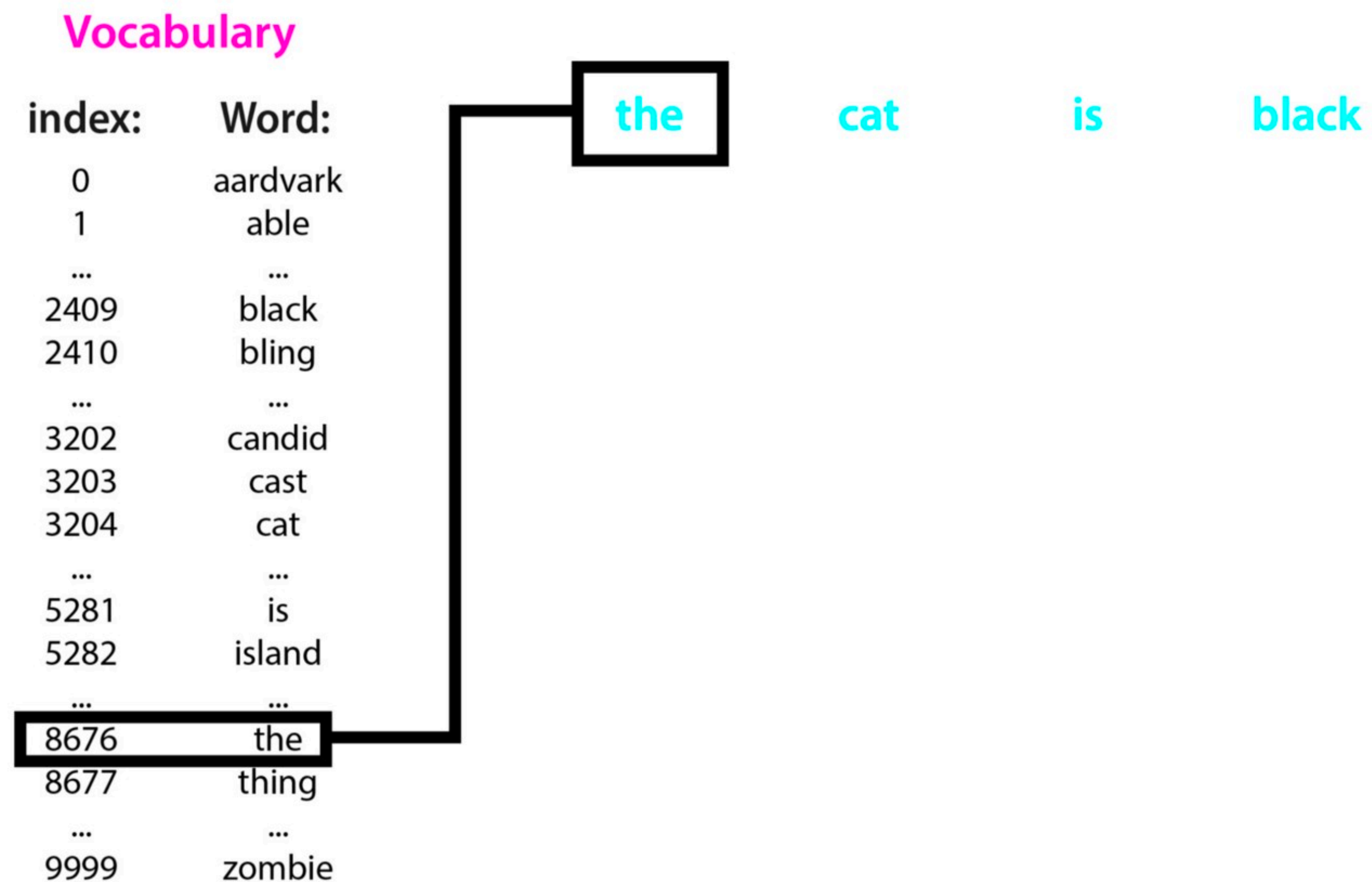
- Define vocabulary with index

Vocabulary

| index:	Word:
0	aardvark
1	able
...	...
2409	black
2410	bling
...	...
3202	candid
3203	cast
3204	cat
...	...
5281	is
5282	island
...	...
8676	the
8677	thing
...	...
9999	zombie

