CS224N : Lecture 6 - Simple and LSTM RNNs

1. RNN Language Models

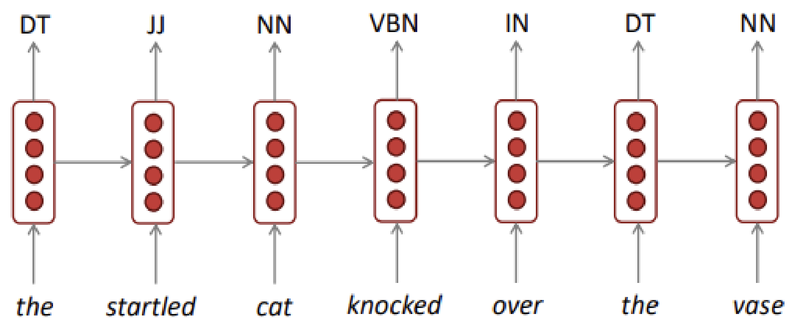
- **Language Model**: A system that predicts the next word
- **Recurrent Neural Network**: A family of neural networks that:
    - Take sequential input of any length
    - Apply the same weights on each step
    - Can optionally produce output on each step
- Recurrent Neural Network ≠ Language Model
- We've shown that RNNs are a great way to build a LM
- But RNNs are useful for much more!

2. Other uses of RNNs

▼ Slides

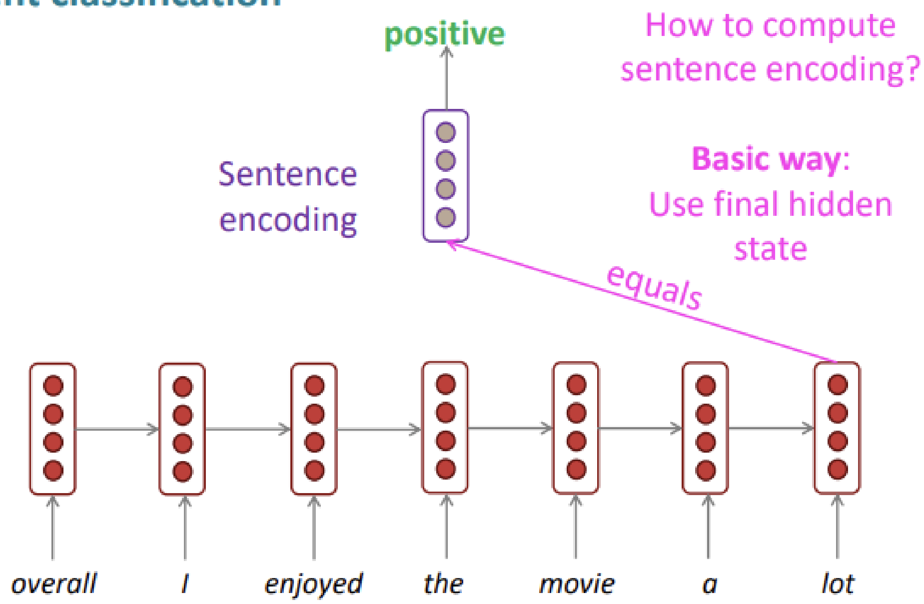
2. Other RNN uses: RNNs can be used for sequence tagging

e.g., **part-of-speech tagging**, named entity recognition



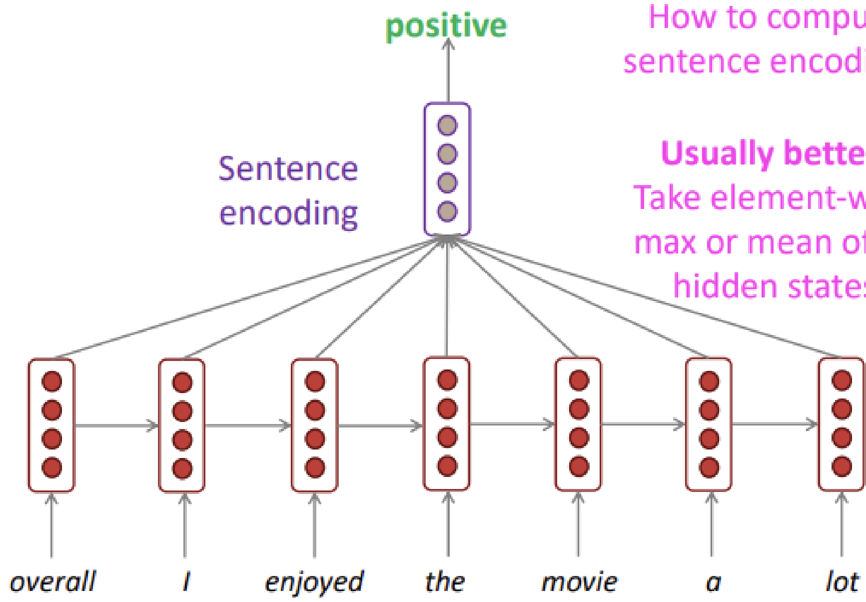
RNNs can be used for sentence classification

