Neural nets unit:

- Introduction and principles (last week Wednesday)
- Neural net language model lab (last week Friday)
- RNNs and LTSMs (**Today**)

Rest of semester:

- Machine translation
- Large language models and text generation
- Ethical questions

Today:

- Context (review of neural net principles, comparison with earlier topics)
- Principles of recurrent neural networks (RNNs)
- Applications of RNNs (language models, POS-tagging, text classification)
- Long short-term memory (LSTM)
- Summary and look-ahead

Diagrams from Jurafsky and Martin, *Speech and Language Processing*, 3rd ed draft (Jan 7, 2023), Chapter 9

[The] temporal nature [of language] is reflected in some language processing algorithms. For example, the Viterbi algorithm we introduced for HMM part-of-speech tagging proceeds through the input a word at a time, carrying forward information gleaned along the way. Yet other machine-learning approaches, like those we've studied for sentiment analysis or other text classification tasks don't have this temporal nature—they assume simultaneous access to all aspects of their input.

The feed forward networks of Chapter 7 also assumed simultaneous access, although they also had a simple model for time. Recall that we applied feed-forward networks to language modeling by having them look at a fixed-size window of words, and then sliding this window over the input, making independent predictions along the way.

Jurafsky and Martin, Ch 9, pg 1

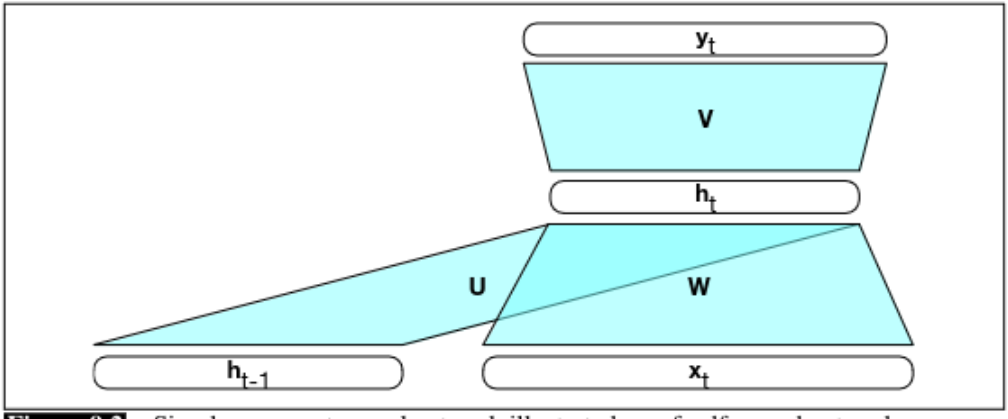

Figure 9.2 Simple recurrent neural network illustrated as a feedforward network.

Jurafsky and Martin, Sec 9.1, pg 3

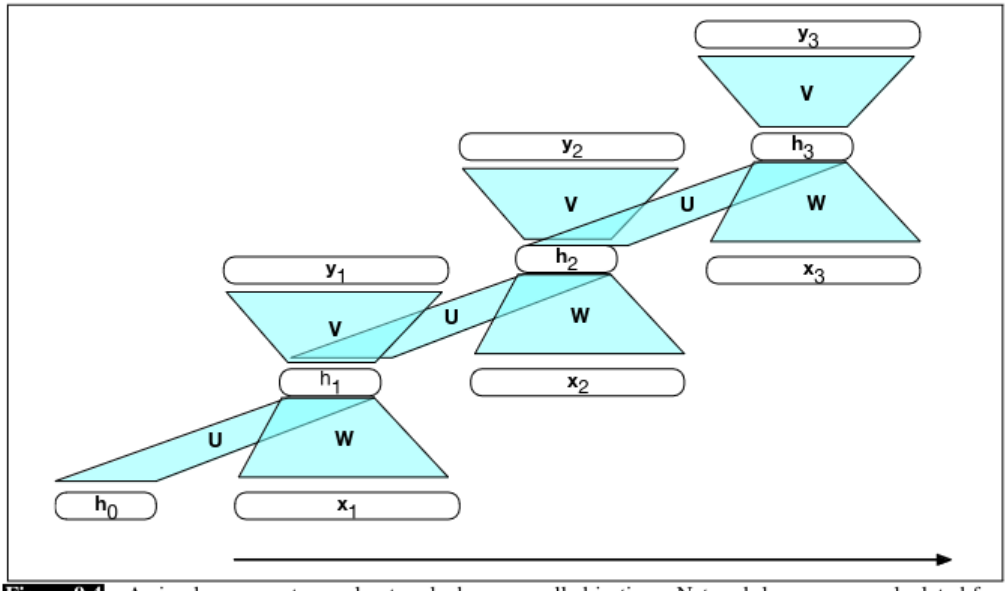

Figure 9.4 A simple recurrent neural network shown unrolled in time. Network layers are recalculated for each time step, while the weights **U**, **V** and **W** are shared across all time steps.