the   cat   is   black
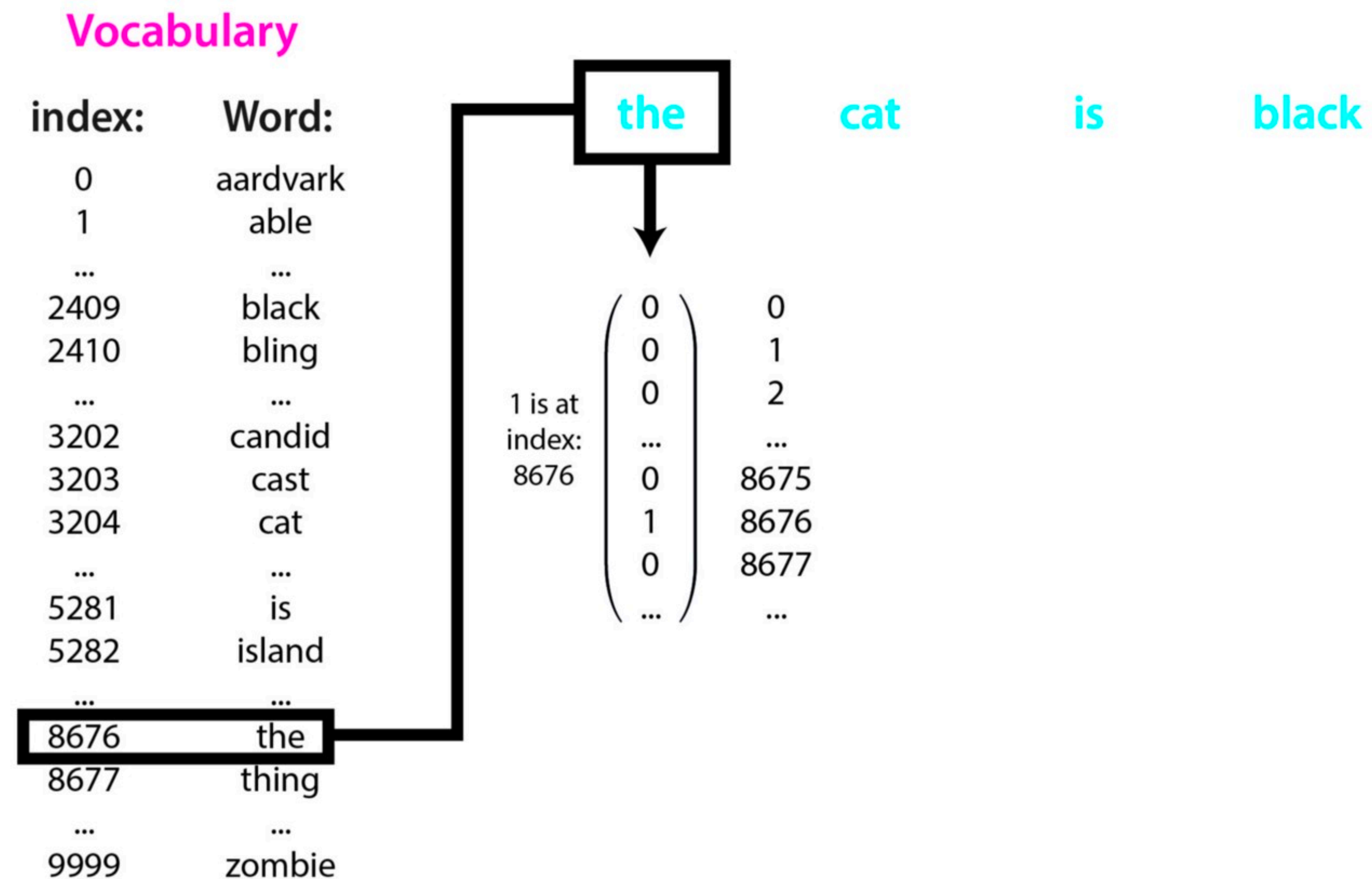
Vocabulary and one-hot vector

- Define vocabulary with index

- Represent the word as one-hot vector based on vocabulary index

Vocabulary

index:	Word:
0	aardvark
1	able
...	...
2409	black
2410	bling
...	...
3202	candid
3203	cast
3204	cat
...	...
5281	is
5282	island
...	...
8676	the
8677	thing
...	...
9999	zombie

the   cat   is   black

1 is at index: 8676

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ ... \\ 0 \\ 1 \\ 0 \\ ... \end{pmatrix} \quad \begin{matrix} 0 \\ 1 \\ 2 \\ ... \\ 8675 \\ 8676 \\ 8677 \\ ... \end{matrix}$$

Vocabulary and one-hot vector

- Define vocabulary with index

- Represent the word as one-hot vector based on vocabulary index
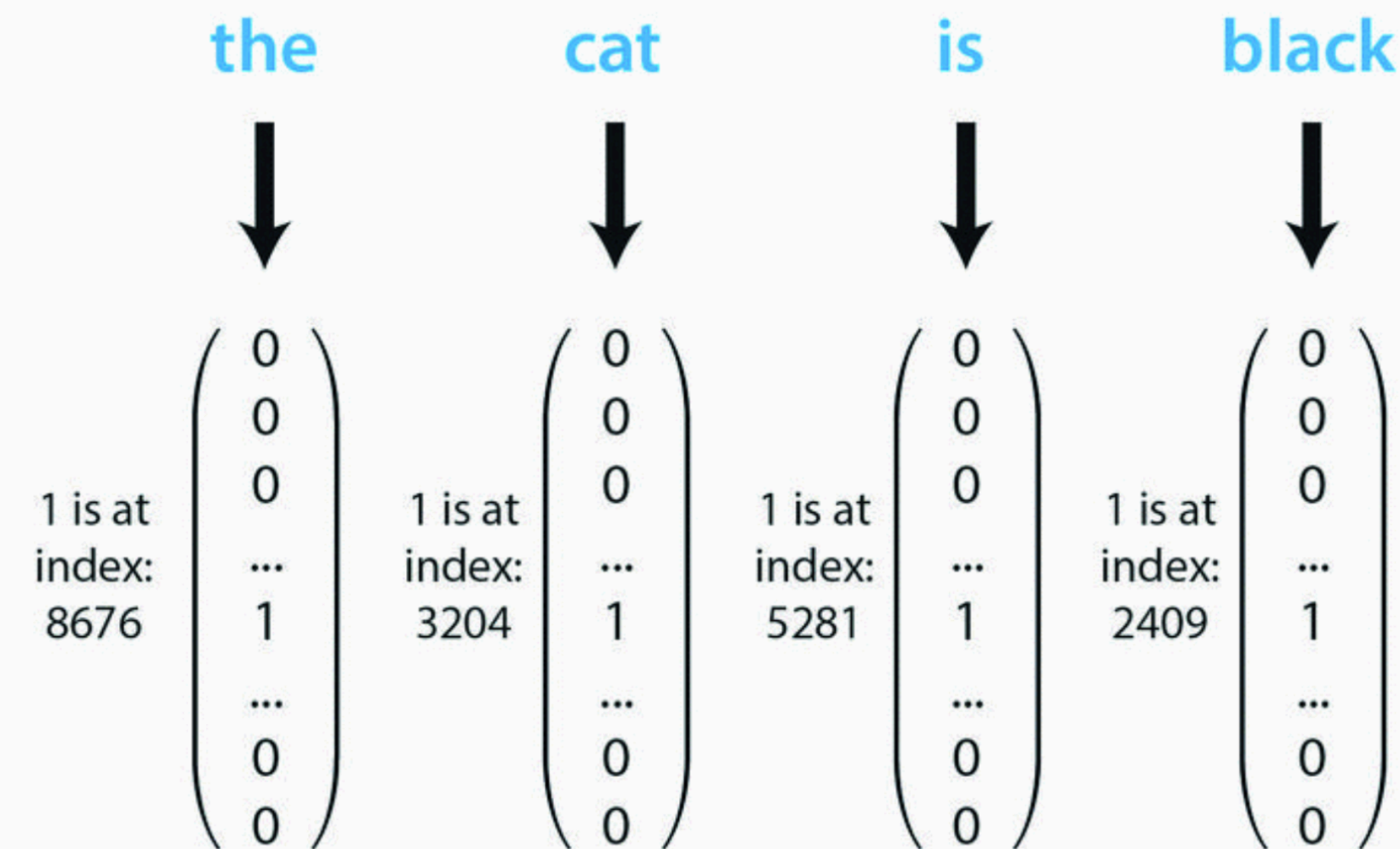
One-hot encoding

