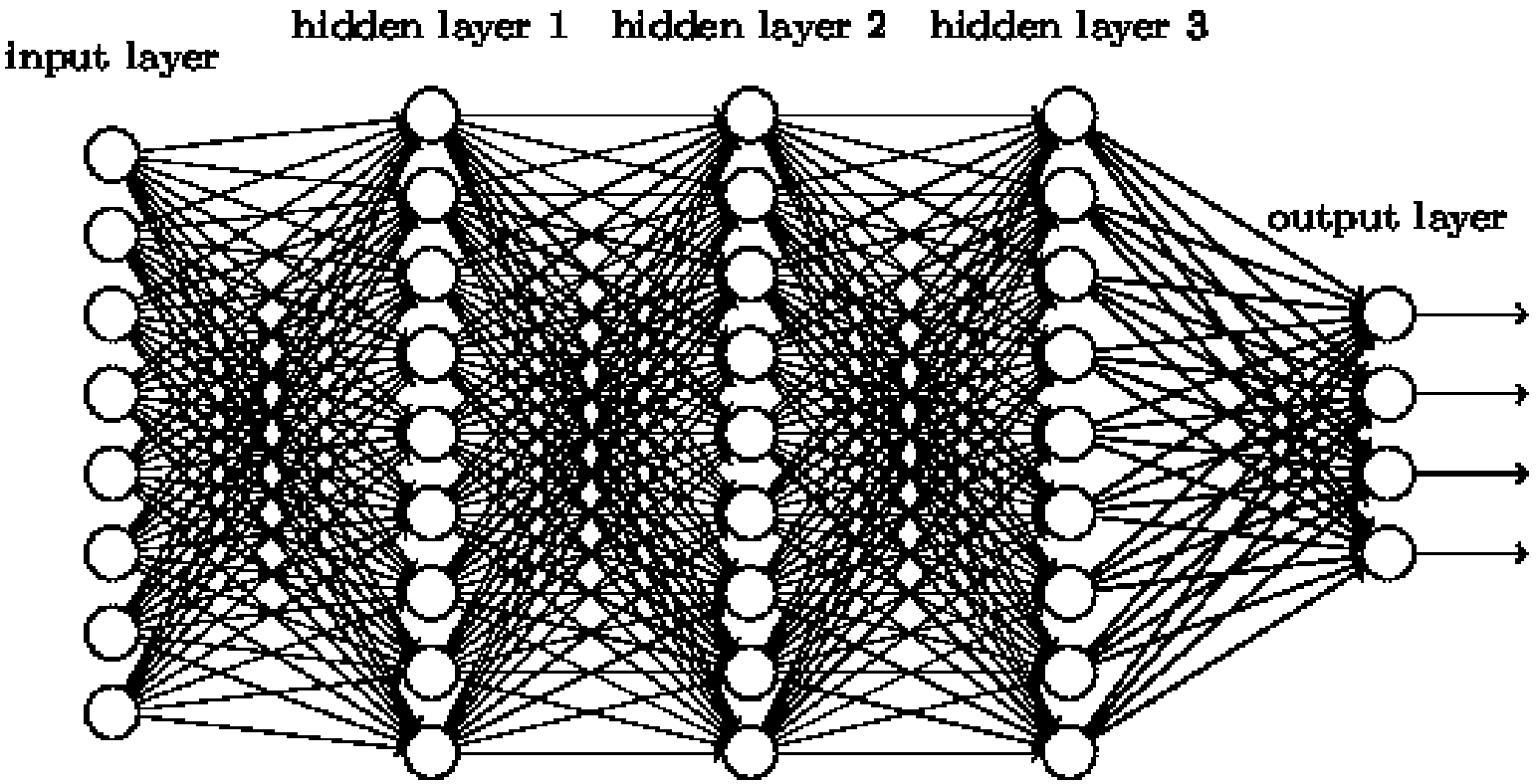
Recurrent neural networks

Motivation: what about sequence prediction?



What can I do when input size and output size vary?

