NLP Techniques and Applications

- Text prediction using RNN

N-Grams

- Large N-grams to capture dependencies between distant words
- Need a lot of space and RAM
    - N-grams consume a lot of memory
- Different types of RNNs are the preferred alternative

Recurrent Neural Networks

Nour was supposed to study with me. I called her but she did not answer / have

want

respond

choose

want

have

ask

attempt

answer

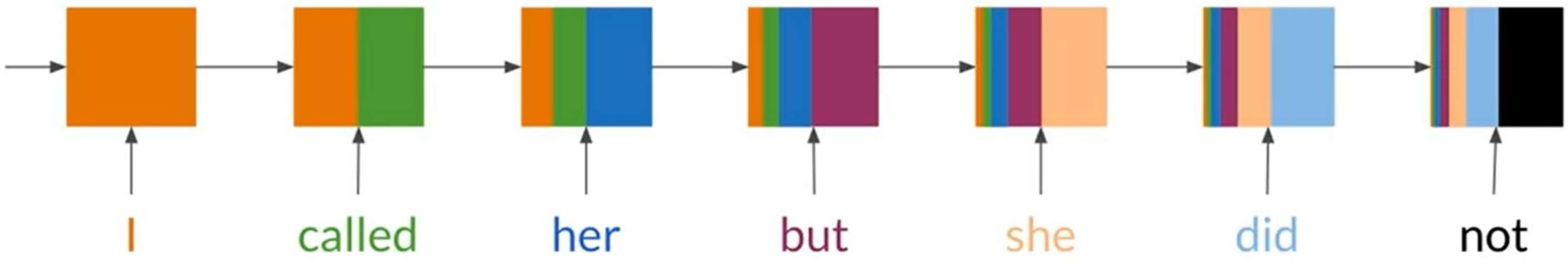
know

RNNs look at every previous word

