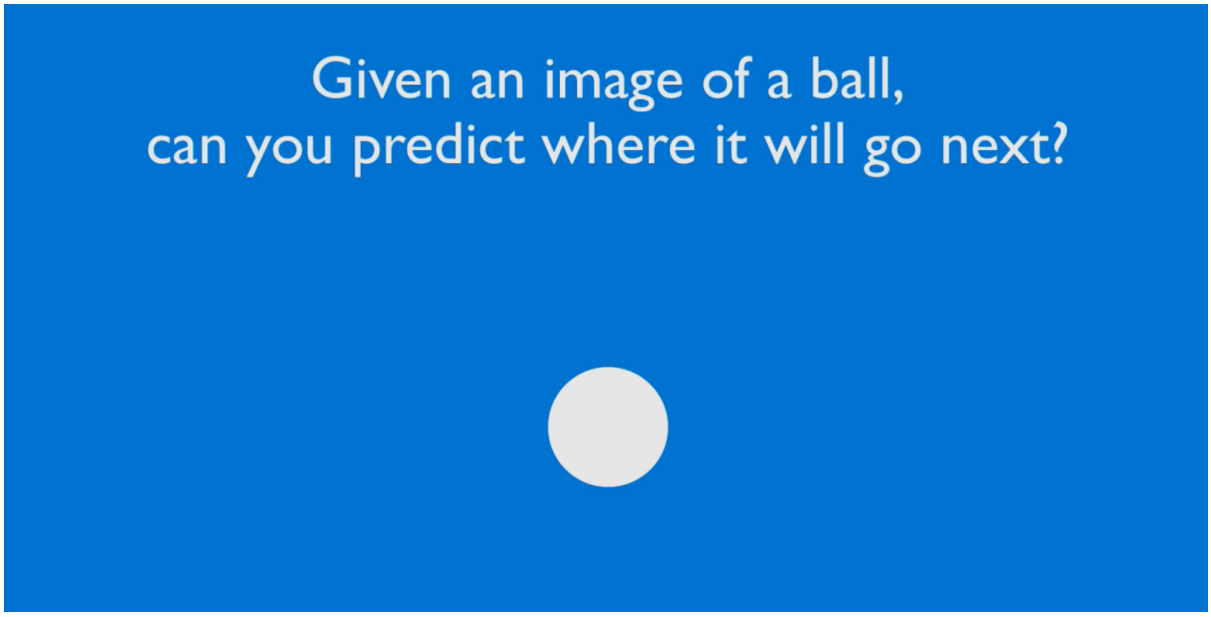
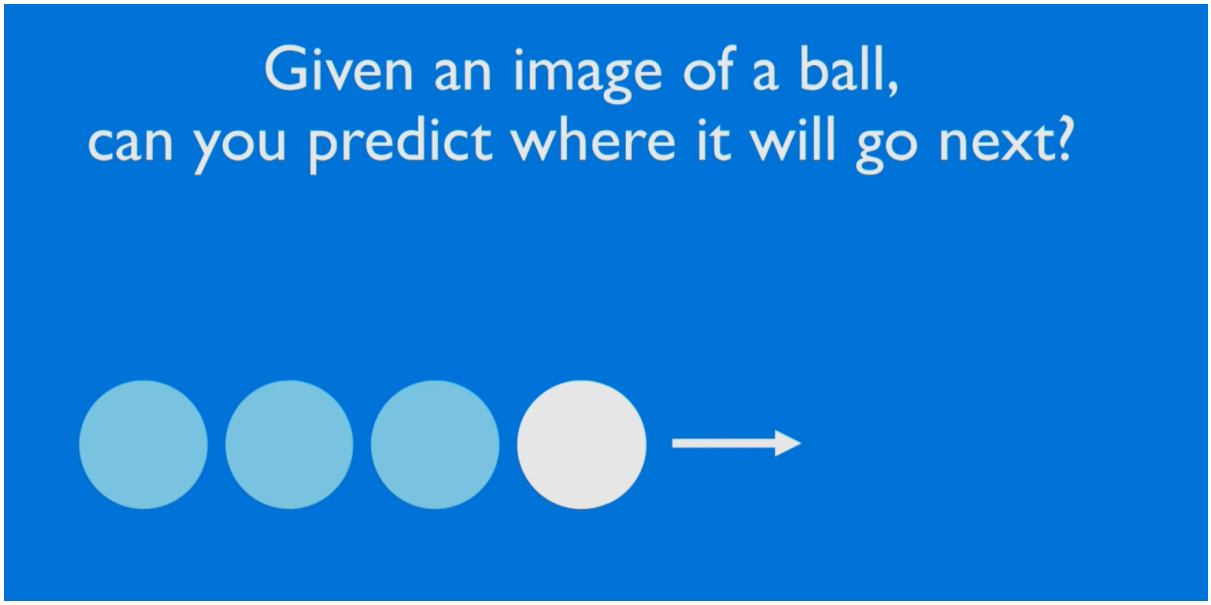
[2] Recurrent Neural Network

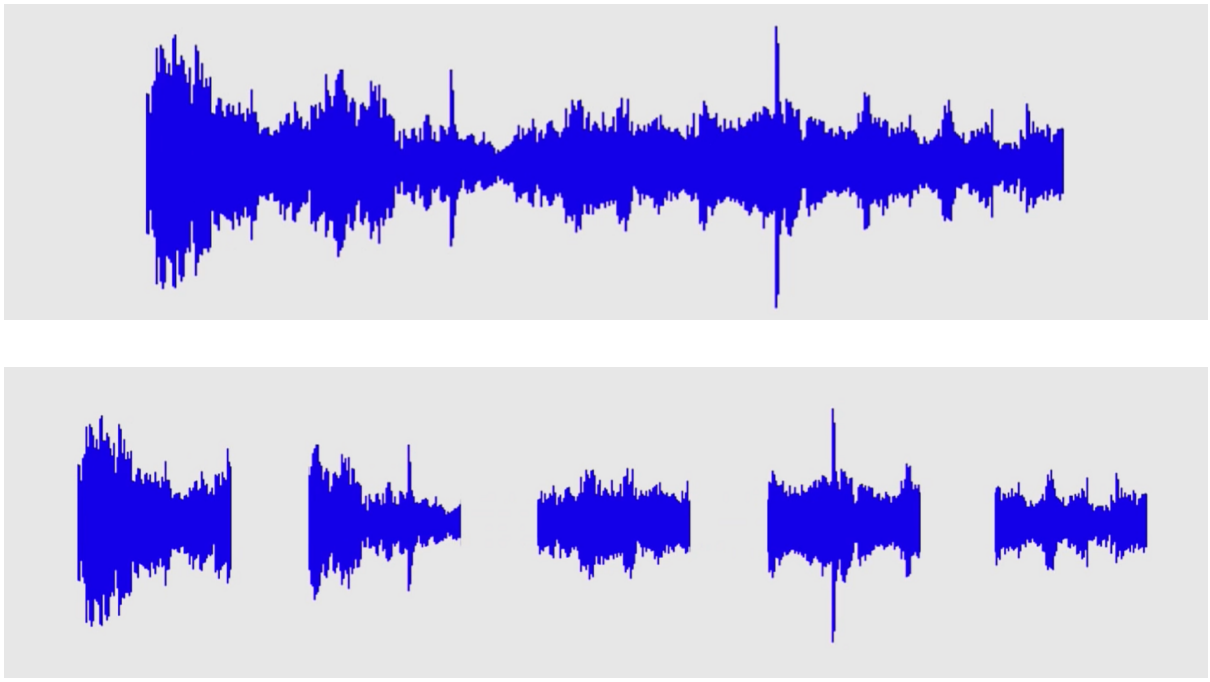
Examples



- If there were no given extra information then it's just a guess
- If previous location is given then it is possible to predict where it will be next