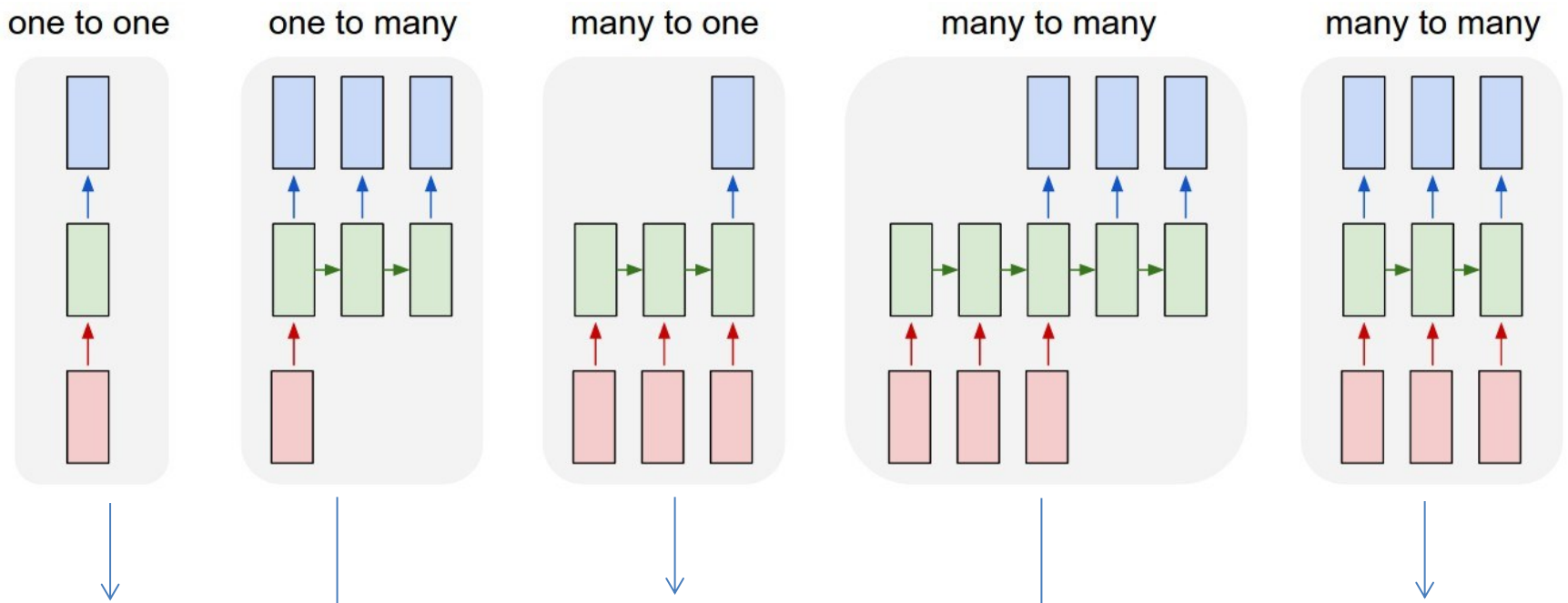
Motivation

- Feed forward networks accept a fixed-sized vector as input and produce a fixed-sized vector as output

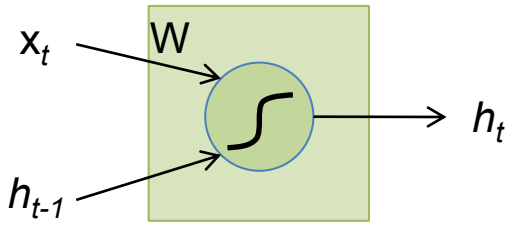
- fixed amount of computational steps

- recurrent nets allow us to operate over *sequences* of vectors

Motivation

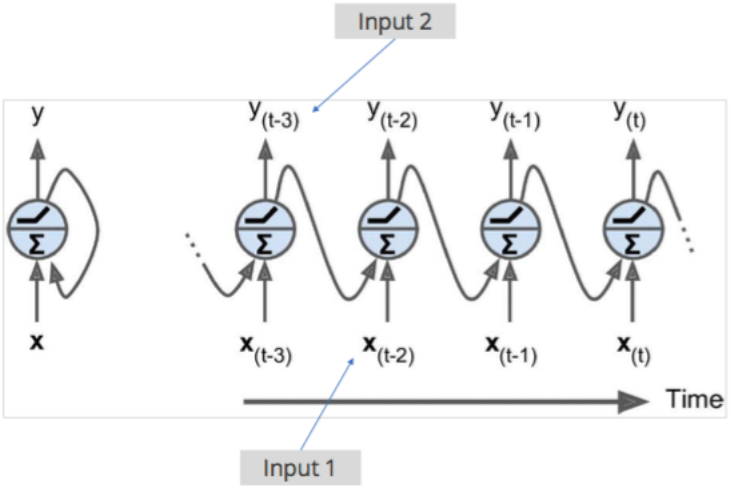

one to one
one to many
many to one
many to many
many to many
Image classification
Image captioning
Sentiment analysis
Machine translation
Synced sequence(vide
classification)