e.g., **sentiment classification**



RNNs can be used for sentence classification

e.g., **sentiment classification**



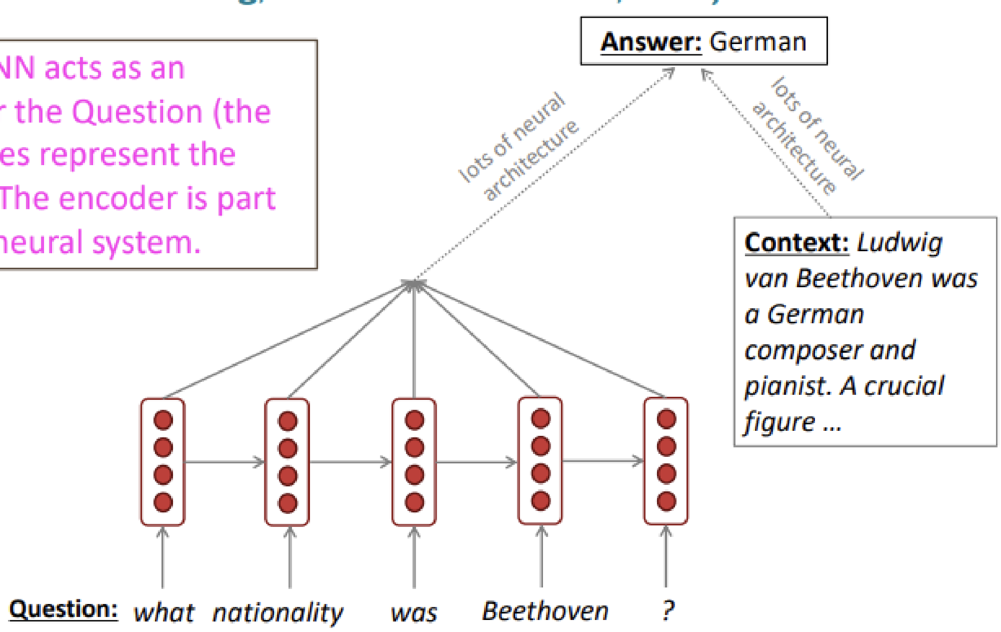
How to compute sentence encoding?