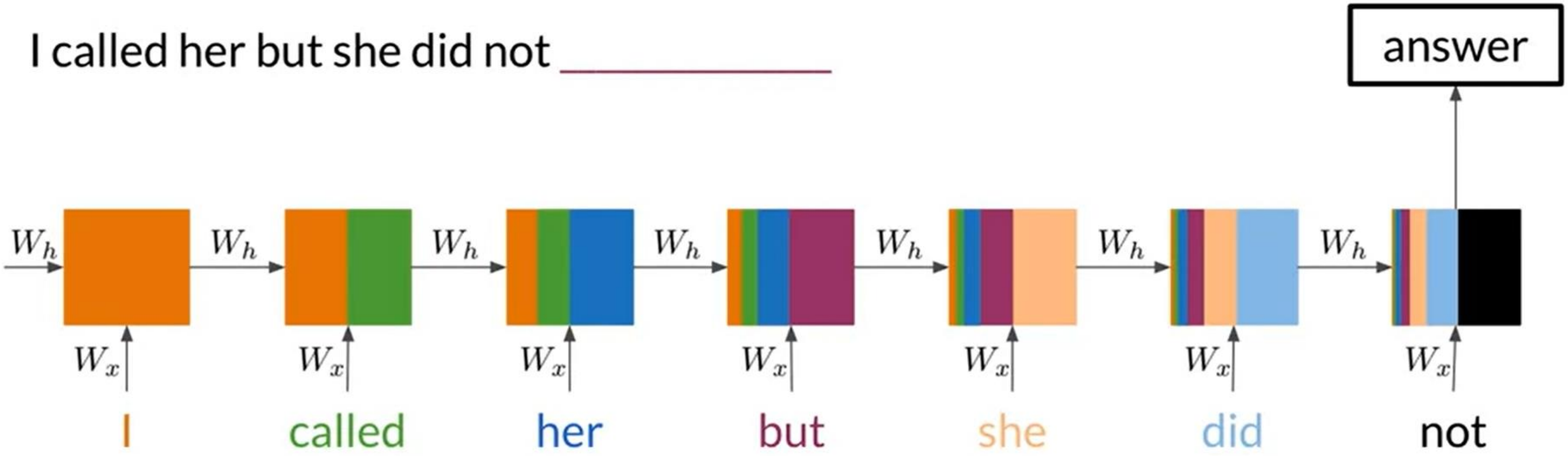
Similar probabilities with trigram

Recurrent Neural Networks

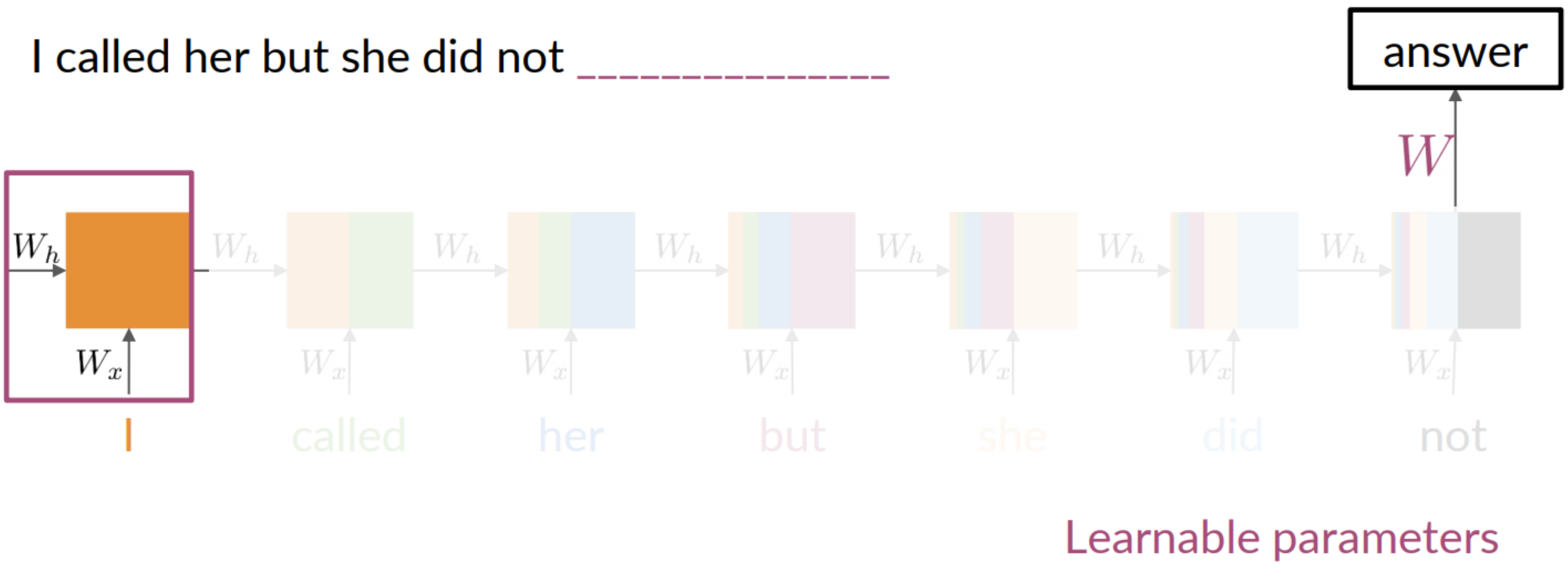
I called her but she did not ____________



Recurrent Neural Networks



Recurrent Neural Networks

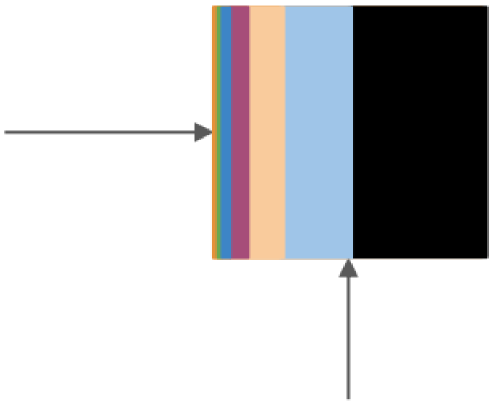


Q. An RNN would have the same number of parameters for word sequences of different lengths?

Recurrent Neural Networks

Summary

- RNNs model relationships among distant words
- In RNNs a lot of computations share parameters