The Vanilla RNN Cell

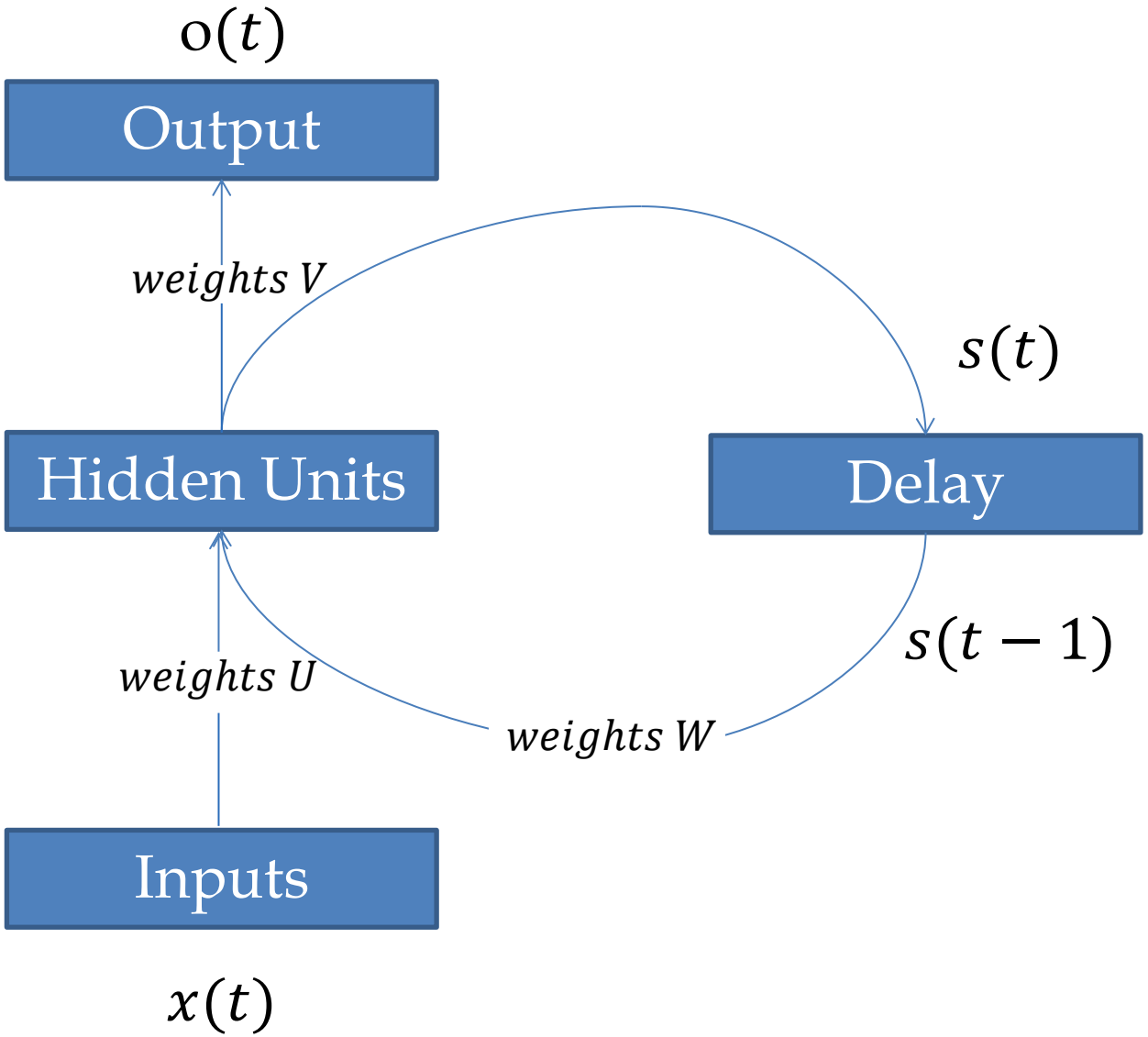


$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

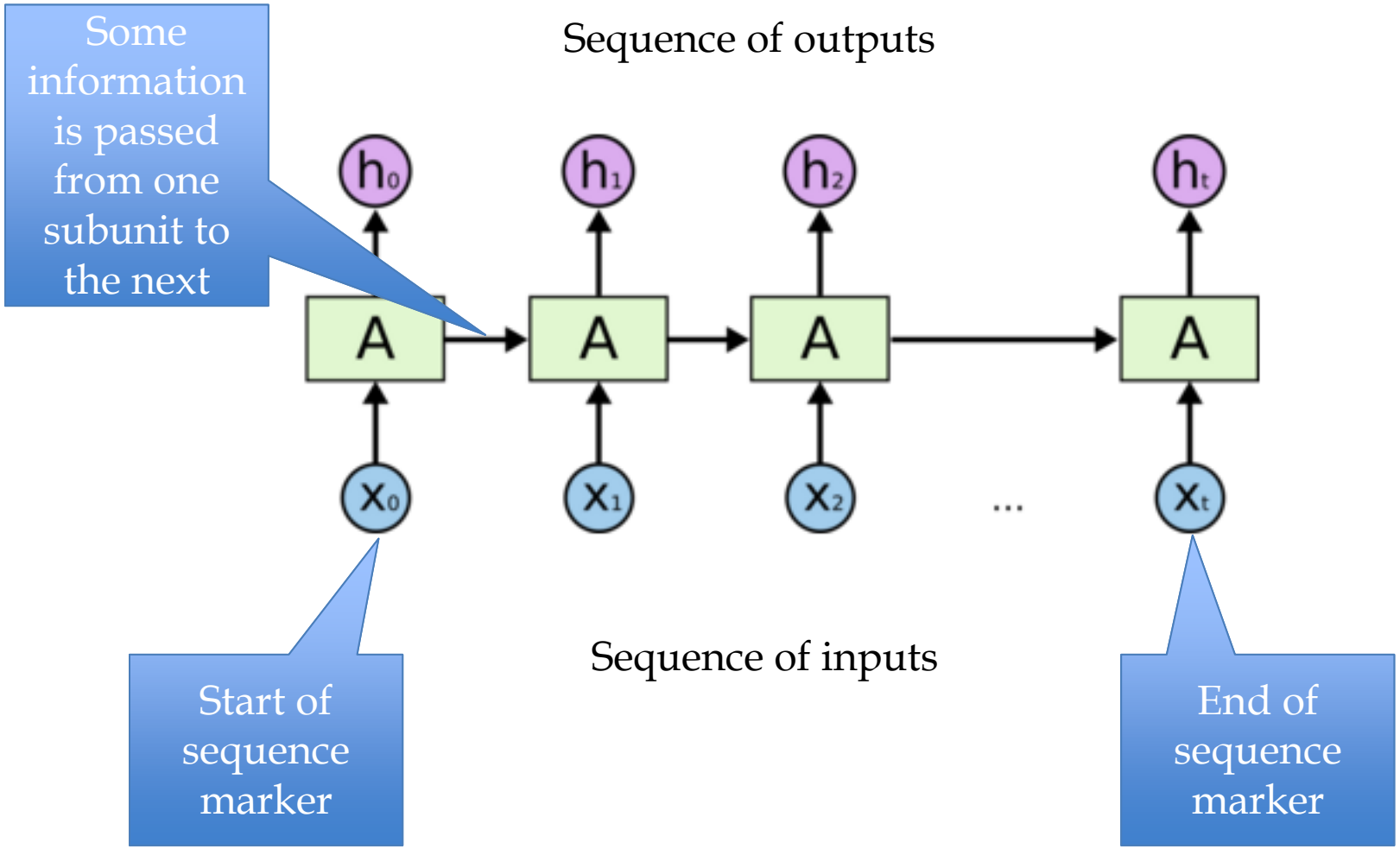
- An RNN looks like a feedforward network except it has connections pointing backward.
- This tiny network can be plotted against the time axis as shown in the figure.
- The figure shows one neuron that receives inputs, produces an output, and sends that output back to itself.
- This is called unrolling the network through time.