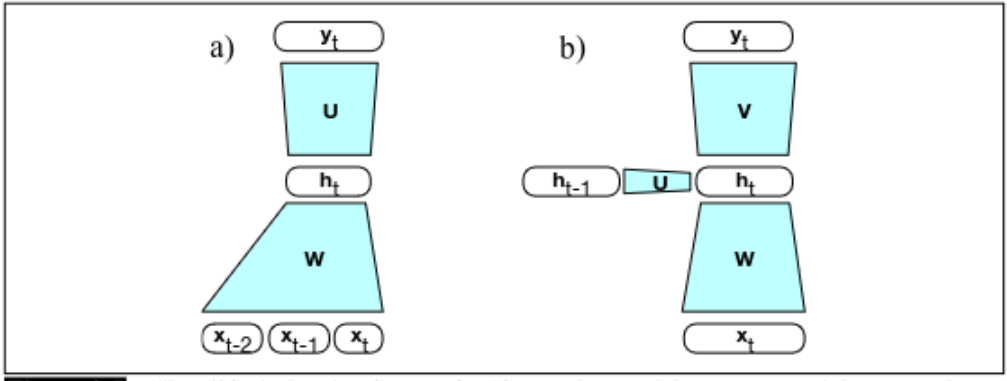

Figure 9.5 Simplified sketch of (a) a feedforward neural language model versus (b) an RNN language model moving through a text.

Jurafsky and Martin, Sec 9.2, pg 6

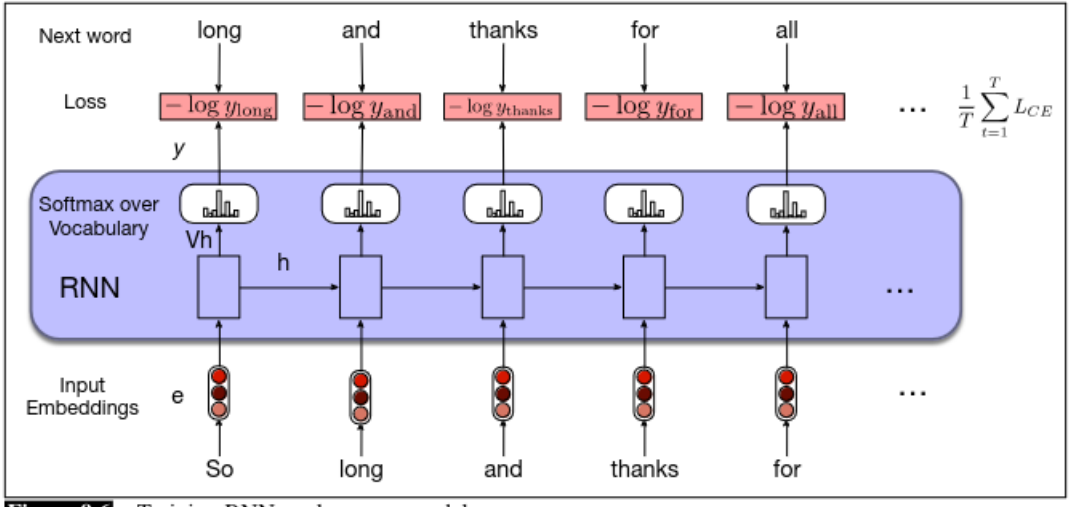

Figure 9.6 Training RNNs as language models.

Jurafsky and Martin, Sec 9.2, pg 7

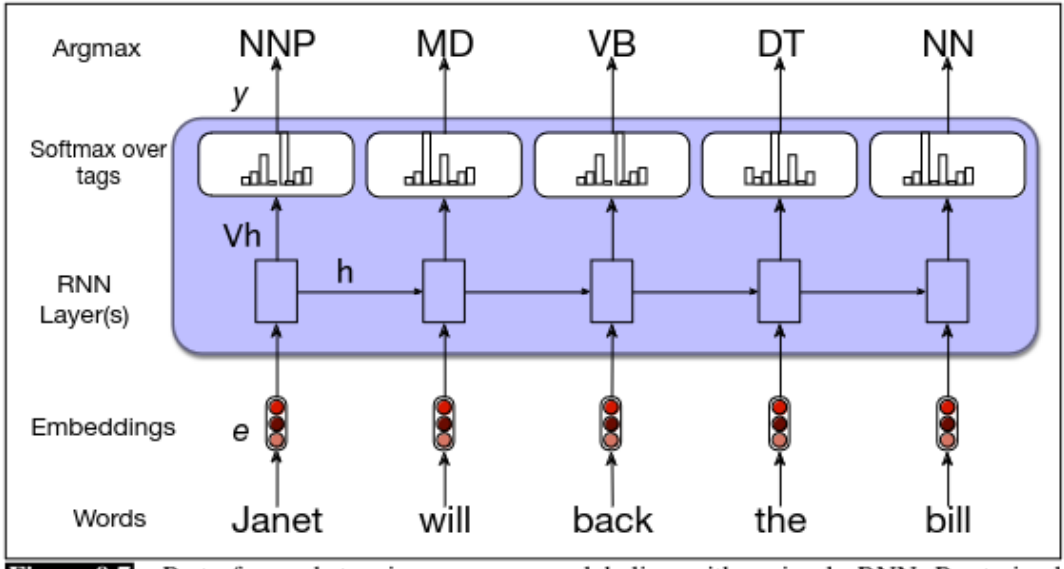

Figure 9.7 Part-of-speech tagging as sequence labeling with a simple RNN. Pre-trained word embeddings serve as inputs and a softmax layer provides a probability distribution over the part-of-speech tags as output at each time step.