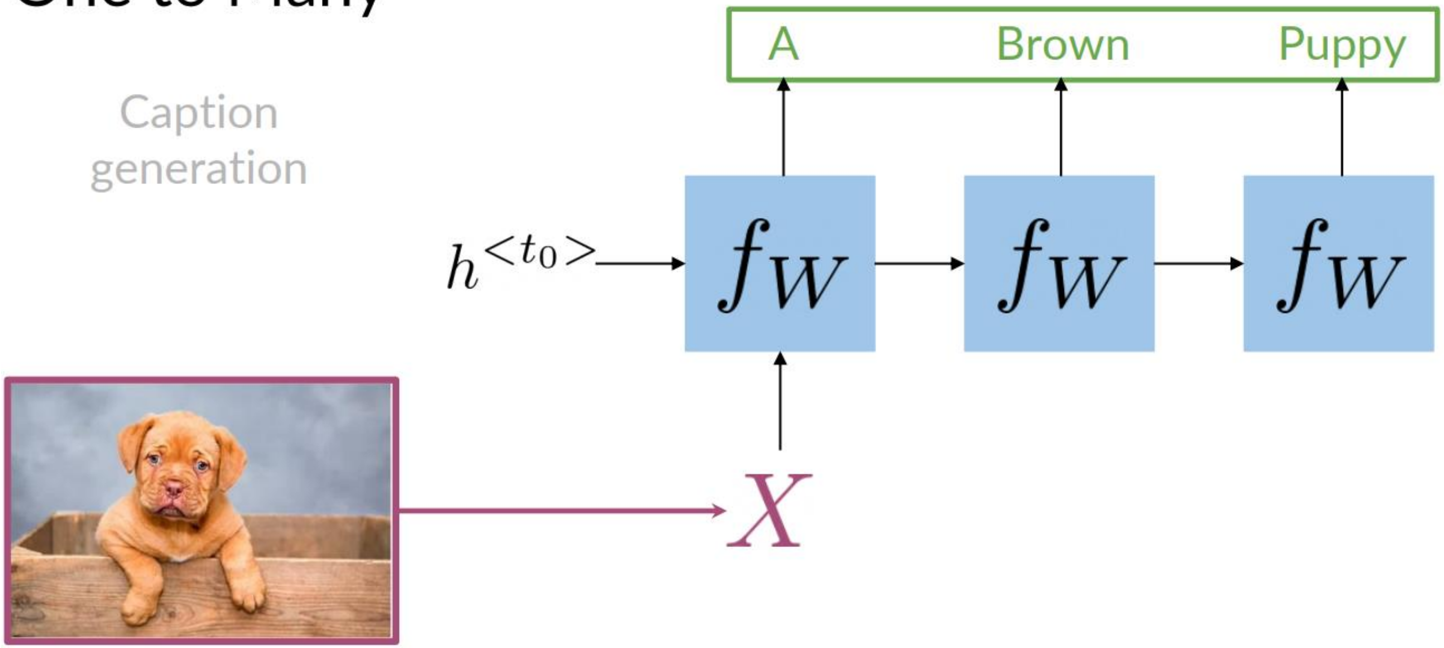
Applications of RNN

One to One

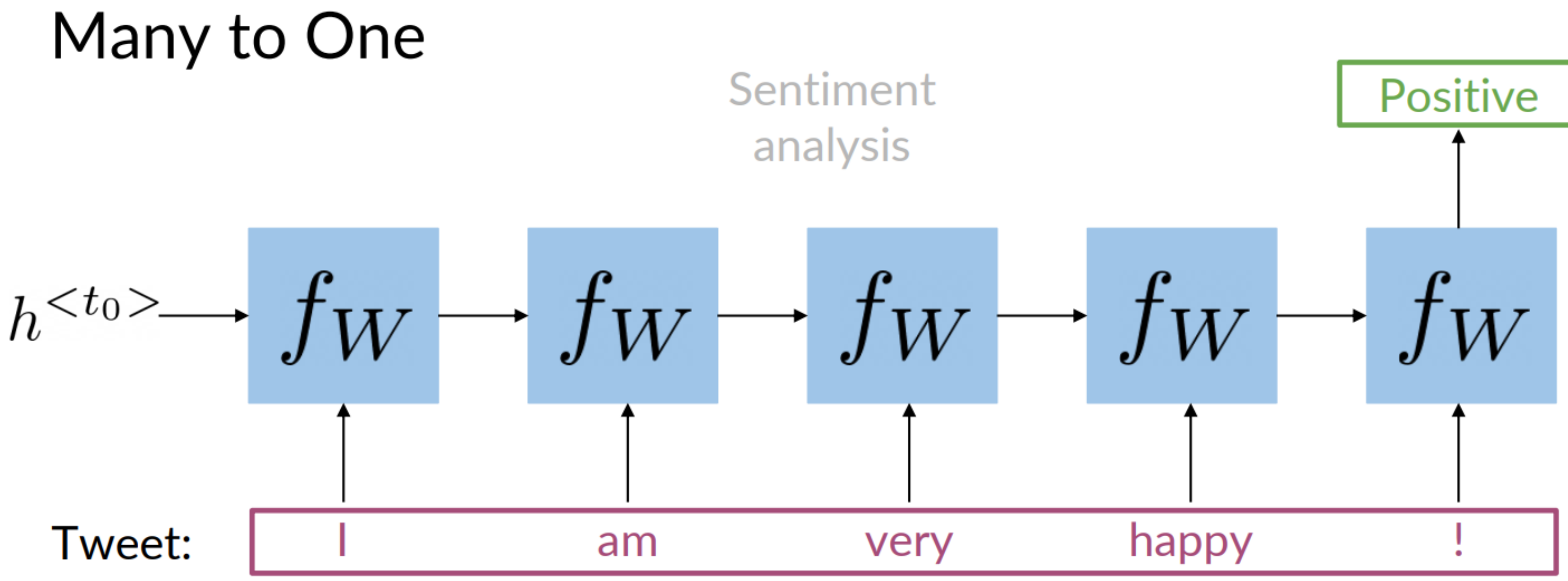
Applications of RNN

One to Many

Caption generation

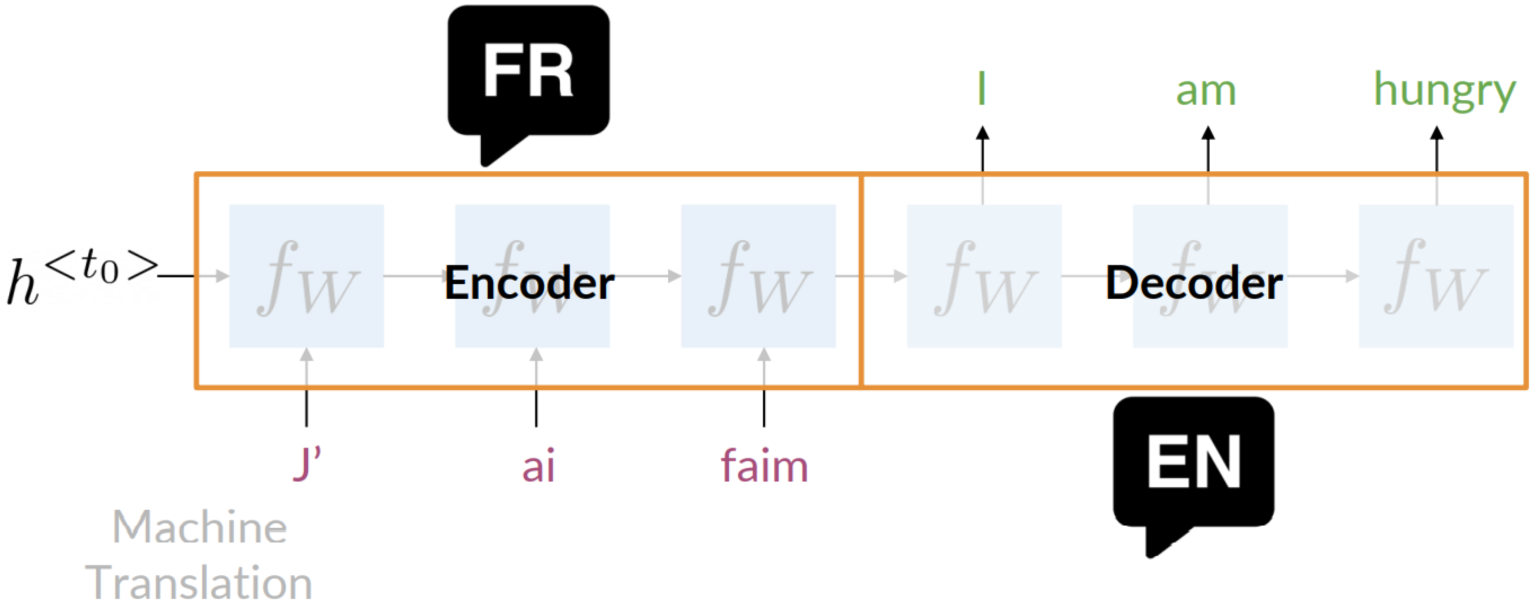


Applications of RNN



Applications of RNN