RNN Architecture



Architecture for an RNN



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

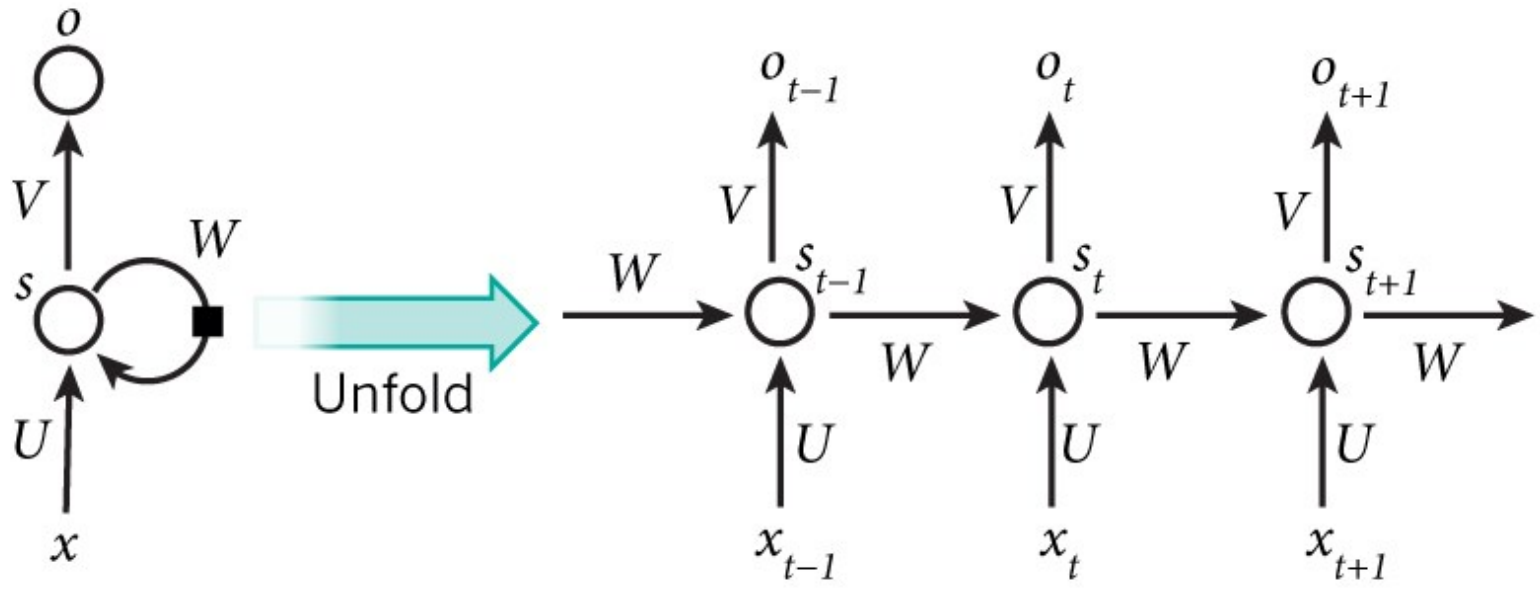
Back Propagation Through Time

- BPTT learning algorithm is an extension of standard backpropagation that performs gradients descent on an unfolded network.

- The gradient descent weight updates have contributions from each time step.

- The errors have to be back-propagated through time as well as through the network

Lets assume RNN as below for BPTT



- The recurrent network can be converted into a feed forward network by **unfolding over time**

RNN Backward Pass

- For recurrent networks, the loss function depends on the activation of the hidden layer through its influence on the output layer **and** through its influence on the hidden layer at the next step.

$$\frac{\partial E}{\partial a_o(t)} = \frac{\partial E}{\partial o(t)}\frac{\partial o(t)}{\partial a_o\,(t)} = \delta_o(t) = \big(d(t) - o(t)\big) f_o'(a_o(t))$$

$$\frac{\partial E}{\partial a_h(t)} = \delta_h(t) = f_h'(a_h(t)) \odot (V^T \delta_O(t) + W^T \delta_h(t+1))$$

$$\delta_j(t) = \frac{\partial E}{\partial a_j(t)}, j \in \{o, h\} \quad , \odot \text{ is the Hadamard product}$$

RNN Backward Pass

- We sum over the whole sequence to get the derivatives w.r.t the network weights:

$$E = \sum_{t=0}^{T} E_t \rightarrow \frac{\partial E}{\partial W} = \sum_{t=0}^{T} \frac{\partial E_t}{\partial W} = \sum_{t=0}^{T} \frac{\partial E_t}{\partial a_h} \frac{\partial a_h}{\partial W} = \sum_{t=0}^{T} \delta_h(t) \frac{\partial a_h}{\partial W}$$

- $\frac{\partial E}{\partial V} = s(t) \cdot \delta_o(t); \frac{\partial E}{\partial U} = x(t) \cdot \delta_h(t); \frac{\partial E}{\partial W} = s(t-1) \cdot \delta_h(t)$