- Sequential data is all around us
  - For example, audio can be split up into multiple sound waves

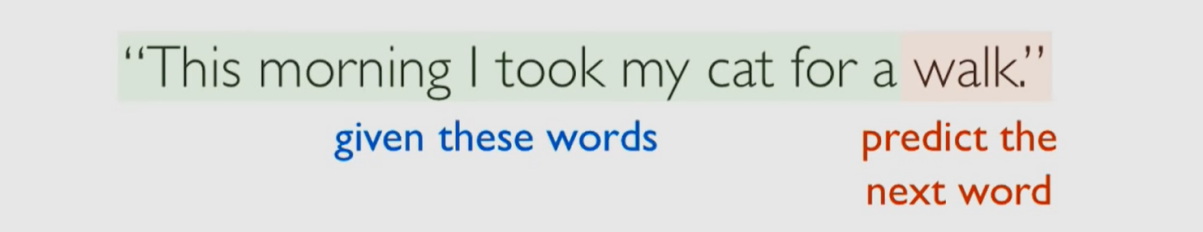


- Text could be split up into into character, words, punctuations and numbers

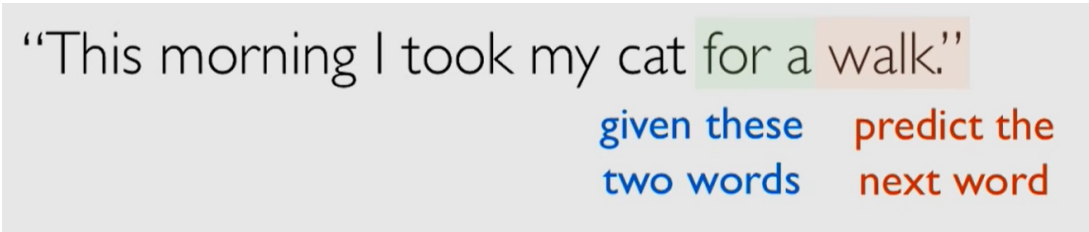
Sequential modeling

Example on text predicting

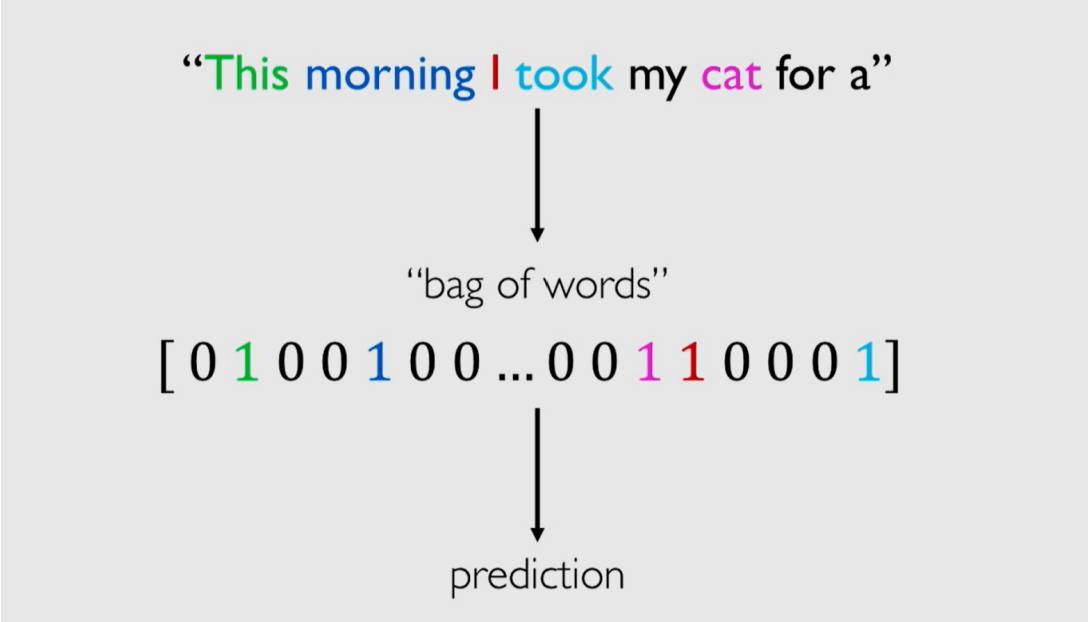
- Here is a string of text, the yellow highlighted words are the text we want to predict



- Issues we encounter
  - The model can only take a fixed length of input vectors, and we have to specify it at the start
- Solution to the issue
  - Fixed window

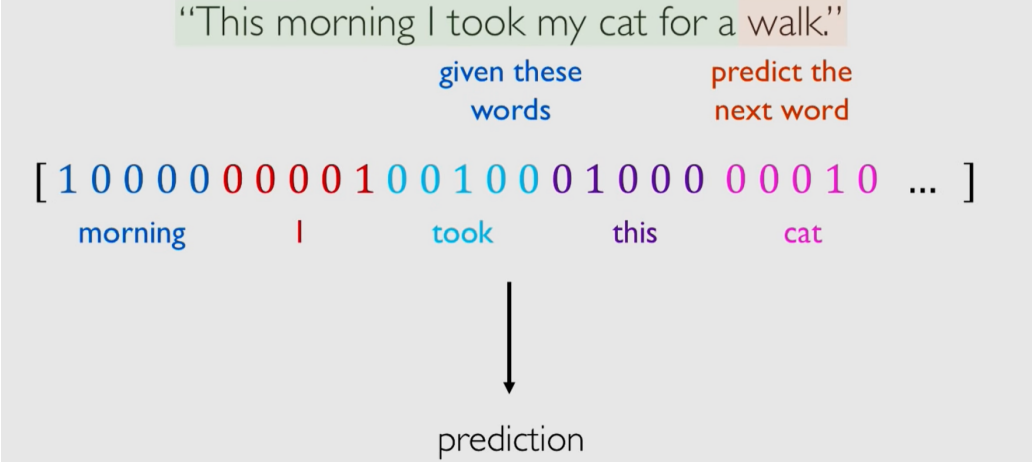


    - Problematic because we cannot see the history of the words
    - We also cannot model long term dependencies
  - Using entire sequence as a set count