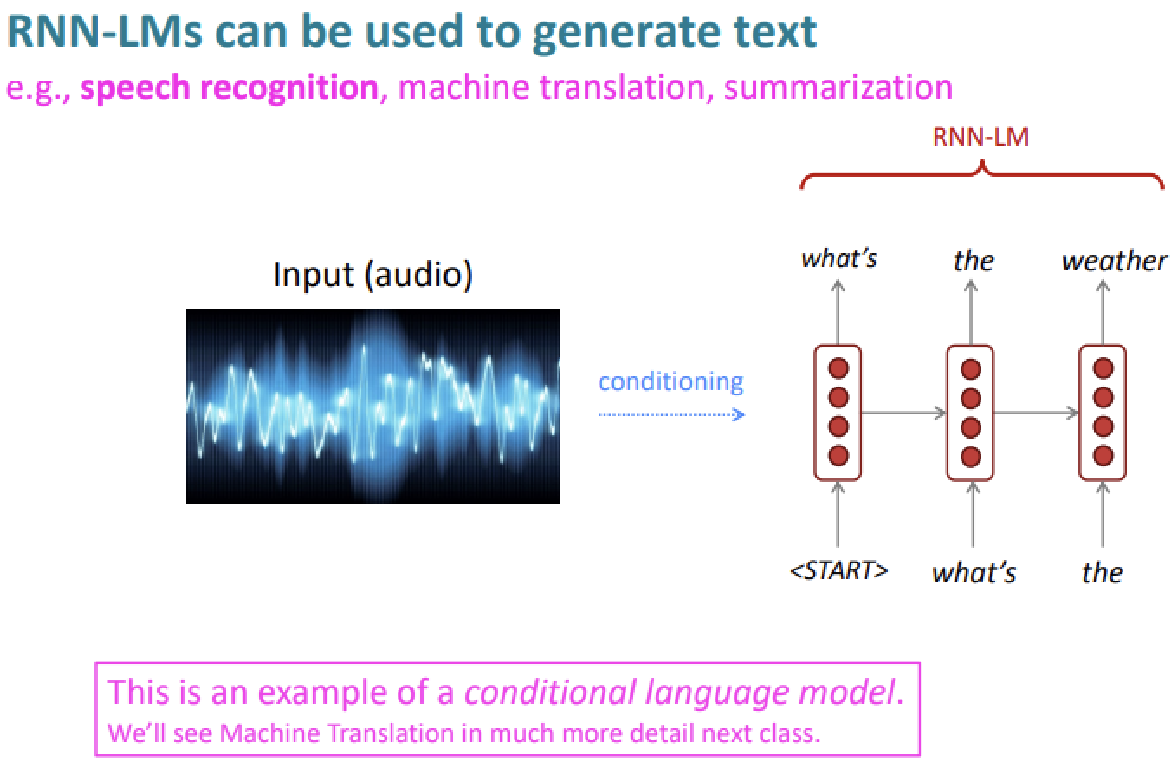
Usually better: Take element-wise max or mean of all hidden states

RNNs can be used as a language encoder module

e.g., **question answering**, machine translation, *many other tasks!*

Here the RNN acts as an **encoder** for the Question (the hidden states represent the Question). The encoder is part of a larger neural system.