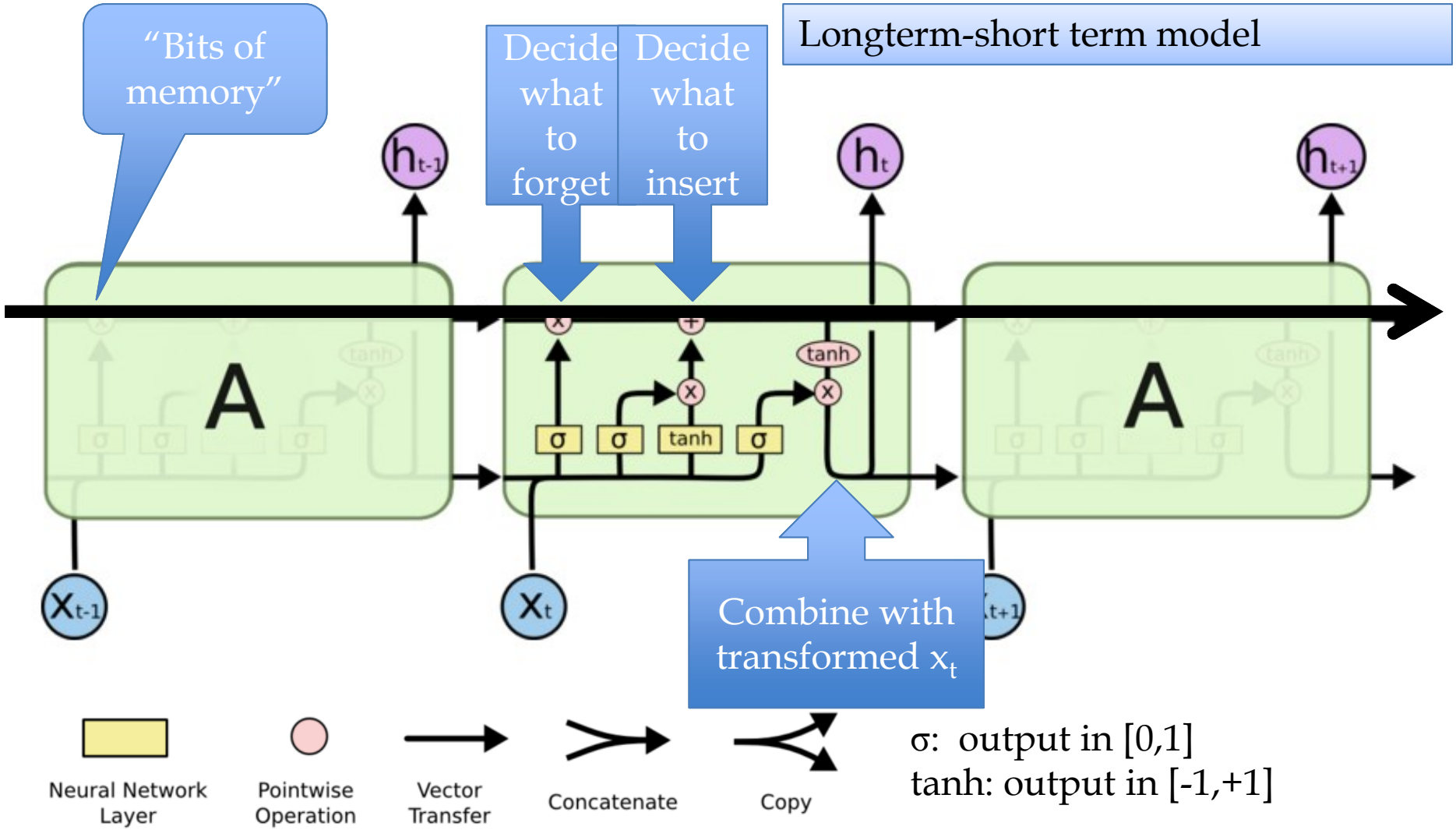
- Updating the weights:

$$V(t+1) = V(t) + \alpha \frac{\partial E}{\partial V}$$

$$U(t+1) = U(t) + \alpha \frac{\partial E}{\partial U}$$

$$W(t+1) = W(t) + \alpha \frac{\partial E}{\partial W}$$

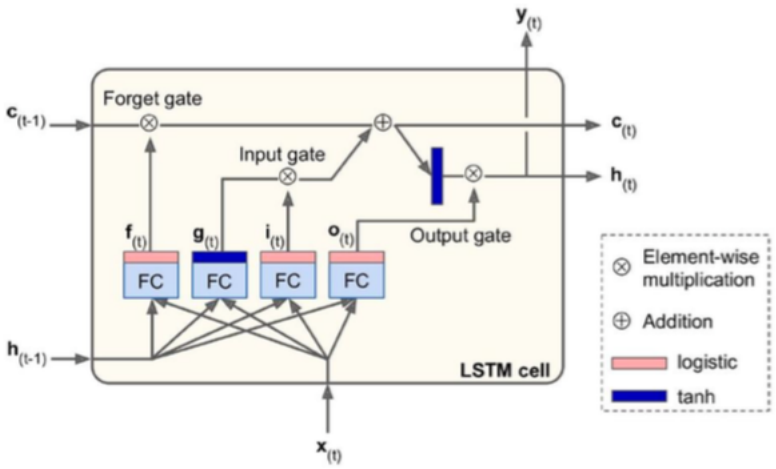
Architecture for an LSTM



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

LSTM Cell

- An LSTM cell has two states:
  - $\boldsymbol{c}_{(t)}$ - long-term state
  - $\boldsymbol{h}_{(t)}$ - short-term state
- The long-term state passes through forget gate to forget some memories and input gate to allow some memories.
- The long-term state is then passed through gate without transformation.
- The long-term state is also taken through tan h operation. The result is filtered by the output gate to produce short-term state $\boldsymbol{h}_{(t)}$. This is equal to the cell's output for the time step $\boldsymbol{y}_{(t)}$.