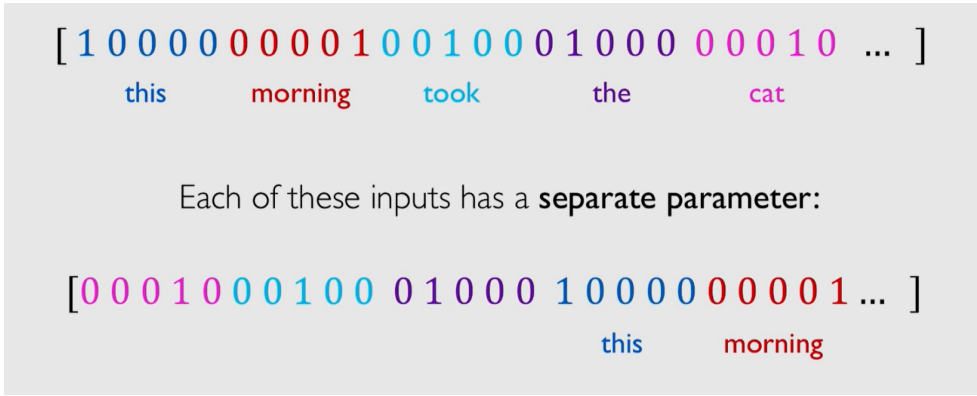


- This is called bag of words
  - Issue with this is that order will not be considered when using counts

- Anther solution could be using a big fixed window

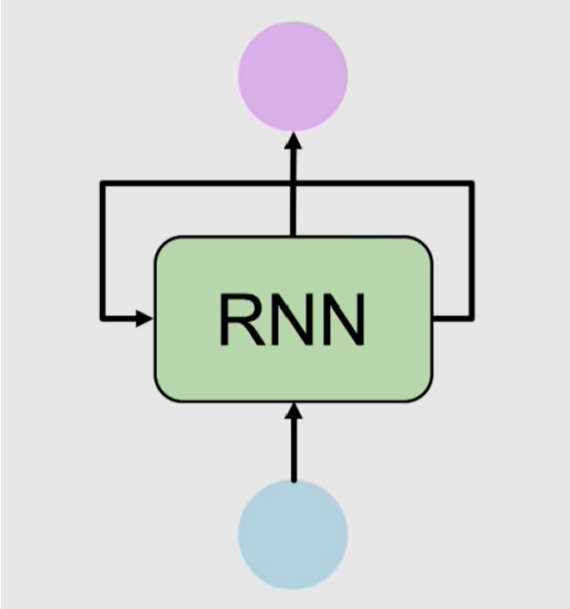


- A issue might be location of the text since it will not recongize if the location of the same words are the same, example given on the image below



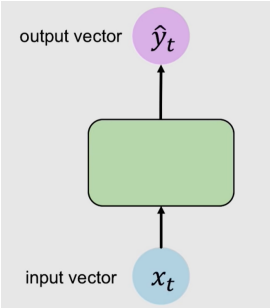
Criteria to Creating a text Prediction Model

- Handle different variable length
- Track long term dependencies in data
- maintain information about order
- share parameter across sequence
- This could be handled through RNN
- Recurrent Neural Networks



Recurrent Neural Network (RNNs)

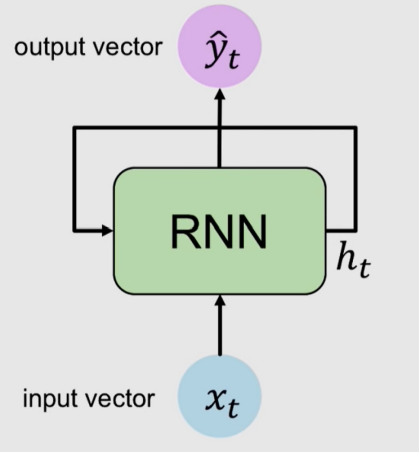
- Standard neural network goes form input to output in one direction
  - not able to maintain information from previous events



- RNN has loops therefore it allows information to persist over time
  - RNN uses a recurrent relation at every time step to process a sequence

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

cell state — function parameterized by W — old state — input vector at time step t



  - Same function and set of parameters are being used

- the equation below shows how the hidden state is updated

Output Vector

$$\hat{y}_t = W_{hy}^T h_t$$

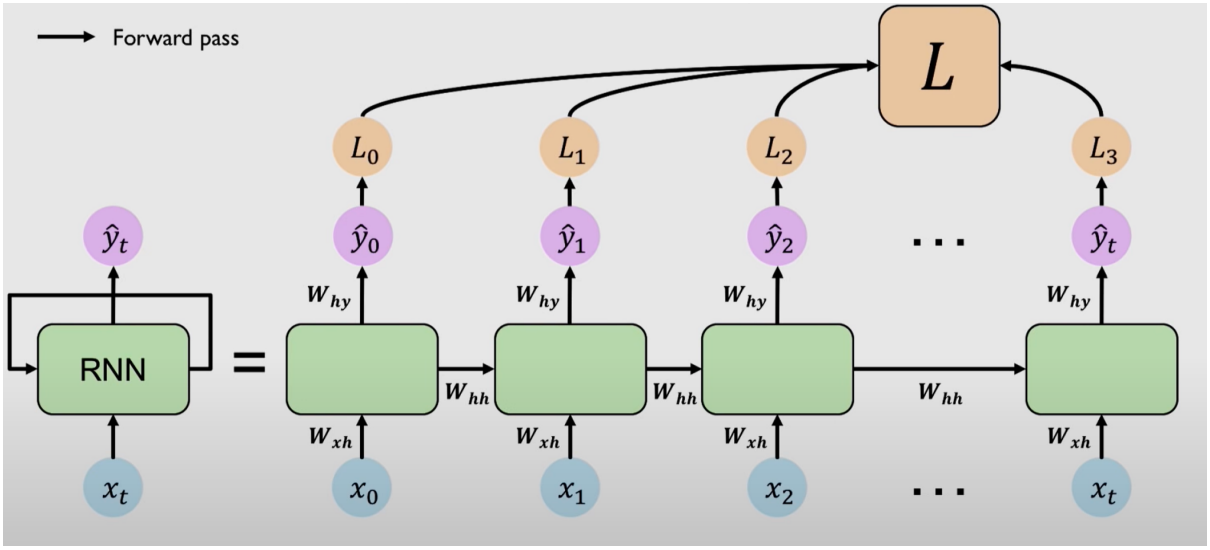
Update Hidden State

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

Input Vector

$$x_t$$

- The graph below shows how RNN works internally
  - It is the same function running over and over but with passing a internal state information down to the next function