- **Issue** of one-hot encoding

  - Using this discrete encoding, hard to find the relationship (similar or dissimilar) between tokens

  - Scale poorly as vocab. size grows

  - NN works poorly since input is very sparse

```
One hot encoding:

Computers : [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
Hardware  : [0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
```

One-hot vector to dense vector

- As a thought exercise

- Hand crafted vectors per word based on some features

- This is called embedding

  - Convert sparse vector to dense vector

- Issues:

  - Limitation in hand craft

 - Any other way to generate these types of vectors?

|	animal	fluffiness	dangerous	spooky
aardvark	0.97	0.03	0.15	0.04
black	0.07	0.01	0.20	0.95
cat	0.98	0.98	0.45	0.35
duvet	0.01	0.84	0.12	0.02
zombie	0.74	0.05	0.98	0.93

Word2vec

- Word2vec embeddings (Ref [3])

  - Project one-hot encodings to high dimensional vector space so that tokens with similar meaning to be close to each other

  - Trained with language model

  - Example

    - Man is to king, and woman is to Queen

      - $king - man + woman \approx queen$

  - Country to capital
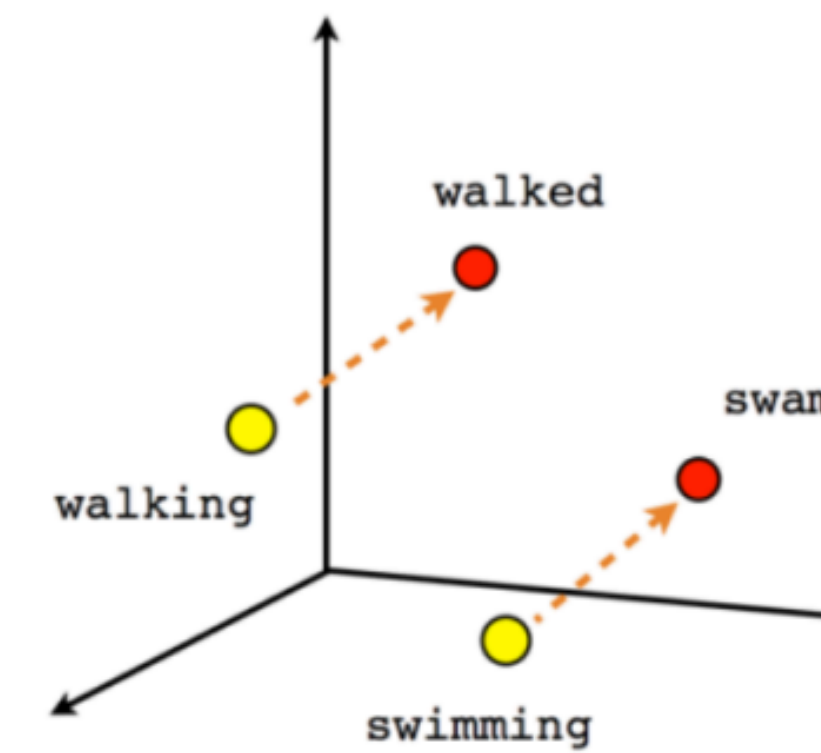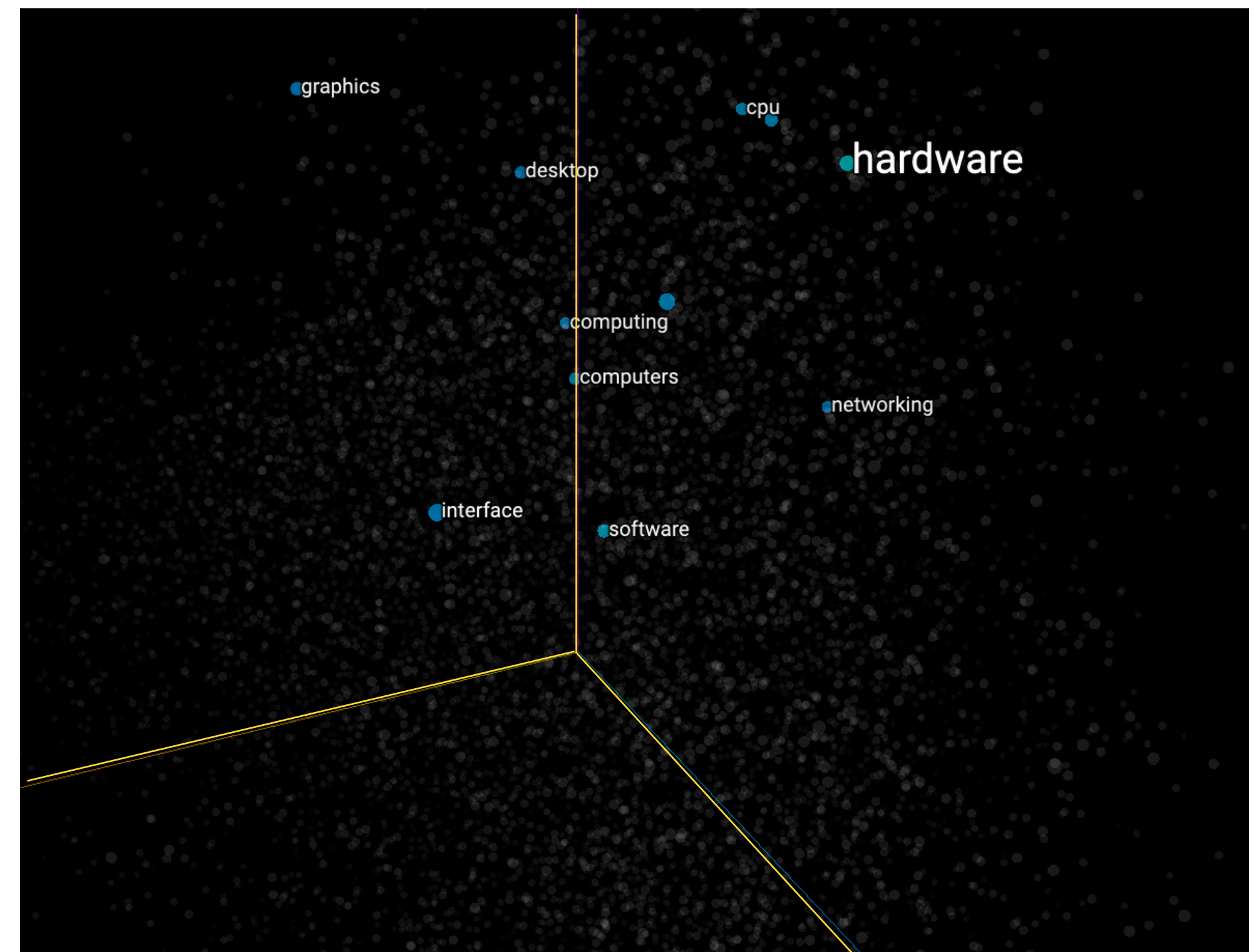
Verb tense
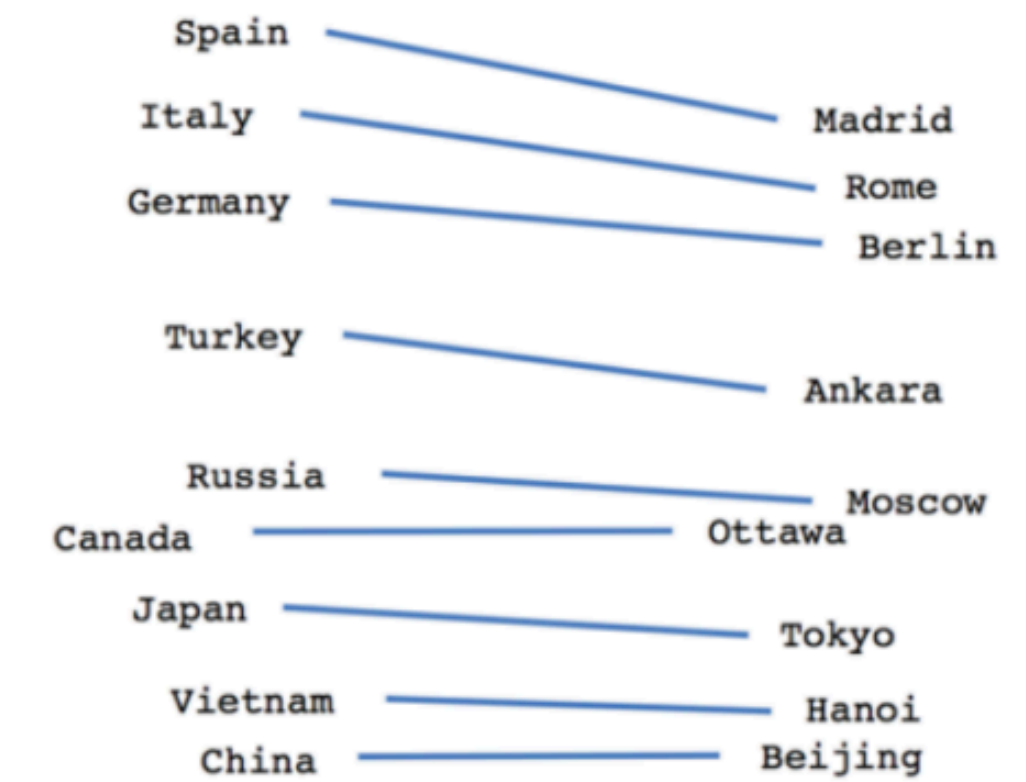
Country-Capital
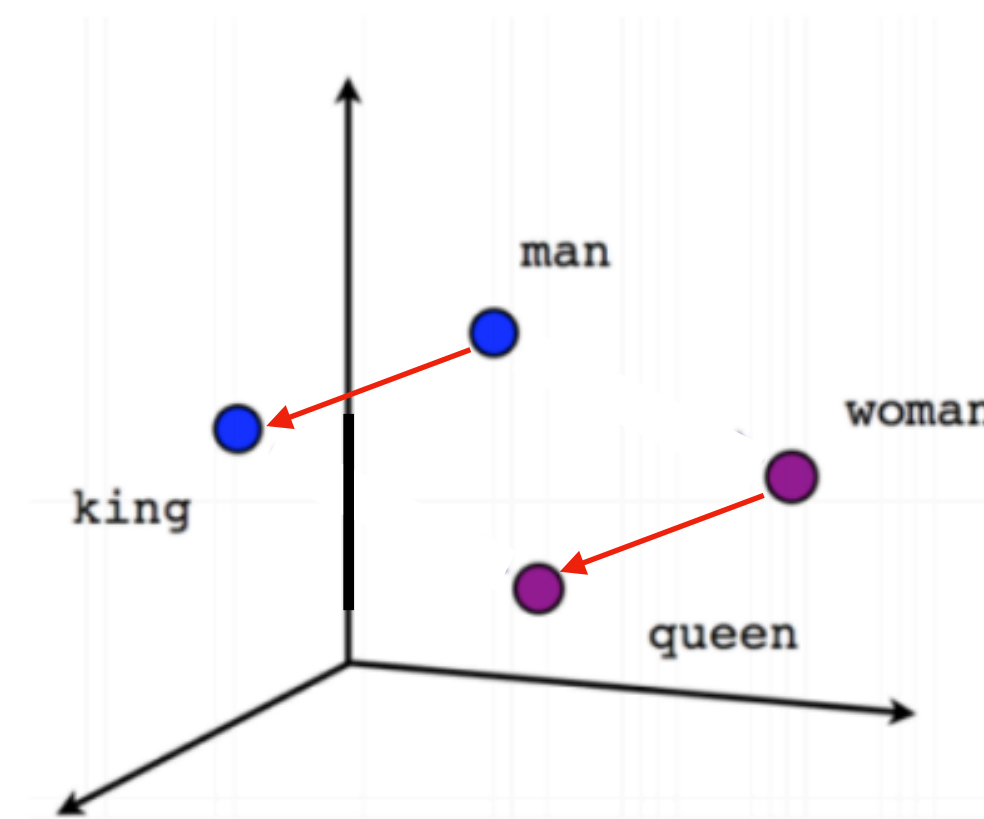
Male-Female