LSTM Cell States and Computation

LSTM Cell Computations

$$
\begin{aligned}
\boldsymbol{i}_{(t)} &= \sigma(\boldsymbol{W}_{xi}^{T} \cdot \boldsymbol{x}_{(t)} + \boldsymbol{W}_{hi}^{T} \cdot \boldsymbol{h}_{(t-1)} + \boldsymbol{b}_{i}) \\
\boldsymbol{f}_{(t)} &= \sigma(\boldsymbol{W}_{xf}^{T} \cdot \boldsymbol{x}_{(t)} + \boldsymbol{W}_{hf}^{T} \cdot \boldsymbol{h}_{(t-1)} + \boldsymbol{b}_{f}) \\
\boldsymbol{o}_{(t)} &= \sigma(\boldsymbol{W}_{xo}^{T} \cdot \boldsymbol{x}_{(t)} + \boldsymbol{W}_{ho}^{T} \cdot \boldsymbol{h}_{(t-1)} + \boldsymbol{b}_{o}) \\
\boldsymbol{g}_{(t)} &= \tanh(\boldsymbol{W}_{xg}^{T} \cdot \boldsymbol{x}_{(t)} + \boldsymbol{W}_{hg}^{T} \cdot \boldsymbol{h}_{(t-1)} + \boldsymbol{b}_{g}) \\
\boldsymbol{c}_{(t)} &= \boldsymbol{f}_{(t)} \otimes \boldsymbol{c}_{(t-1)} + \boldsymbol{i}_{(t)} \otimes \boldsymbol{g}_{(t)} \\
\boldsymbol{y}_{(t)} &= \boldsymbol{h}_{(t)} = \boldsymbol{o}_{(t)} \otimes \tanh(\boldsymbol{c}_{(t)})
\end{aligned}
$$

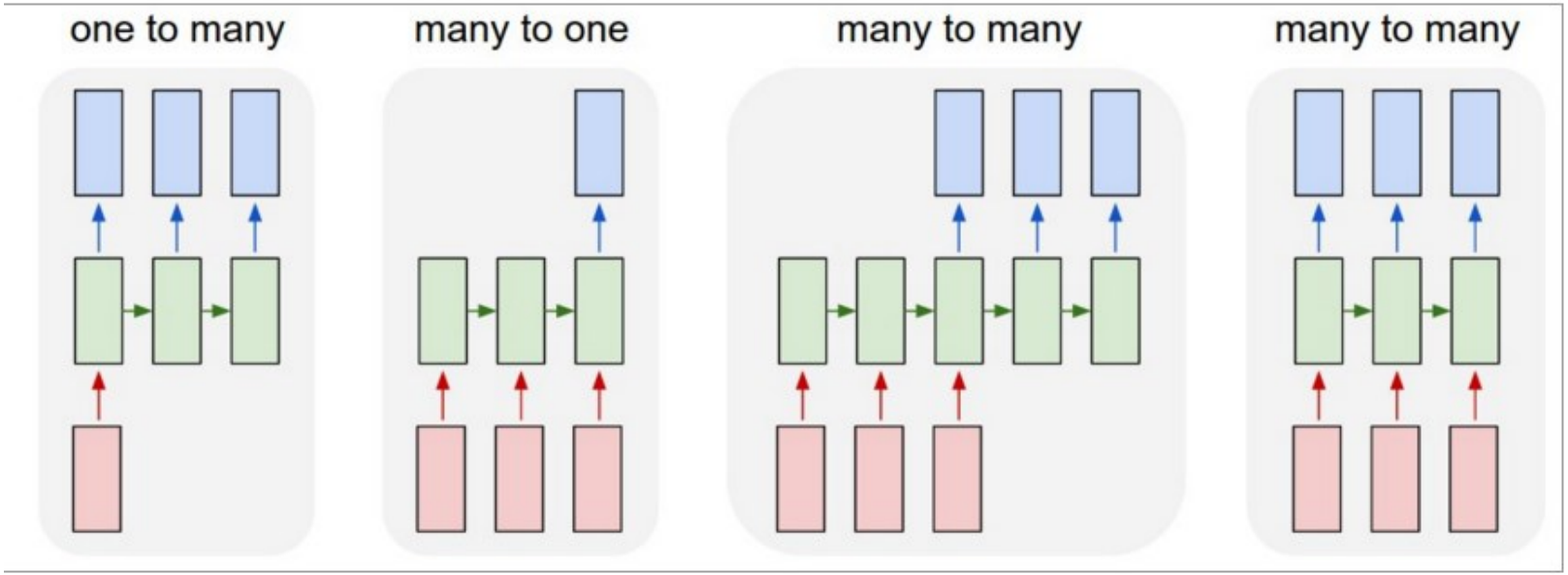
LSTMs can be used for other sequence tasks