Many to Many



Applications of RNN

Q. “one to many” architecture?

- An RNN which inputs a conversation and determines the topic.
- An RNN which inputs a topic and generates a conversation about that topic.
- An RNN which inputs a sentiment and generates a sentence.
- An RNN which inputs a sentence and determines the sentiment.

RNN

Summary

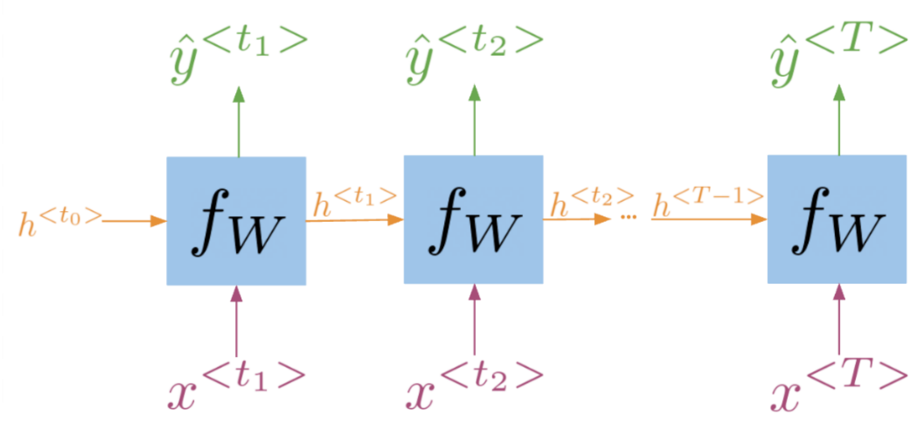
- RNNs can be implemented for a variety of NLP tasks
- Applications include Machine translation and caption generation