- L stands for loss

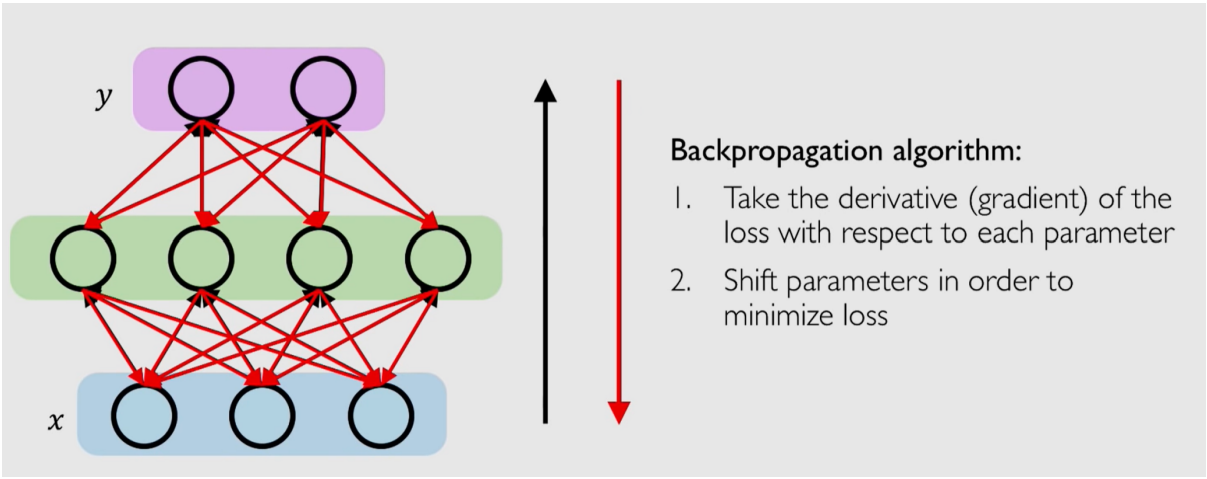
Implementing In TensorFlow

```
tf.keras.layers.SimpleRNN(rnn_units)
```

Gradient Issues

Backpropagation Through Time (BPTT)

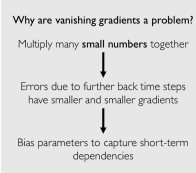
- In order to train the model we have to go back through the network as the graph shows below



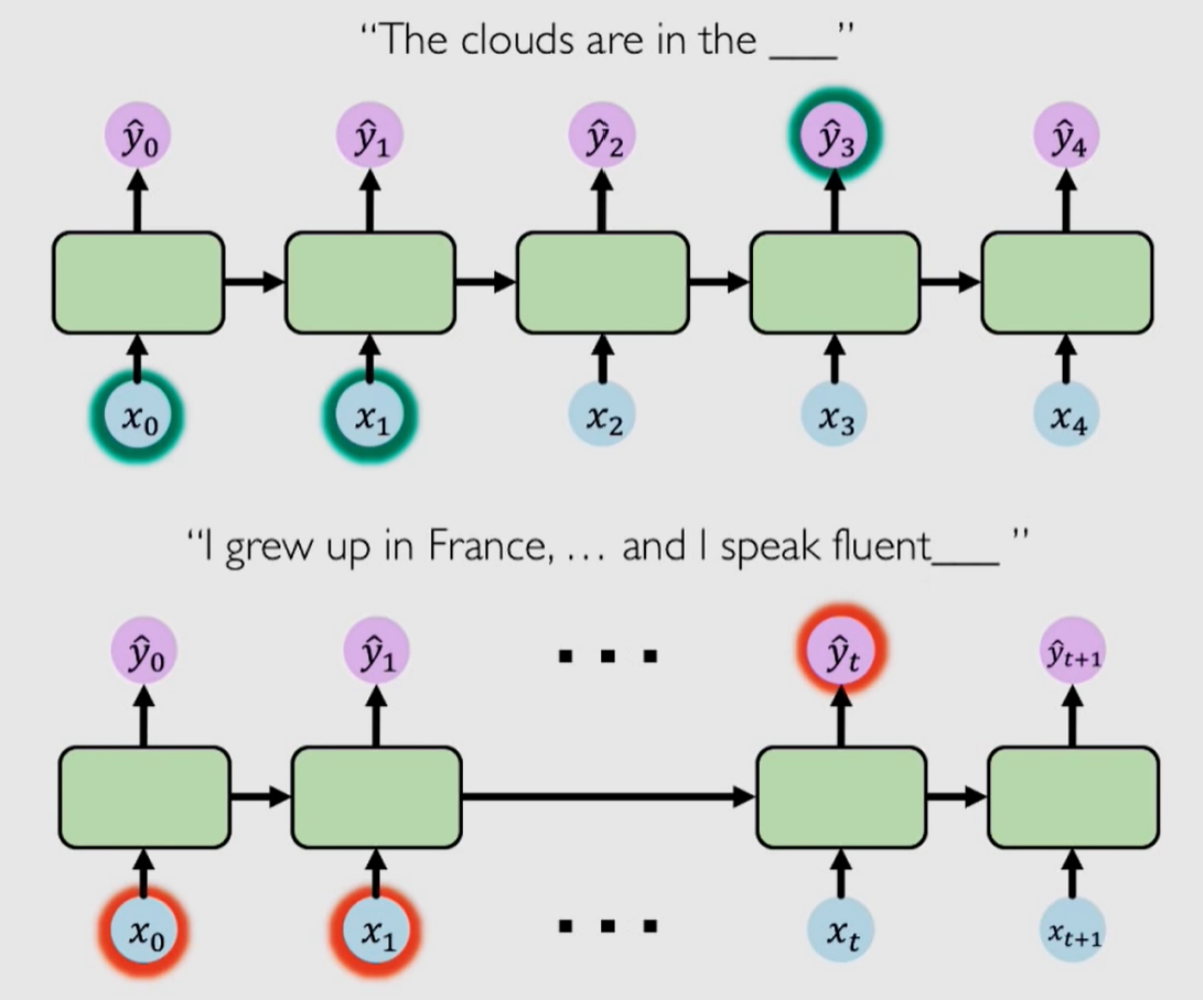
- And computing the gradient h could be problematic because of the many repeated gradient computation:
  - Many Values > 1: Exploding gradient
    - Gradient clipping to scale big gradient
  - Many Values < 1: Vanishing gradient
    - Choosing activation function
    - weight initializing
    - Network architecture

Vanishing gradient

- multiplying small numbers together for a long period of time will result in the number completely vanishing