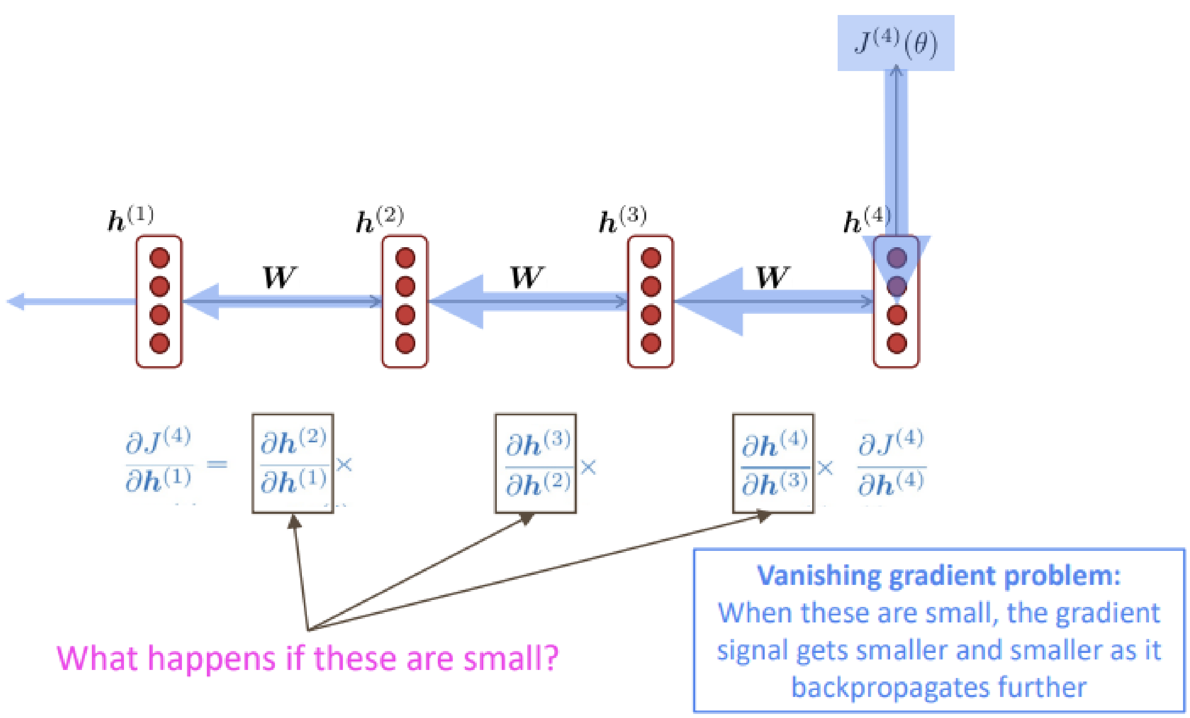
3. Exploding and vanishing gradients

- Vanishing Gradients
- Why is vanishing gradient a problem?
    - Gradient signal from far away is lost because it's much smaller than gradient signal from close-by. So, model weights are updated only with respect to near effects, not long-term effects.
- Effect of vanishing gradient on RNN-LM
    - RNN-LM needs to model the dependency
    - But if the gradient is small, the model can't learn this dependency
    - So, the model is unable to predict similar long-distance dependencies at test time
- Why is exploding gradient a problem?
    - If the gradient becomes too big, then the SGD update step becomes too big

- This can cause bad updates: we take too large a step and reach a weird and bad parameter configuration (with large loss)
- In the worst case, this will result in Inf or NaN in your network (then you have to restart training from an earlier checkpoint)

- Gradient clipping
  - solution for exploding gradient
  - Gradient clipping: if the norm of the gradient is greater than some threshold, scale it down before applying SGD update
  - Intuition: take a step in the same direction, but a smaller step
  - In practice, remembering to clip gradients is important, but exploding gradients are an easy problem to solve
- How to fix the vanishing gradient problem?
  - The main problem : it's too difficult for the RNN to learn to preserve information over many timesteps.
  - In a vanilla RNN, the hidden state is constantly being rewritten