Maths in simple RNN

- How RNNs propagate information (Through time!)
- How RNNs make predictions

Maths in simple RNN

A Vanilla RNN

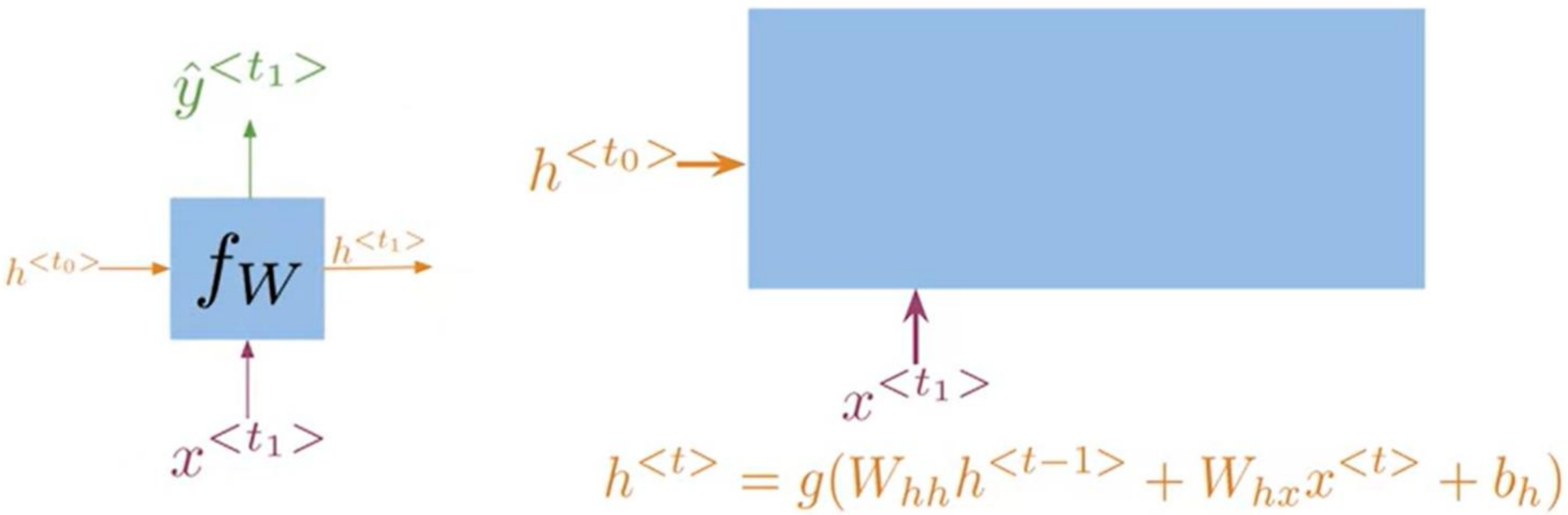


$$h^{<t>} = g(\boxed{W_h[h^{<t-1>}, x^{<t>}]}\boxed{+ b_h})$$

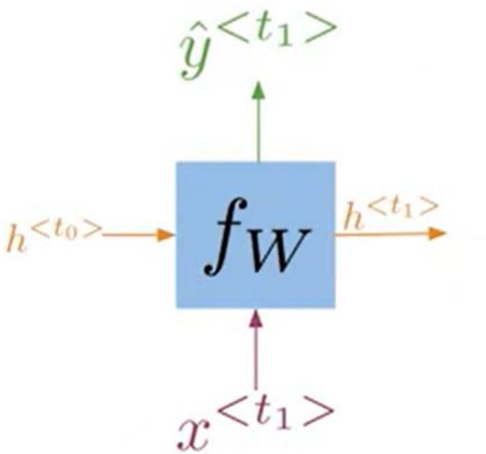
$$\hat{y}^{<t>} = g(W_{yh}h^{<t>} + b_y)$$

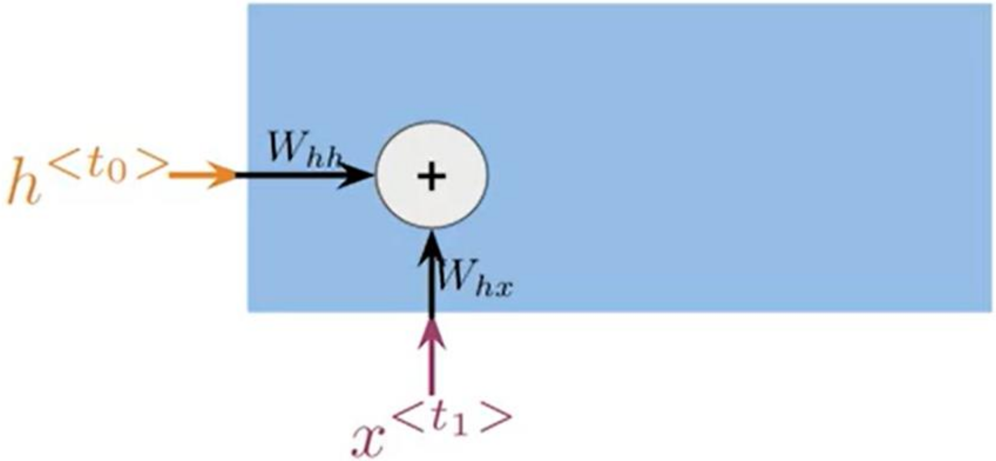
$$\boxed{h^{<t>} = g(W_{hh}h^{<t-1>} \oplus W_{hx}x^{<t>} + b_h)}$$