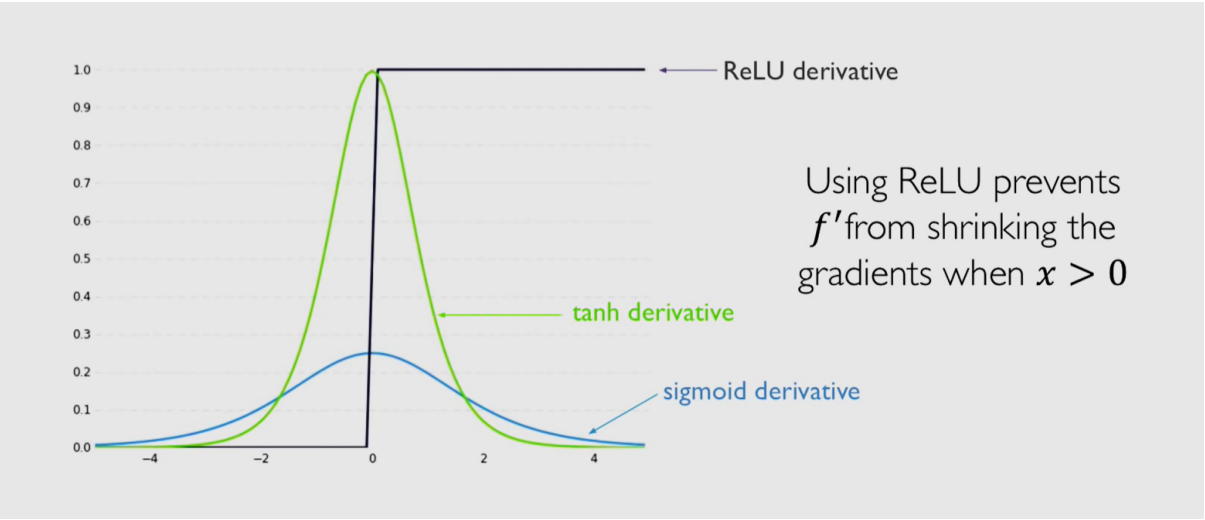
- Sometime this won't be an issue like the first example since all the information we need are in the present close by



- But the second sentence would require information from the past because the model don't know which language it would be

Solution 1: Activation functions

- RELU has only the value of 1 if x is greater than 0 which prevents f' to shrink the gradient when x is greater than 0



Solution 2: Parameter initialization

- Initializing the weights to identity matrix helps to prevent the gradient from shrinking to zero

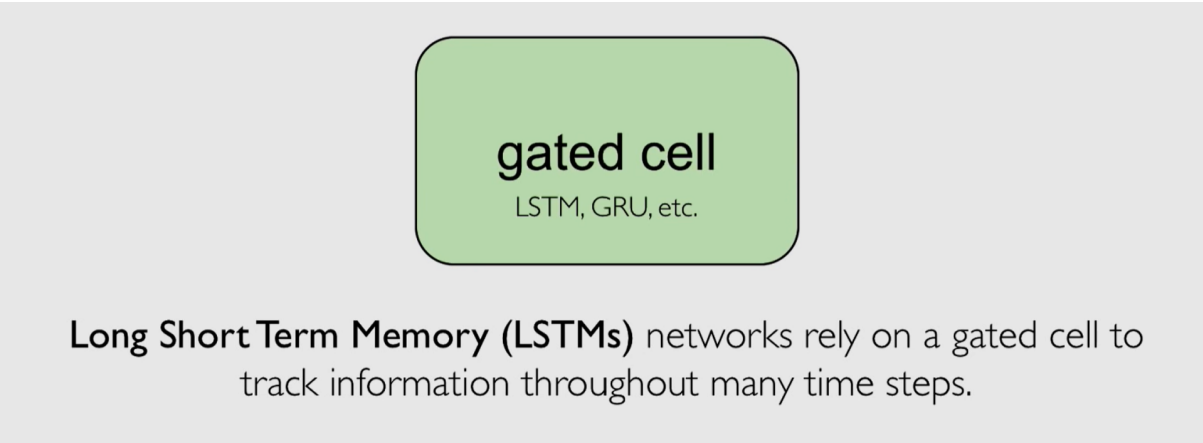
Initialize **weights** to identity matrix

Initialize **biases** to zero

$$I_n = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

Solution 3: Gated Cell

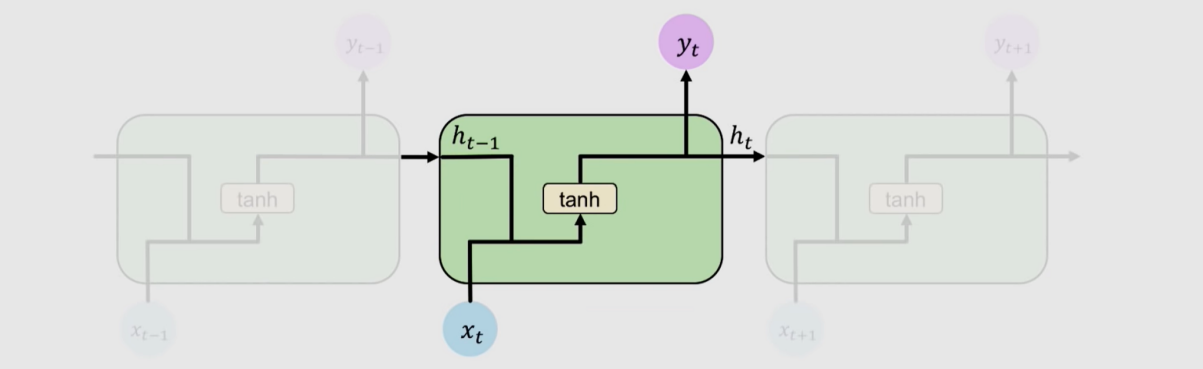
- This is the idea of using a more complex recurrent unit with gates to control what information goes through



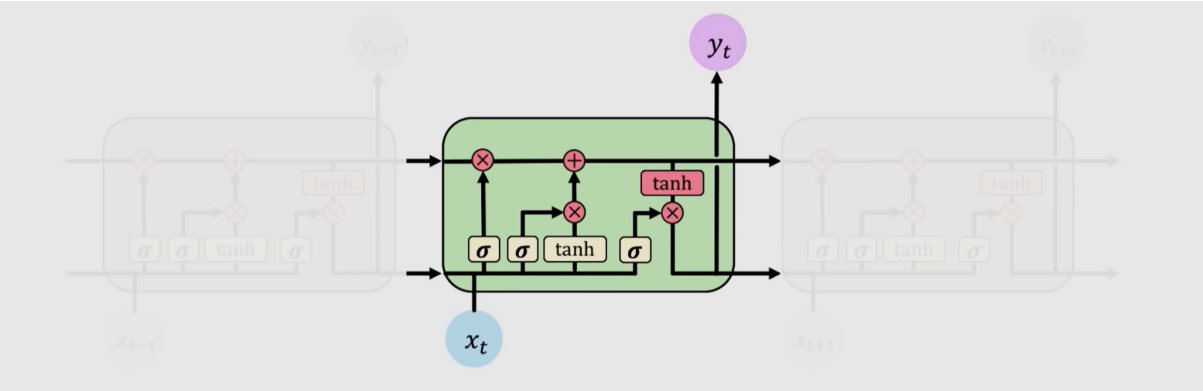
- The one we will be using is Long Short Term Memory (LSTMs)

