Lecture 3: Recurrent Neural Networks - II

COMP 5801H/4900A: Generative AI and LLMs

2026-01-13

Sriram Subramanian
Assistant Professor & Canada Research Chair, Carleton University
Faculty Affiliate, Vector Institute for Artificial Intelligence
Faculty Affiliate, Schwartz Reisman Institute for Technology and Society

Carleton University

VECTOR INSTITUTE

SCHWARTZ REISMAN INSTITUTE FOR TECHNOLOGY AND SOCIETY

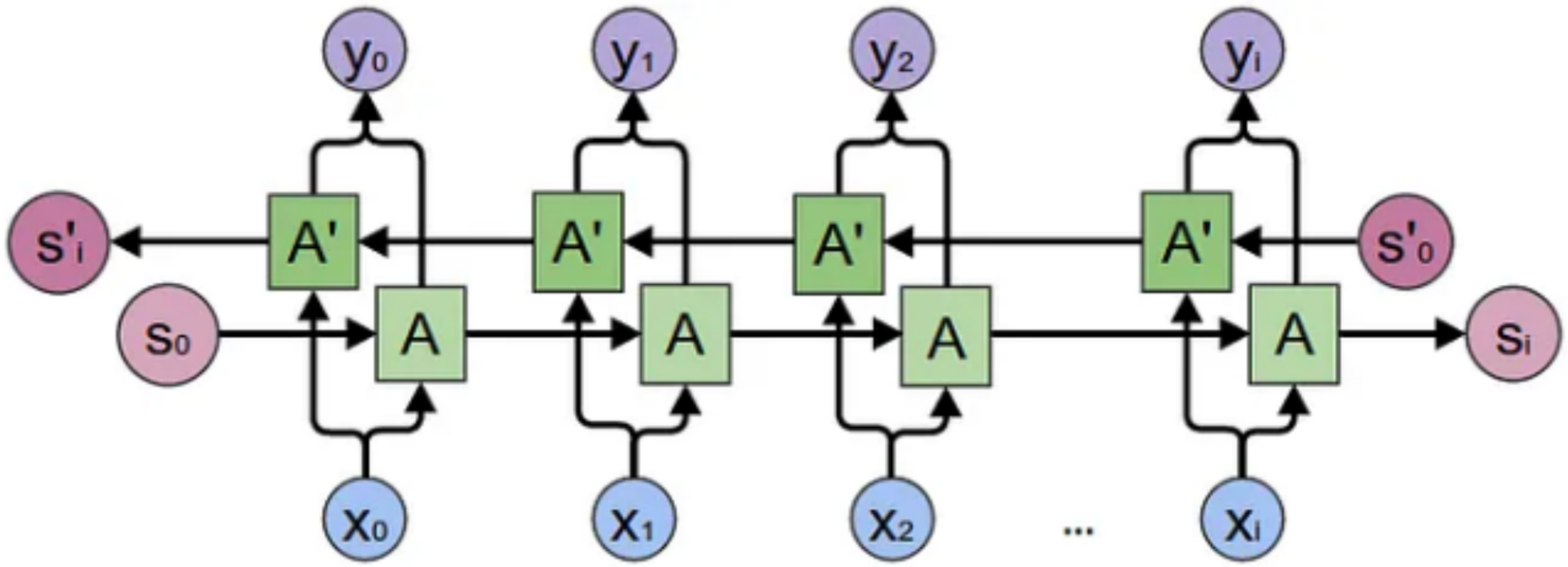
Outline

- Bidirectional RNNs
- Deep (Stacked) RNNs

Bidirectional RNNs (Bi-RNNs) - Motivation

- **Limitation**: Standard RNNs are “causal”. To understand word t, they only use words 0 to $t-1$
- **Problem**: In language, the meaning of a word often depends on what comes *after* it
- **Example**: "The crane flew away" vs. "The crane lifted the steel" You don't know what "crane" means until you see the verb following it
- **The Solution**: Process the sequence in two directions simultaneously

Architecture



Credit: https://medium.com/data-science/understanding-bidirectional-rnn-in-pytorch-5bd25a5dd66