Word2vec

- Word2vec embeddings (Ref [3])

  - Visualization: https://
   projector.tensorflow.org

Embedding example (word2vec)

RNN

Recurrent neural network

- Neural network has two inputs

  - Input from the current time t

  - Hidden state from time $(t-1)$

- Allows stateful computation

- Any input sequence lengths can be processed

  - This cannot be done with NN

- Apply same W repeatedly

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

y_t Output

W Hidden state

h_t

x_t Input sequence (varying length)
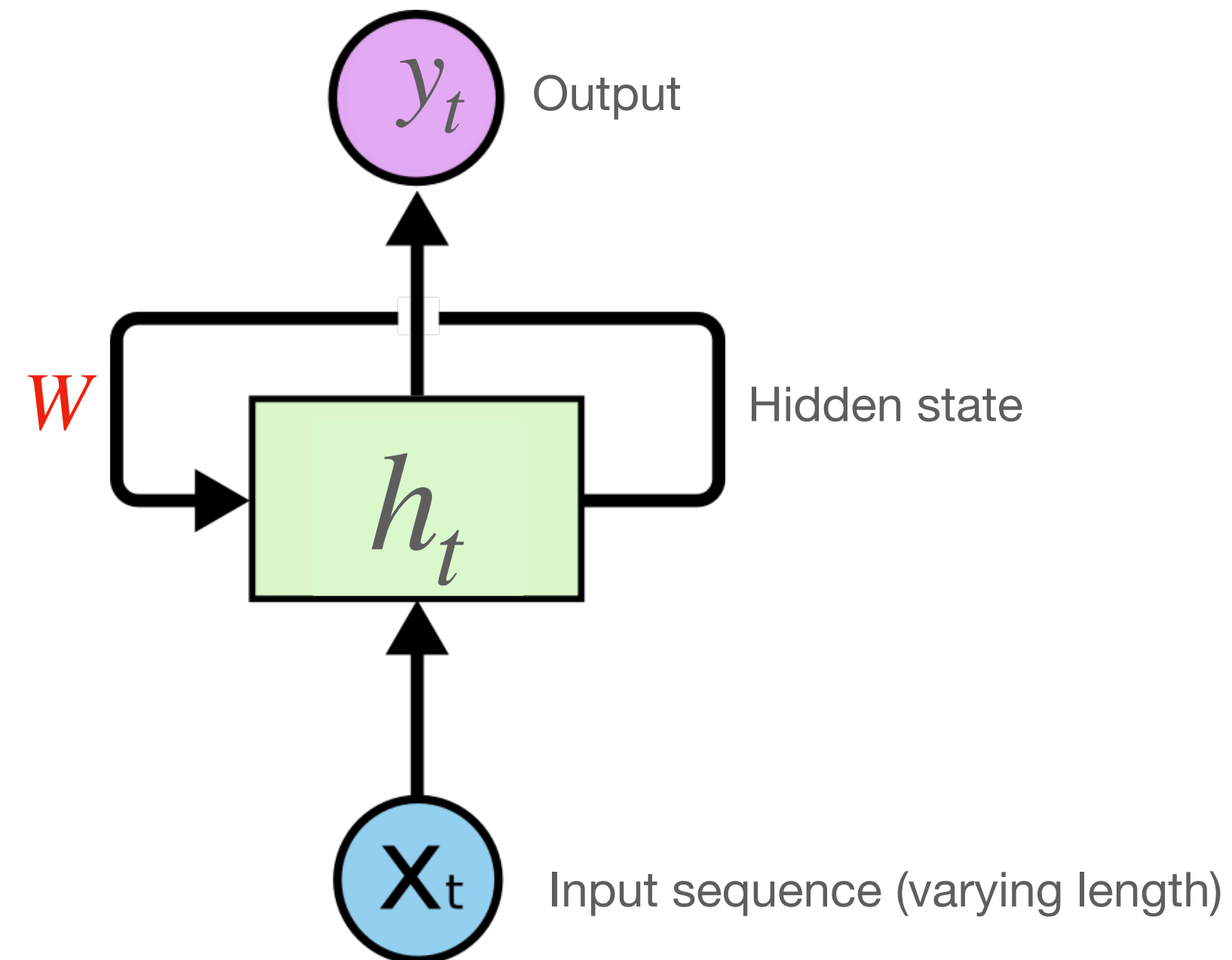
RNN

- The rollout representation

- Same model is used
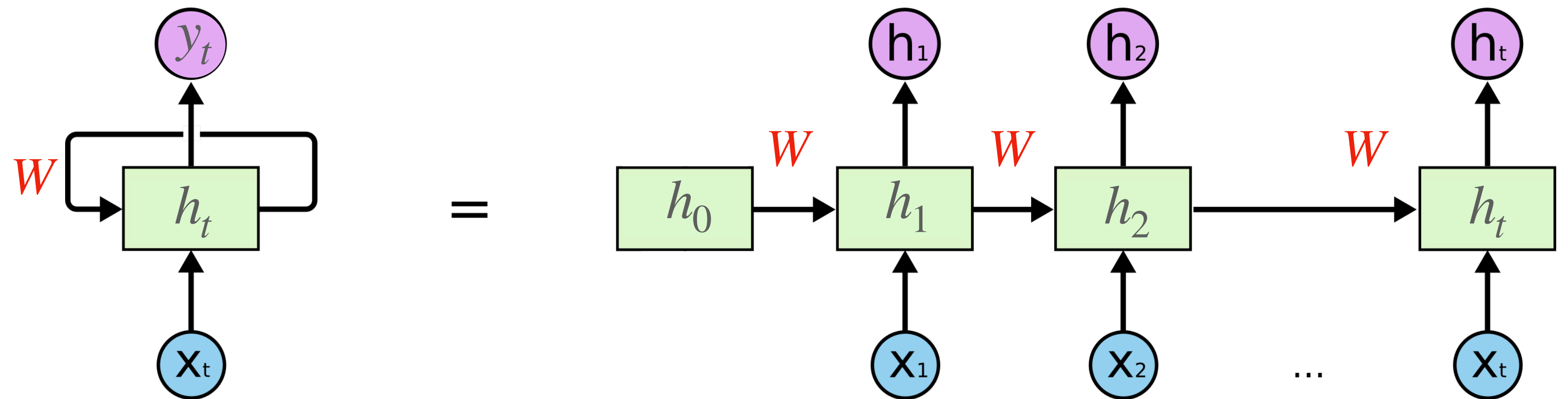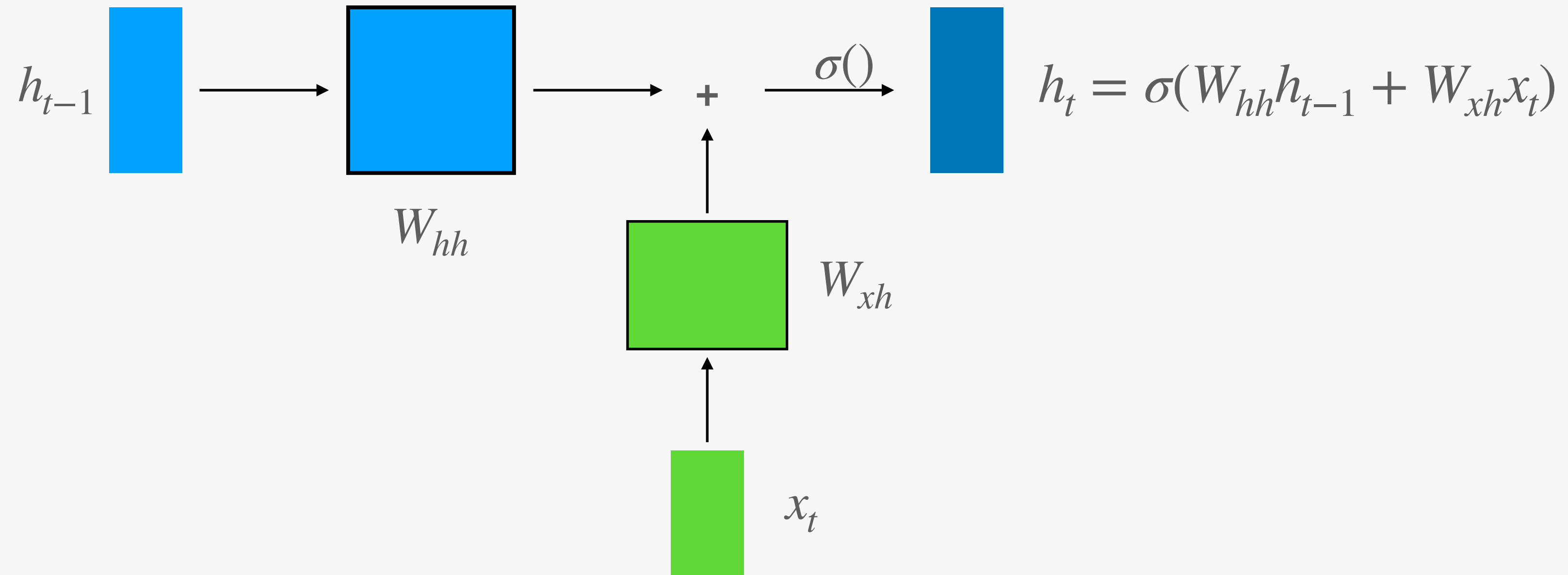