Long Short Term Memory (LSTMs) Networks

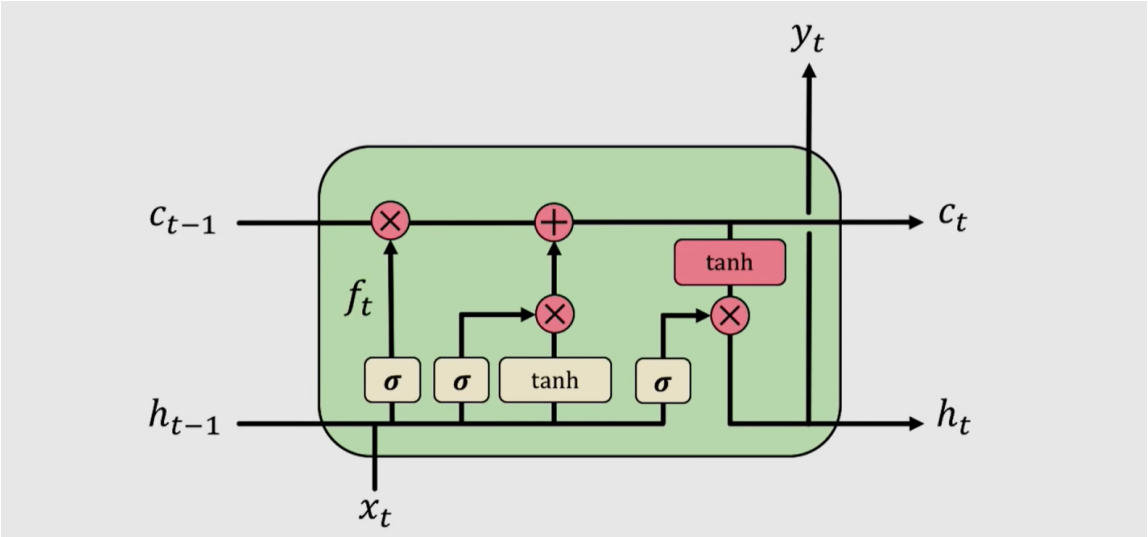
- In RNN, repeating modules contain a simple computation node



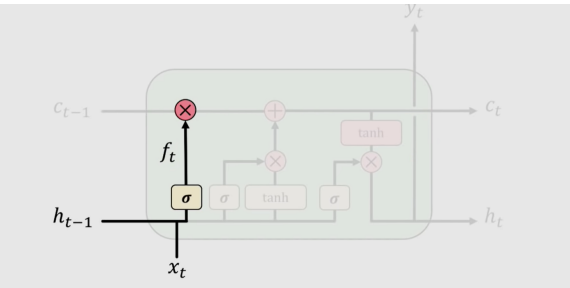
- For LSTM, the layers interact to selectively control the flow of information within the cell



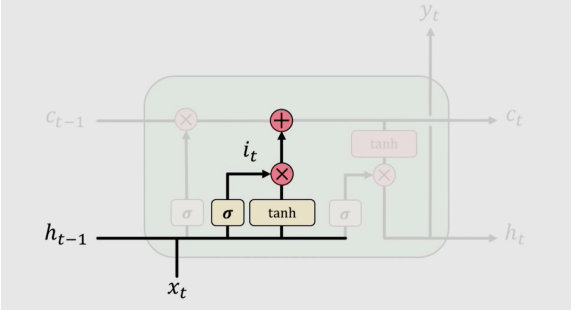
- How does LSTM work?



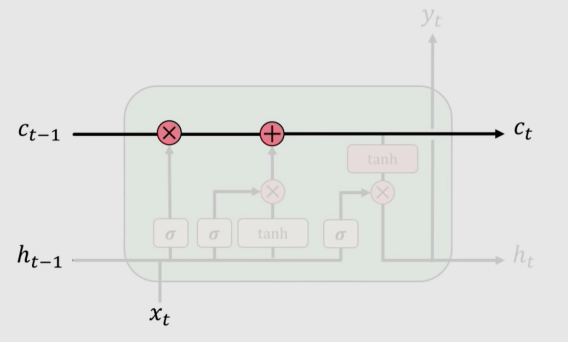
  - Forget
  - Store
  - Update
  - Output



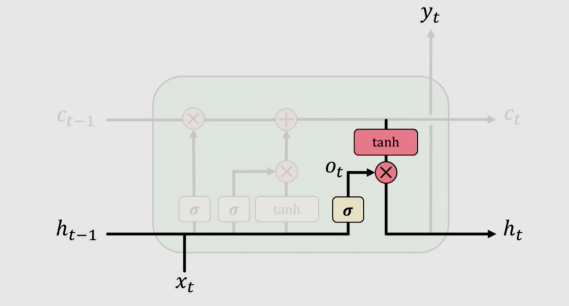
#1: LSTM forget irrelevant parts of previous state



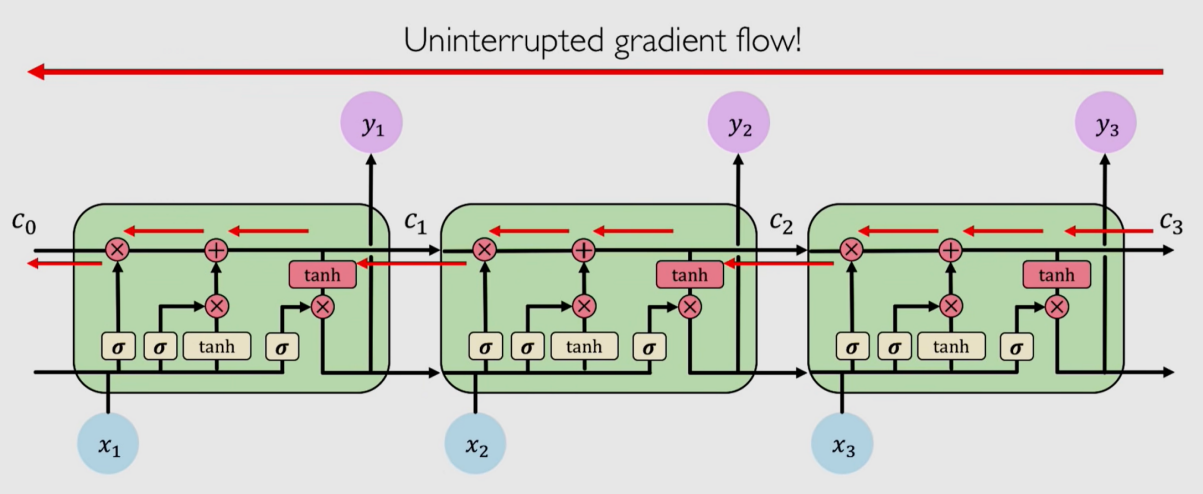
#2: LSTM stores relevant new information into cell state