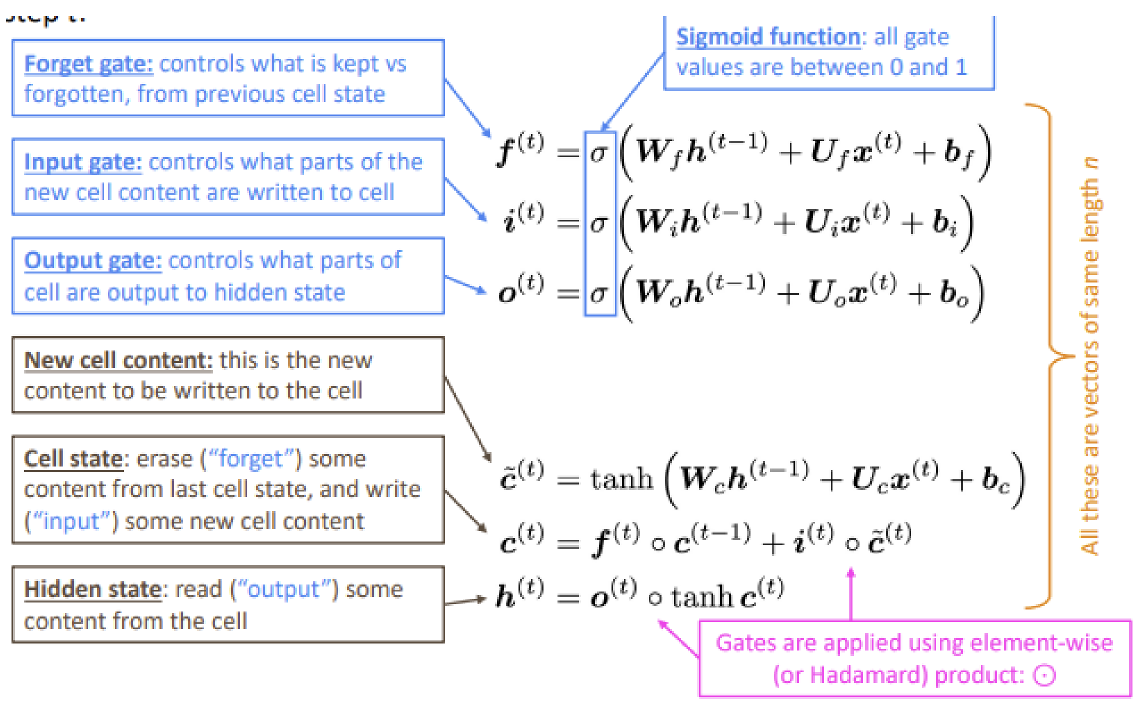
▼ Slides

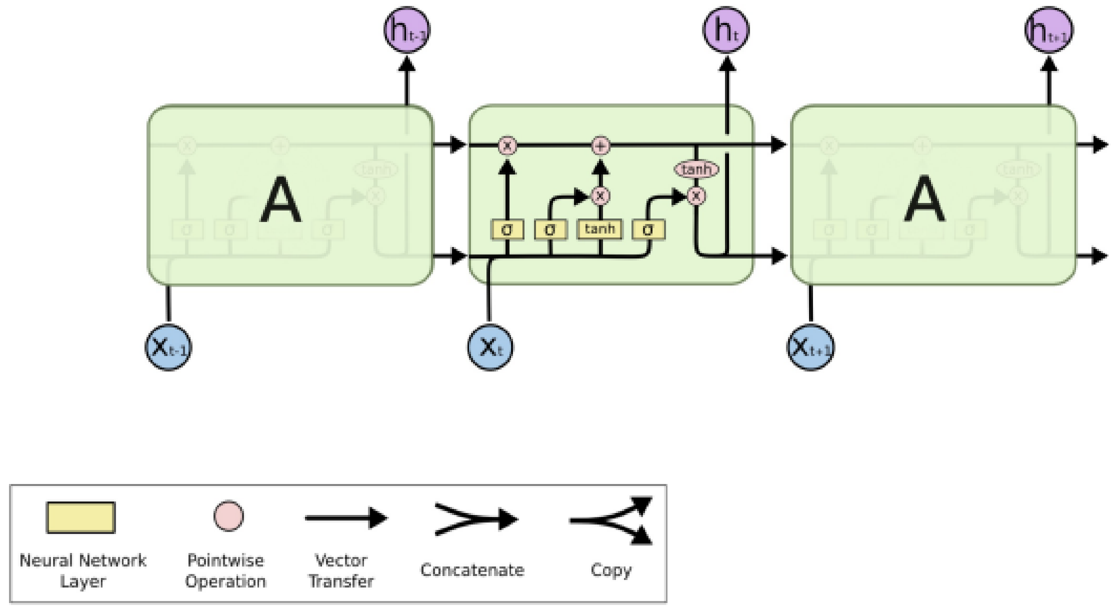


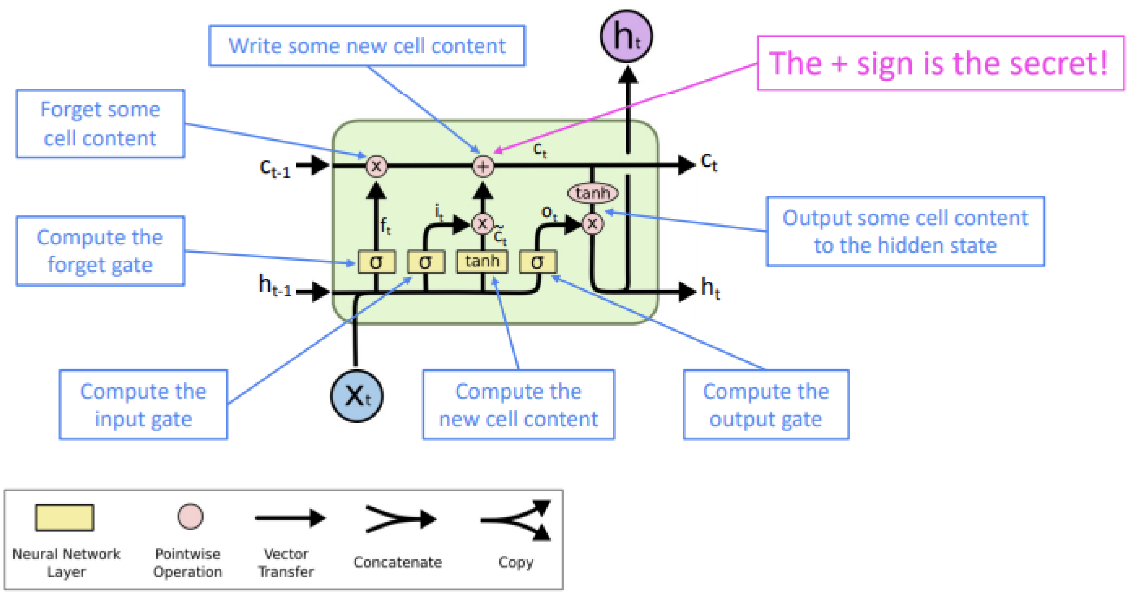
4. LSTMs

- Long Short-Term Memory RNNs (LSTMs)
    - A type of RNN proposed by Hochreiter and Schmidhuber in 1997 as a solution to the vanishing gradients problem