$$y_t, h_t = f(x_t, h_{t-1})$$

RNN in code

```python
class RNN:
  # ...
  def compute_next_h(self, x):
    # Vanilar RNN hidden computation
    self.h = np.tanh(np.dot(self.W_hh, self.h) + np.dot(self.W_xh, x))
    return h
  # ...
  def step(self, x):
    # update the hidden state
    self.h = self.compute_next_h(x)
    # compute the output vector
    y = np.dot(self.W_hy, self.h)
    return y
```

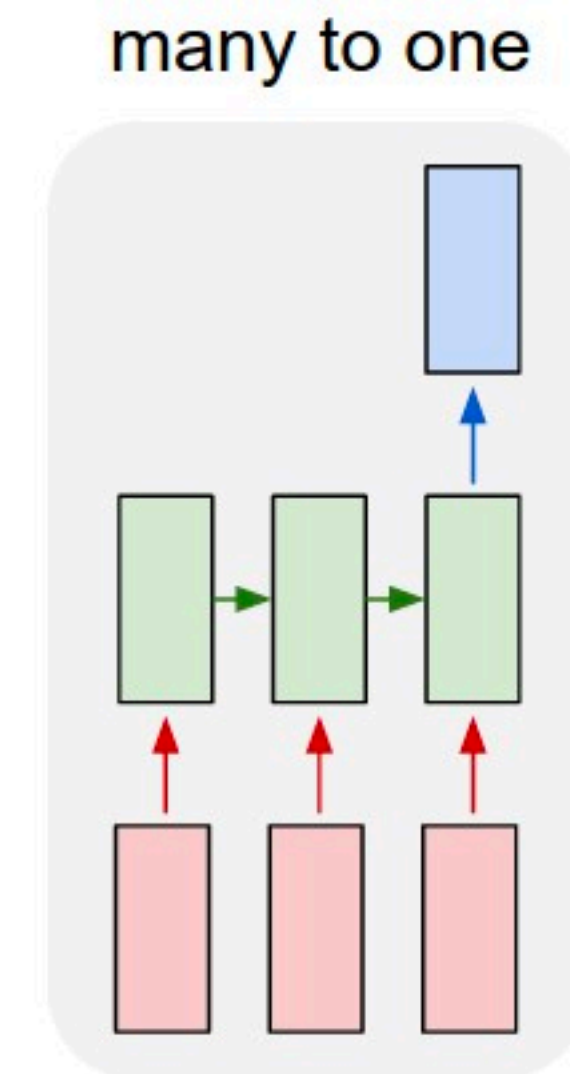$$h_t = \sigma(W_{hh}h_{t-1} + W_{xh}x_t)$$

RNN in code


```
class RNN:
```

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

W_{xh}

x_t

Applications

Many to one setup

- Give input sequence, compute a single output

- Example

  - sentiment analysis

many to one

Many to one setup

- Add the fully connected layer at the last hidden state

- Use it as a classification

- RNN is working as encoder, encodes input text

- FC is working as a decoder that use the encoded information from RNN to use it for classification
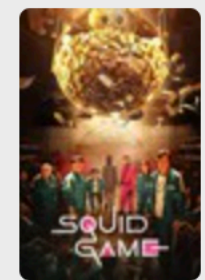
Season 1

Critics Consensus: *Squid Game*'s unflinching brutality is not for the faint of heart, but sharp social commentary and a surprisingly tender core will keep viewers glued to the screen - even if it's while watching between their fingers.