#3: LSTM selectively update cell state values



#4: The output controls what gets sent to the next time step



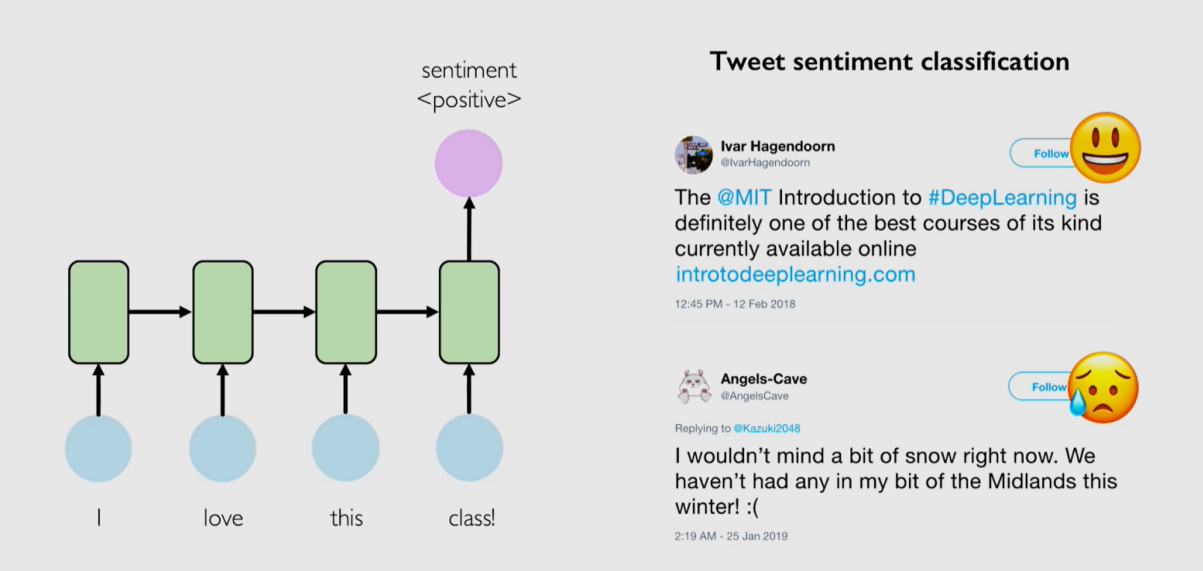
- LSTM creates a uninterrupted flow of gradient so it's easier for training

LSTM Key Concepts

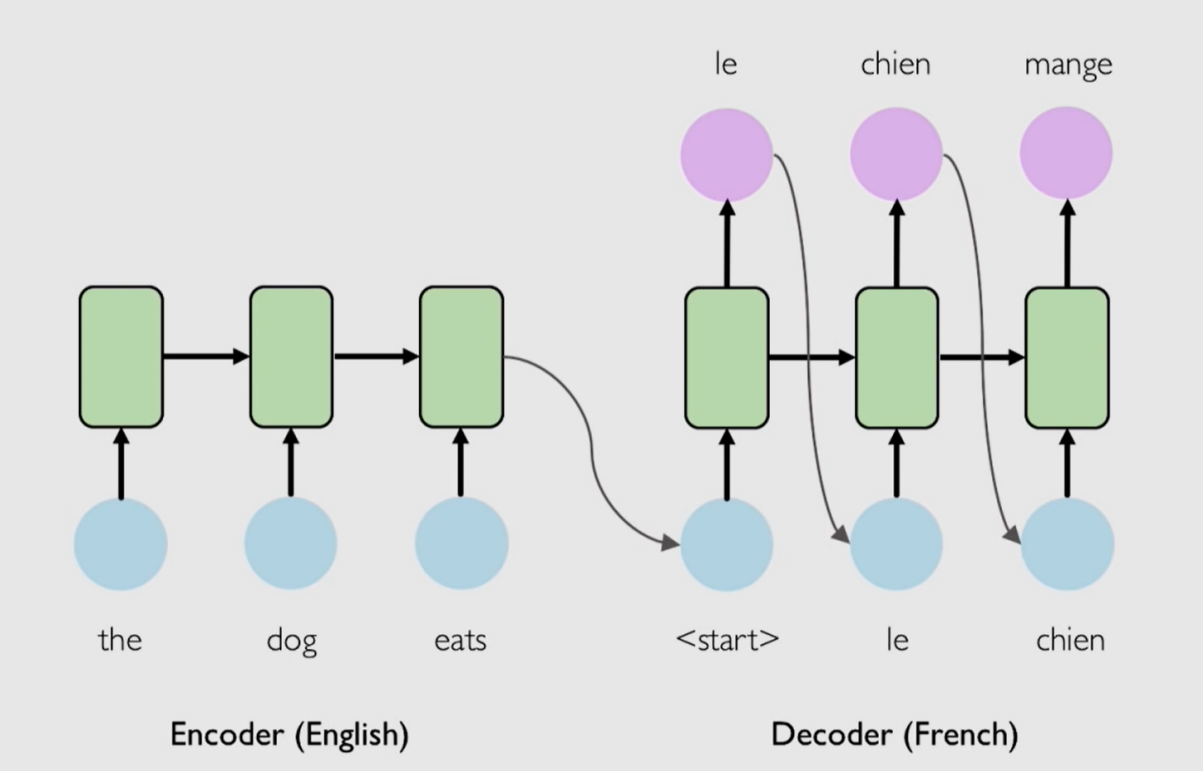
- Maintain separate cell state from what is outputted
- Use gates to control the flow of information
  - Forget gates get rid of irrelavent information
  - Store relevant information from current output
  - selectively update cell state
  - Output gate returns a filtered version of the cell state
- Backpropagation through time with uninterrupted gradient flow

RNN Application

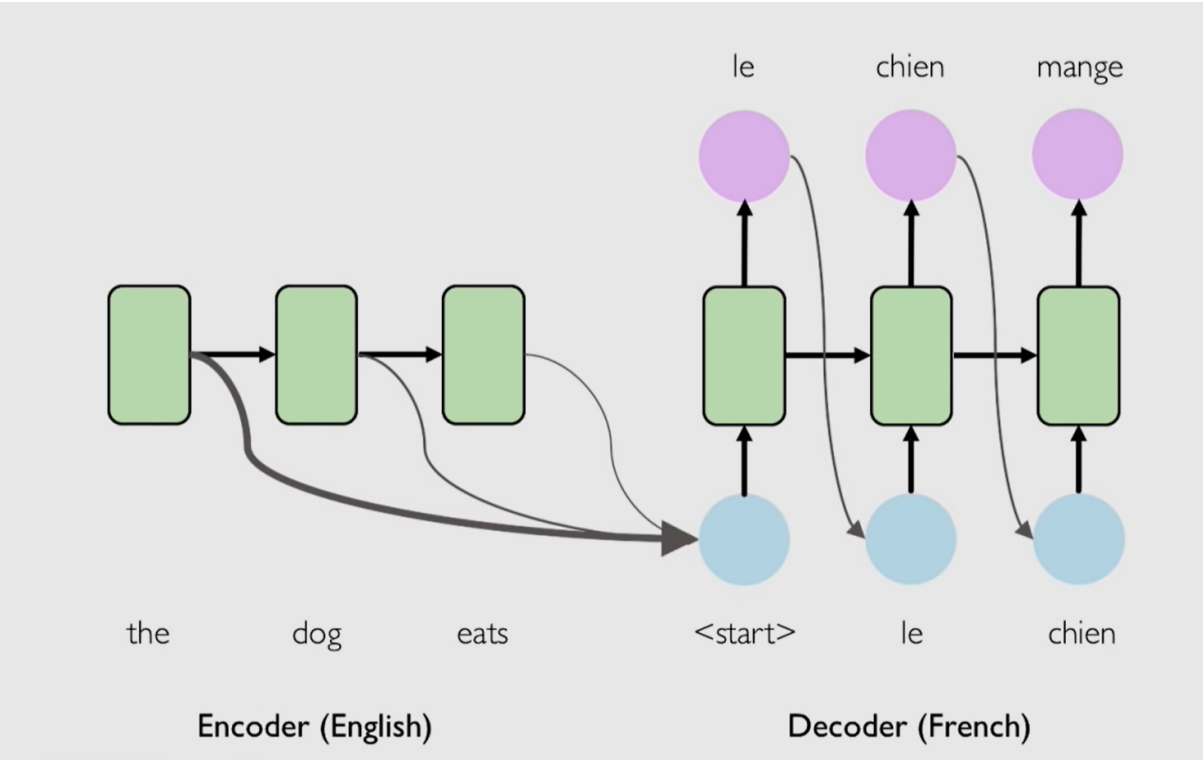
- Music generation
  - input music sheet and output next character of the node
- Sentiment classifcation
  - input sequence of words and output probability of having a positive statement