image captioning

sequence classification

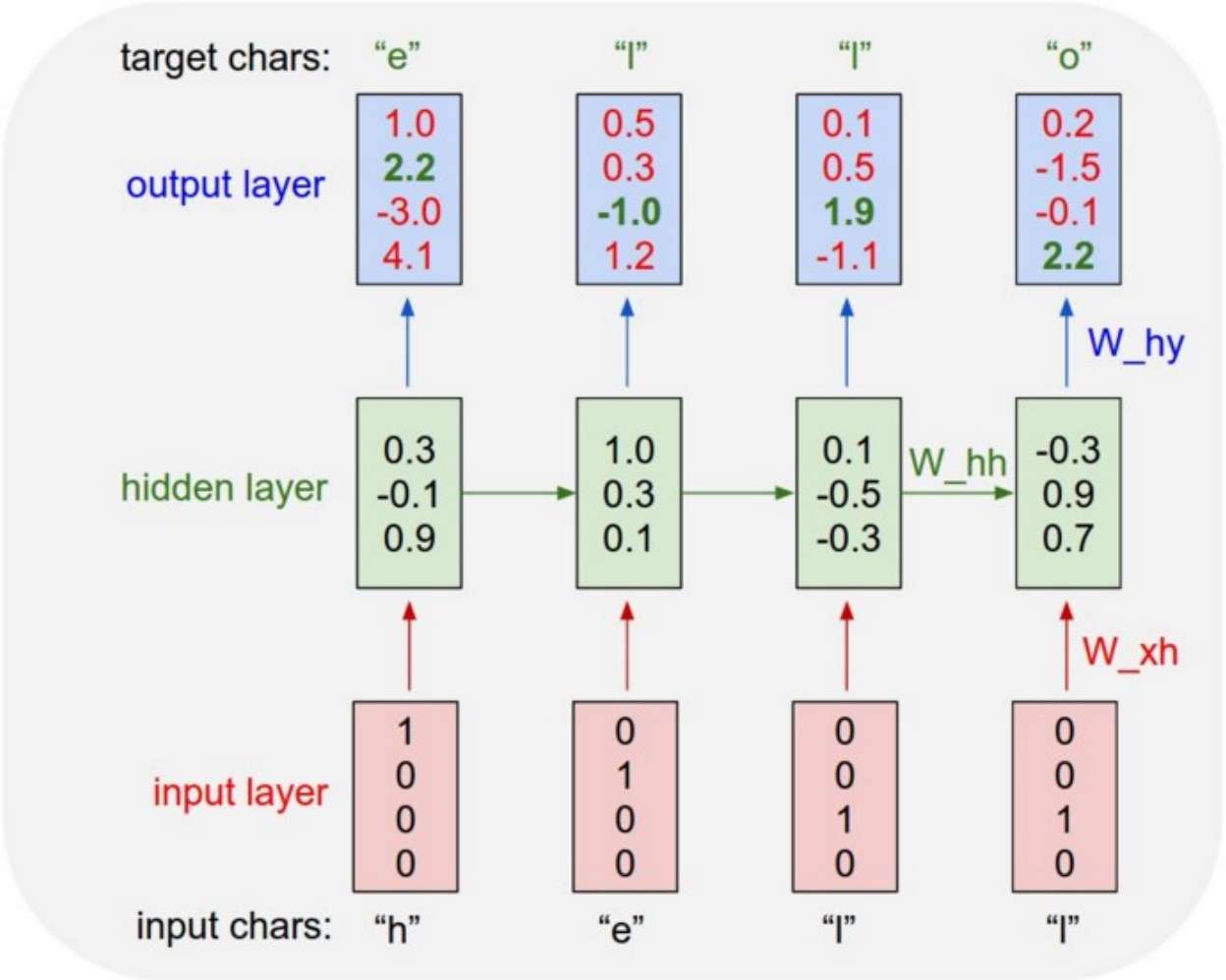
translation

named entity recognition



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

Character-level language model



Test time:

- pick a seed character sequence
- generate the next character
- then the next
- then the next …

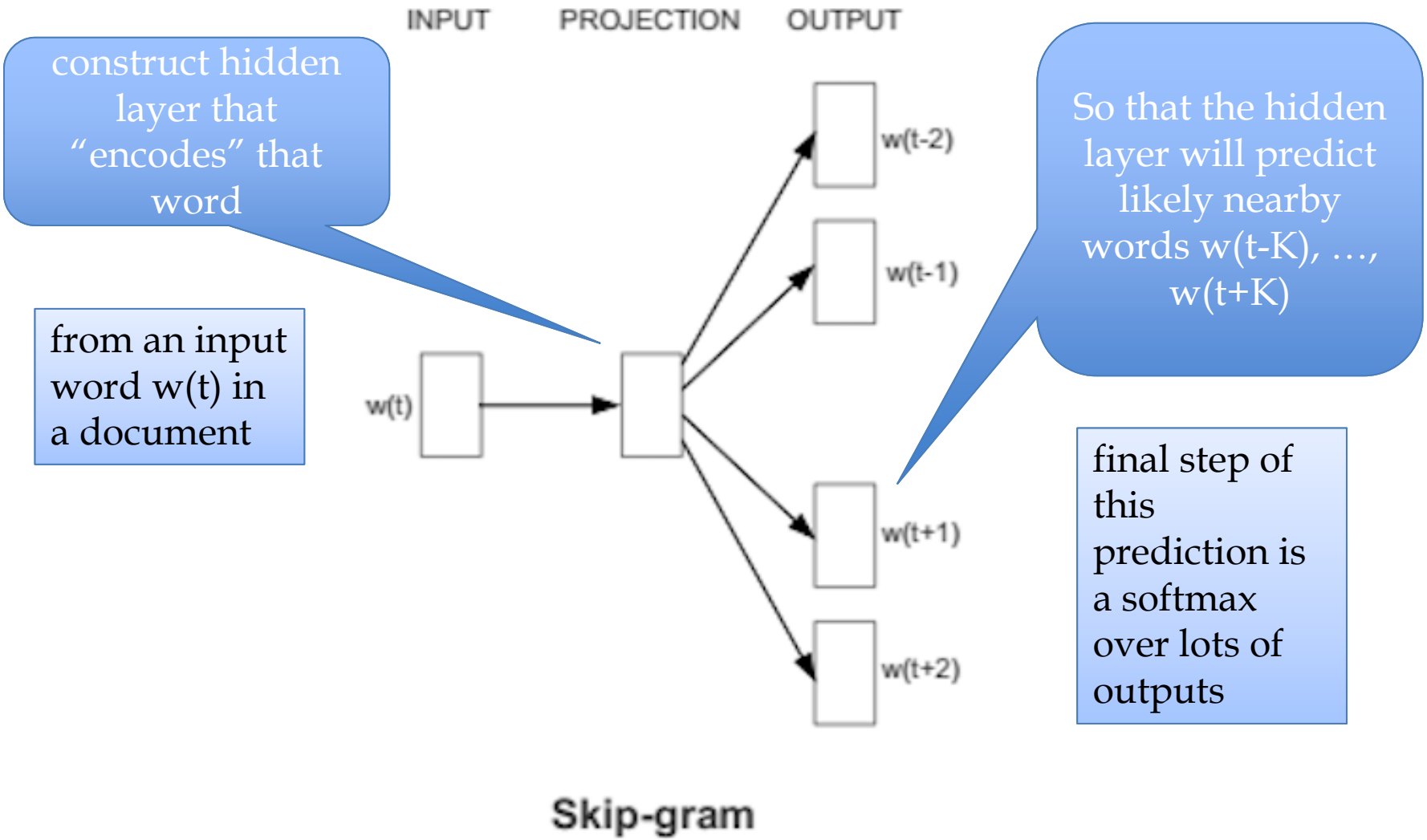
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Character-level language model

```
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.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.
```