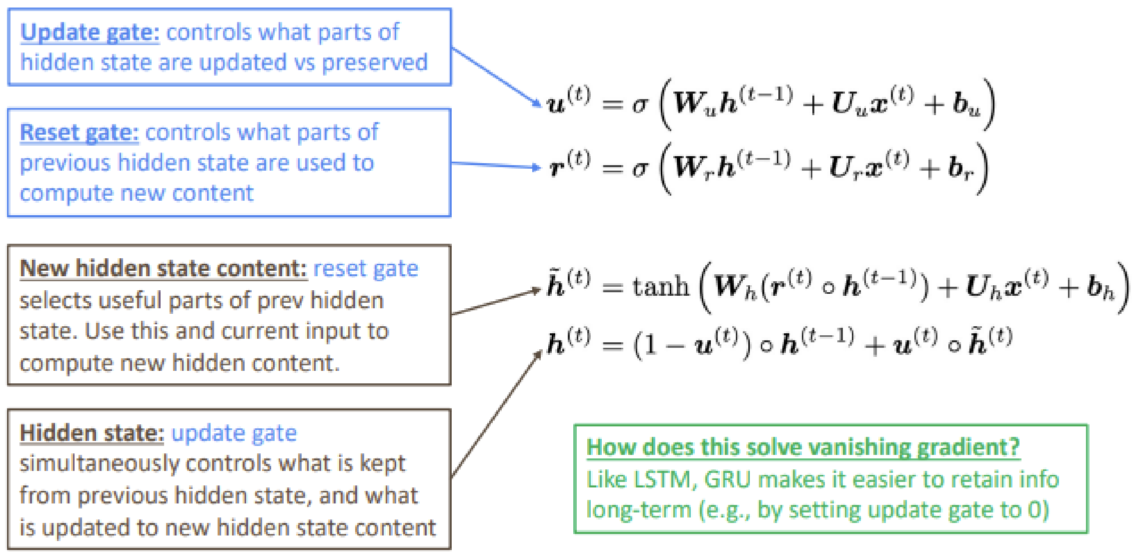
▼ Slides







- Gated Recurrent Units (GRU)
  - Proposed by Cho et al. in 2014 as a simpler alternative to the LSTM.
  - On each timestep t we have input and hidden state (no cell state)
  - ▼ Slides



- LSTM vs GRU
  - Researchers have proposed many gated RNN variants, but LSTM and GRU are the most widely-used
  - Rule of thumb: LSTM is a good default choice (especially if your data has particularly long dependencies, or you have lots of training data); Switch to GRUs for speed and fewer parameters.
  - LSTMs can store unboundedly large values in memory cell dimensions, and relatively easily learn to count. (Unlike GRUs.)
- How does LSTM solve vanishing gradients?
  - The LSTM architecture makes it easier for the RNN to preserve information over many timesteps
  - LSTM doesn't guarantee that there is no vanishing/exploding gradient, but it does provide an easier way for the model to learn long-distance dependencies
- Is vanishing/exploding gradient just a RNN problem?
  - No! It can be a problem for all neural architectures (including feed-forward and convolutional), especially very deep ones.
  - Solution: lots of new deep feedforward/convolutional architectures add more direct connections (thus allowing the gradient to flow)

- Conclusion: Though vanishing/exploding gradients are a general problem, RNNs are particularly unstable
  due to the repeated multiplication by the same weight matrix [Bengio et al, 1994]

5. Bidirectional and multi-layer RNNs

- Bidirectional RNNs
  - Bidirectional RNNs are only applicable if you have access to the entire input sequence
  - If you do have entire input sequence (e.g., any kind of encoding), bidirectionality is powerful (you should use it by default).