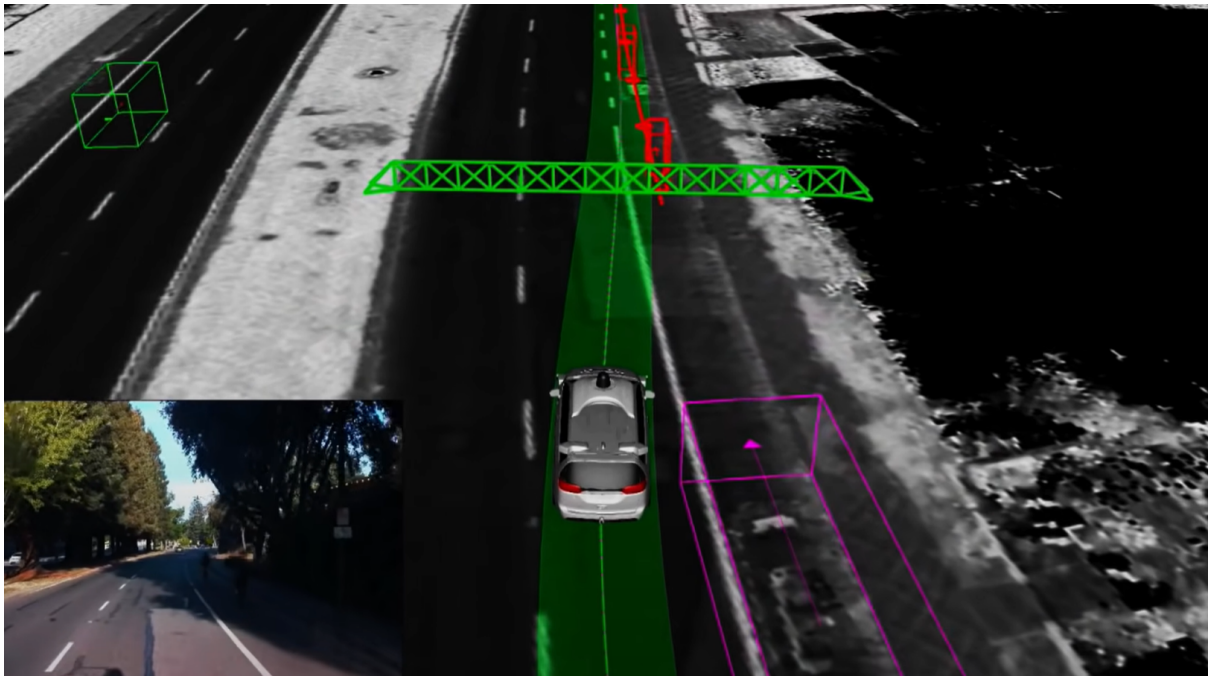
- Machine Translation
  - input a language and output another language sentence of the same meaning



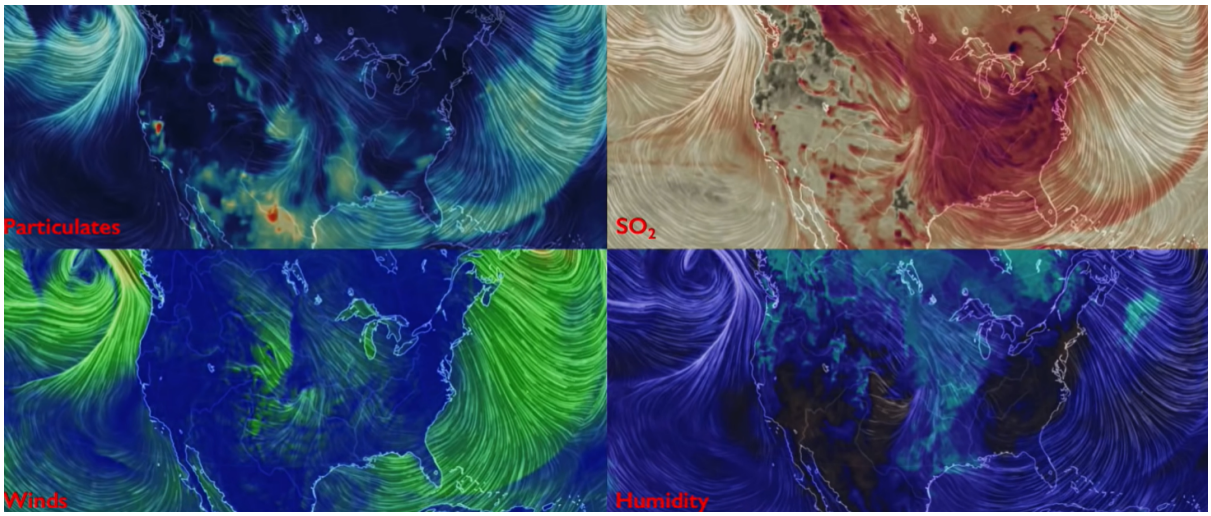
- To Avoid bottleneck with large amount of text, google developed attention mechanism to deal with the issue



- Instead of having to access the final encoded state, it could access all the state of the time steps
- attention mechanism in neural networks provide learnable memory access
- Self Driving cars use trajectory prediction to predict what the cars around it would do



- Another instance where this could be helpful is in weather predictions / enjoinment modeling



Summary

1. RNNs are well suited for **sequence modeling** tasks
2. Model sequences via a **recurrence relation**
3. Training RNNs with **backpropagation through time**
4. Gated cells like **LSTMs** let us model **long-term dependencies**
5. Models for **music generation**, classification, machine translation, and more