Maths in simple RNN



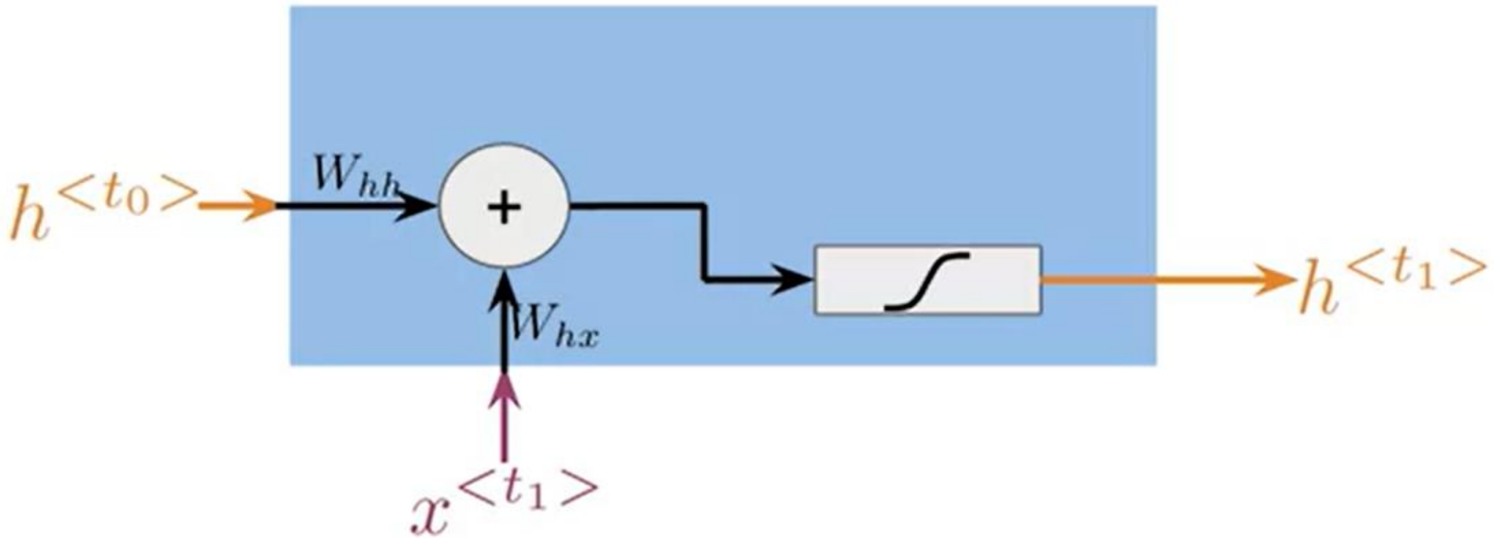
Maths in simple RNN





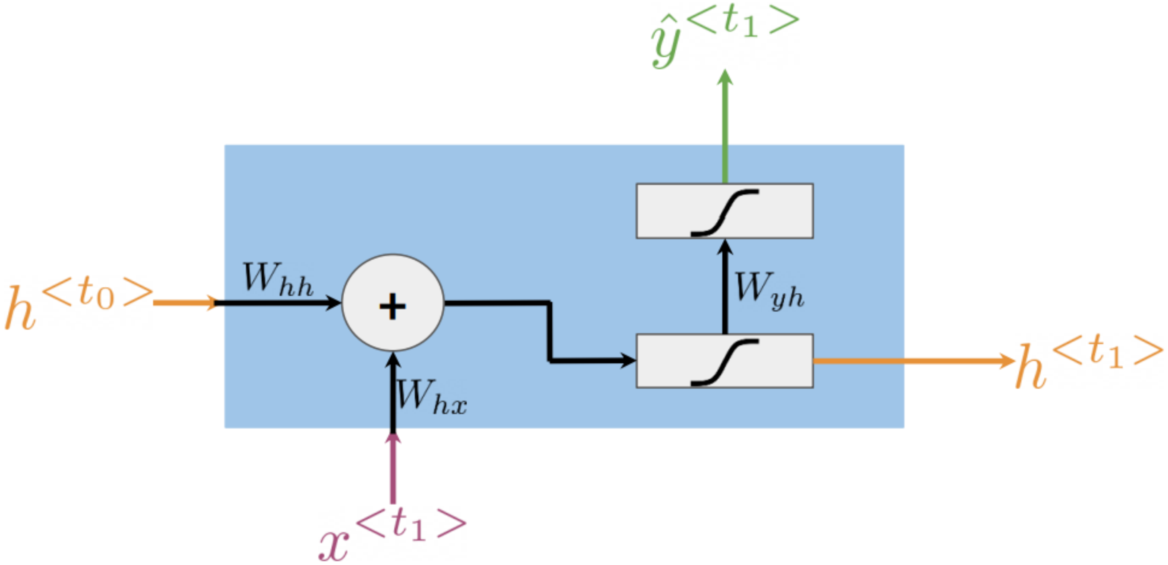
$$h^{<t>} = g(\boxed{W_{hh}h^{<t-1>} + W_{hx}x^{<t>}} + b_h)$$