2021, Netflix, 9 episodes, View details

Input text

Tokenize

Embedding

h_0 $\quad W \quad$ h_1 $\quad W \quad$ h_2 $\quad W \quad$

e_1 $\qquad$ e_2 $\qquad$ ... $\qquad$ e_t

FC
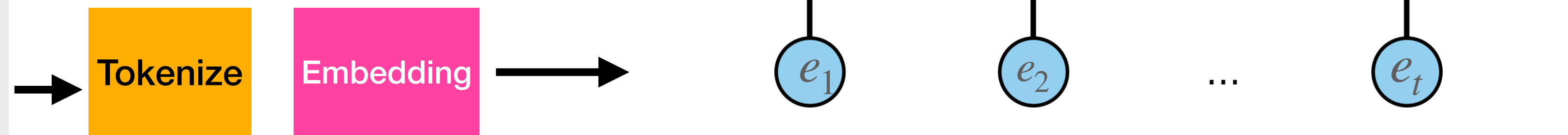
Many to many setup

- Many to many setup consists of encoder RNN and decoder RNN

- Encoder decoder architecture

  - Also called sequence to sequence (seq2seq)

- Dramatically replace all statistical machine translation (SMT) method to RNN based neural machine translation (NMT) in 2014
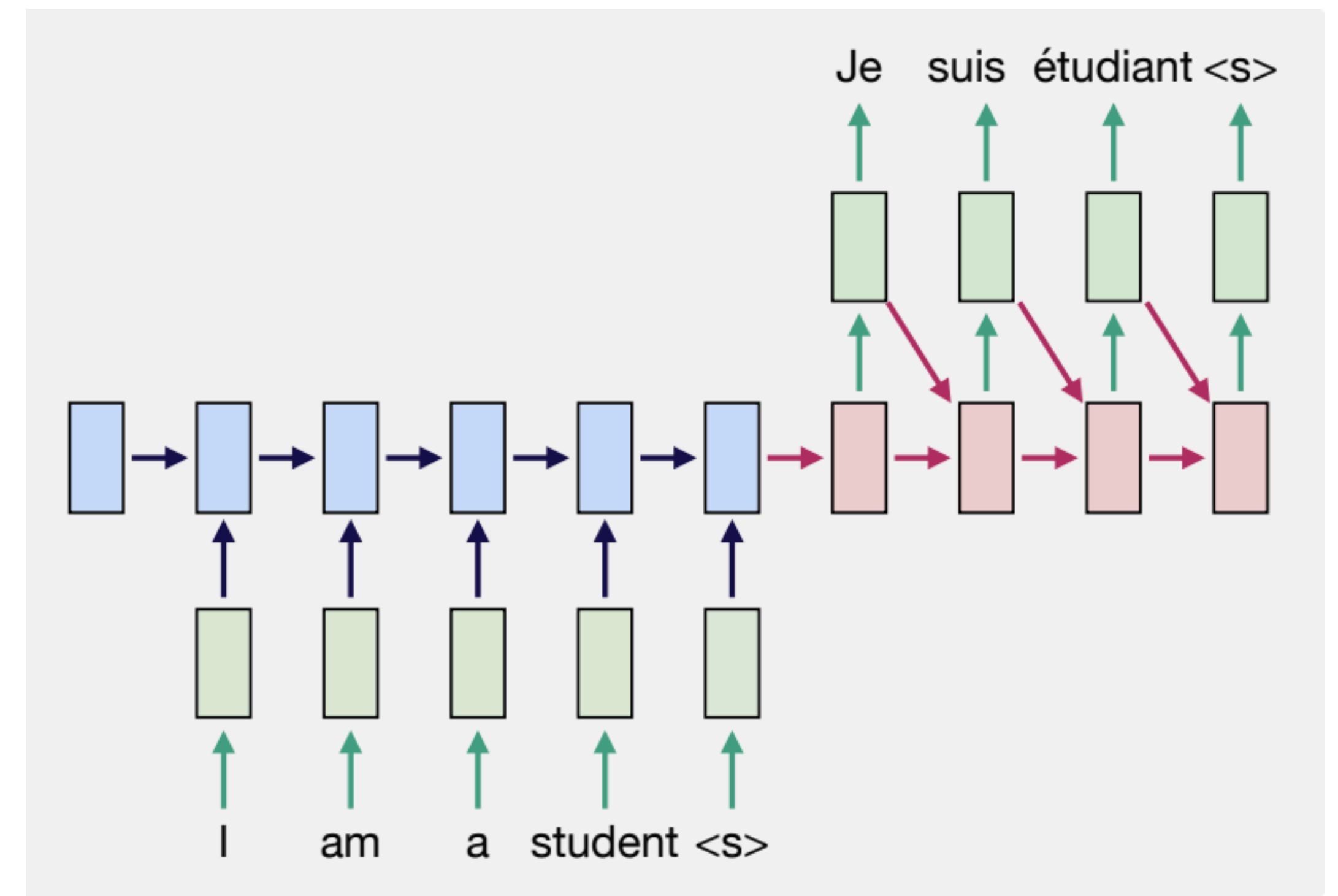
Fancy RNNs

Fancy RNNs

- Vanilla RNN has difficulty of capturing long term relations due to vanishing gradient

  - Fancy RNN that allows to capture past state better, LSTM (long short term memory), GRU (Gated recurrent unit)

- Stacked RNN: stacking RNNs to capture more complex information

- Bi-directional RNN: to capture in both forward and backward directional relations

Summary

Summary

- RNN can capture sequential relations from input data

- There are several configurations to use RNN, many-to-one, many-to-many, etc.

- Text dataset will go through tokenization, and then use embedding to represent high-dim information efficiently

- There are advanced RNN models, LSTM, GRU, and stacked RNN and bi-directional RNN

- Spoiler alert: RNN based NMT is replaced with Transformer based NMT in 2019!

Credits
The following materials are used for this slide.

1. "Recurrent Neural Networks" section of Dive into Deep Learning

2. "RNNs" section of Full Stack Deep Learning

3. Understanding LSTM Networks

4. Why do we use word embeddings in NLP?

References

1. Neural network

2. Language Models and Recurrent Neural Networks

3. The Illustrated Word2vec