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

Basic idea behind skip-gram embeddings



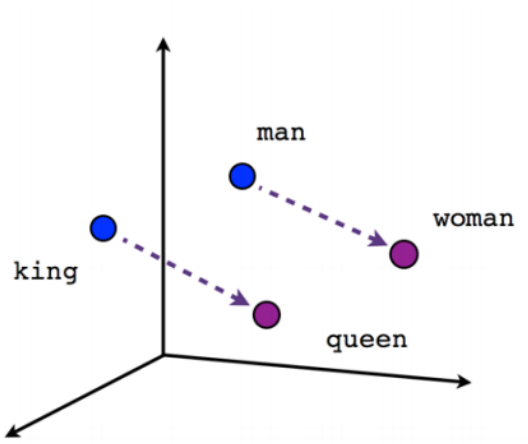
construct hidden layer that "encodes" that word

from an input word w(t) in a document

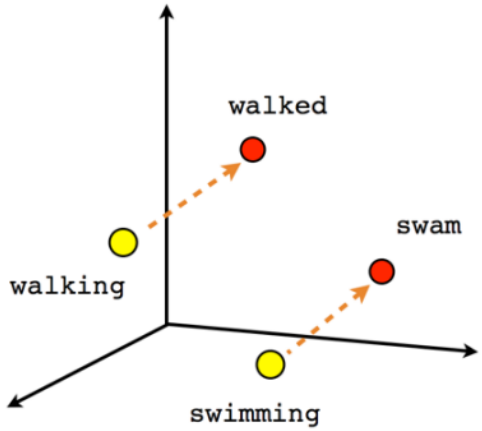
So that the hidden layer will predict likely nearby words w(t-K), …, w(t+K)

final step of this prediction is a softmax over lots of outputs

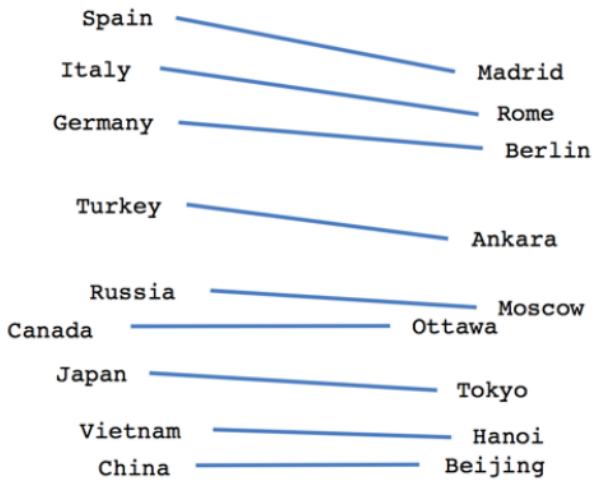
Results from word2vec



Male-Female



Verb tense



Country-Capital

https://www.tensorflow.org/versions/r0.7/tutorials/word2vec/index.html

Results from word2vec

https://www.tensorflow.org/versions/r0.7/tutorials/word2vec/index.html

Examples of attention

(a) Translated MNIST inputs.

(b) Cluttered Translated MNIST inputs.

https://papers.nips.cc/paper/5542-recurrent-models-of-visual-attention.pdf

Some current hot topics

- Knowledge-base embedding: extending word2vec to embed large databases of facts about the world into a low-dimensional space.
  - TransE, TransR, …
- “NLP from scratch”: sequence-labeling and other NLP tasks with minimal amount of feature engineering, only networks and character- or word-level embeddings

Some current hot topics

- Computer vision: complex tasks like generating a natural language caption from an image or understanding a video clip
- Machine translation
  - English to Spanish, …
- Using neural networks to perform *tasks*
  - Driving a car
  - Playing games (like Go or …)