Maths in simple RNN



$$h^{<t>} = g(W_{hh}h^{<t-1>} + W_{hx}x^{<t>} + b_h)$$

Maths in simple RNN



$$h^{<t>} = g(\boxed{W_{hh}h^{<t-1>} + W_{hx}x^{<t>}} + b_h)$$

$$\hat{y}^{<t>} = g(W_{yh}h^{<t>} + b_y)$$

Note that we end up training with : $W_{hh}, W_{hx}, W_{yh}, b_h, b_y$

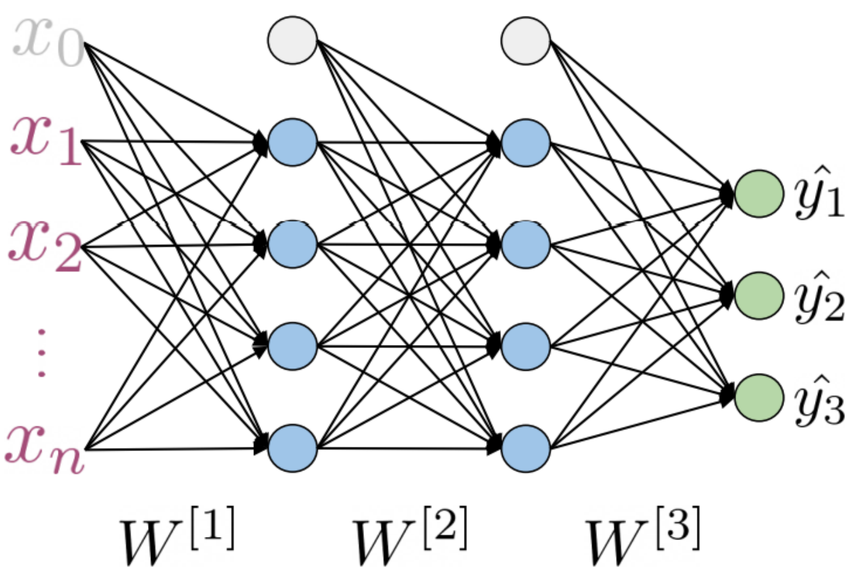
Maths in simple RNN

Summary:

- Hidden states propagate information through time
- Basic recurrent units have two inputs at each time: $h^{<t-1>} \quad x,^{<t>}$

Cost Function for RNNs

Cross Entropy Loss



K - classes or possibilities

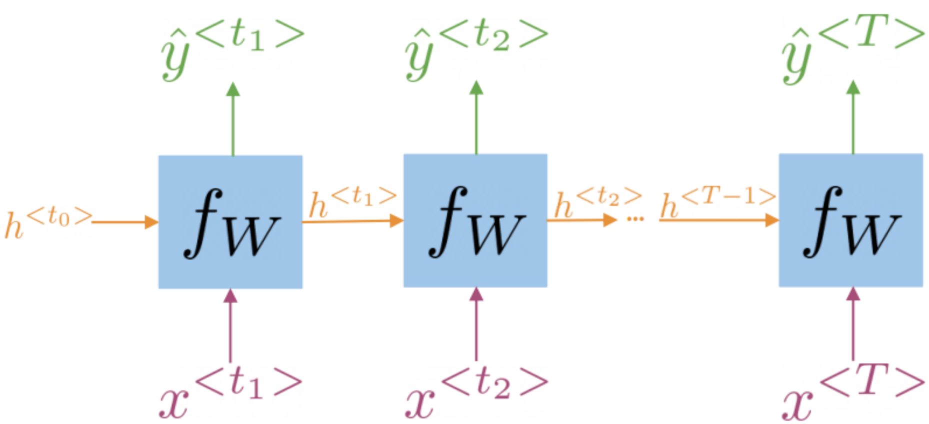
Either 0 or 1

$$J = -\sum_{j=1}^{\boxed{K}} \boxed{y_j} \log \hat{y}_j$$

Looking at a single example (x, y)

Cost Function for RNNs

Cross Entropy Loss



$$h^{<t>} = g(W_h[h^{<t-1>}, x^{<t>}] + b_h)$$

$$\hat{y}^{<t>} = g(W_{yh}h^{<t>} + b_y)$$

$$J = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{K} y_j^{<t>} \log \hat{y}_j^{<t>}$$

Average with respect to time

For RNNs the loss function is just an average through time!