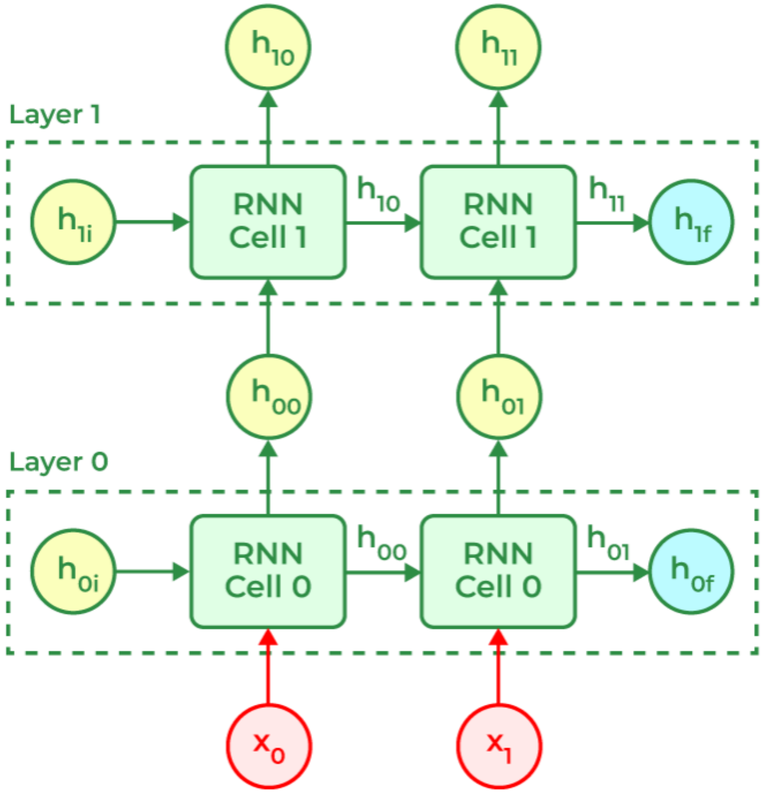
How it works? - Two Independent Streams

- **Forward Layer ($\overrightarrow{h}_t$):** Processes the sequence from start to finish **($x_0 \rightarrow x_t$)**
- **Backward Layer ($\overleftarrow{h}_t$):** Processes the sequence from finish to start ($x_t \leftarrow x_0$)
- **Concatenation:** At every time step t, we combine the two hidden states to get the final representation $h_t = [\overrightarrow{h}_t, \overleftarrow{h}_t]$

Key Characteristics

- **Parameters**: Doubles the number of weight matrices - forward and backward layers have their own unique W_{hh} and W_{xh}
- **Computational Cost**: Twice as expensive to compute as a unidirectional RNN
- **Constraint**: Cannot be used for real-time (online) stream processing because you need the "end" of the sentence before you can compute the first backward state

Deep (Stacked) RNNs



Credit: https://www.geeksforgeeks.org/nlp/stacked-rnns-in-nlp/

Deep (Stacked) RNNs - Increasing Model Capacity

- **Shallow vs. Deep**: A standard RNN is technically "deep" in time but "shallow" in its transformation of a single time step
- **The Layered Approach:** We stack multiple RNN layers on top of each other. The output ($h_t^{(0)}$) of the first layer becomes the input (x_t) for the second layer
- **Why do this?** Just as adding layers to an MLP allows it to learn more complex, hierarchical features, stacking RNNs allows the model to capture different levels of temporal abstraction

The Architecture: Hierarchical Processing

- **Lower Layers**: Typically capture low-level features (e.g., specific words or characters)
- **Higher Layers**: Capture more abstract, global features (e.g., semantic meaning or sentence structure)
- **Final Output**: The prediction (y_t) is usually derived from the hidden state of the top-most layer

Mathematical Representation

- For a 2-layer RNN, the hidden state at layer l and time t is:

$$h_t^{(l)} = \sigma\left(W_{hh}^{(l)} h_{t-1}^{(l)} + W_{xh}^{(l)} h_t^{(l-1)} + b^{(l)} \right)$$

- Note that $h_t^{(0)}$ is simply the original input x_t

Trade-offs

- **Pros**: Can model significantly more complex patterns than a single-layer RNN
- **Cons**: Much harder to train due to increased risk of vanishing/exploding gradients
- **Optimization**: Often requires Skip Connections (Residuals) between layers to help the gradient flow through the vertical stack