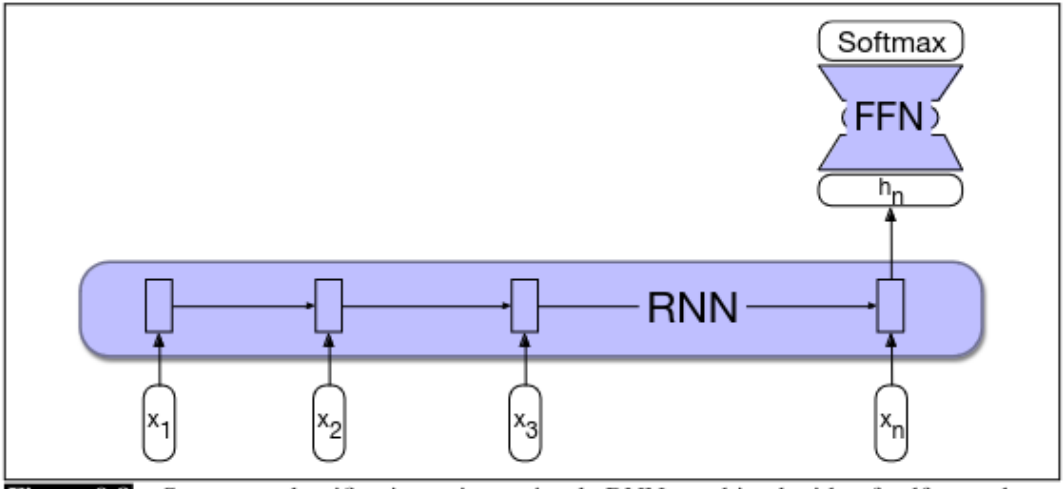

Figure 9.8 Sequence classification using a simple RNN combined with a feedforward network. The final hidden state from the RNN is used as the input to a feedforward network that performs the classification.

Jurafsky and Martin, Sec 9.3, pg 9

Autoregressive text-generation

```
def autoregress(clm, sample_len) :
    auto_reg_txt = ''
    vocab = clm.get_vocab()
    for i in range(sample_len) :
        # random number between 0 and 1
        nonce = random()
        running = 0.0
        j = 0
        while j < len(vocab) and running < nonce :
            c = vocab[j]
            running += clm.p_cond(c, auto_reg_txt)
            j += 1
        auto_reg_txt += c
    return auto_reg_txt
```

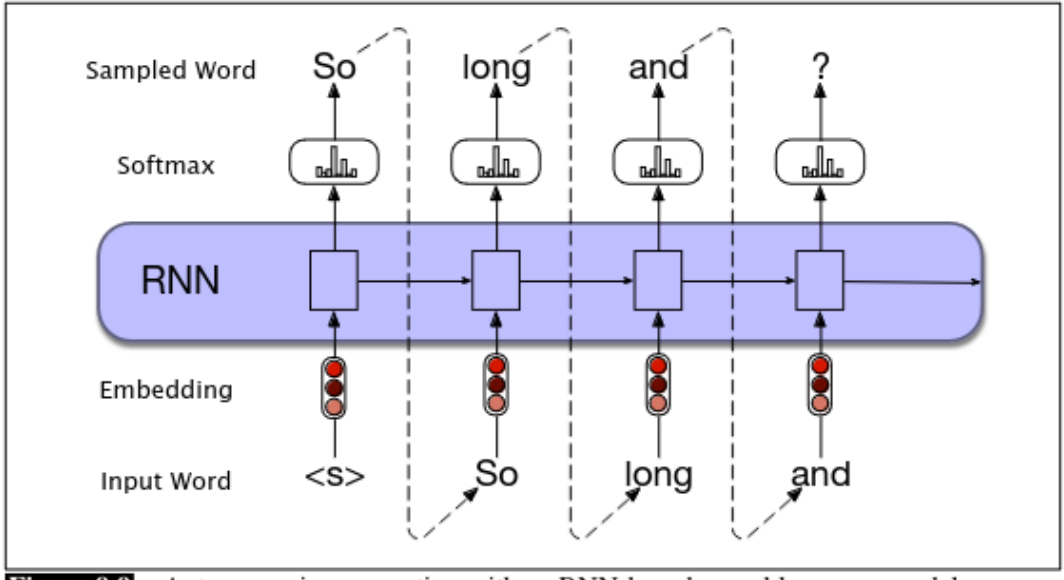

Figure 9.9 Autoregressive generation with an RNN-based neural language model.

Jurafsky and Martin, Sec 9.3, pg 11

In practice, it is quite difficult to train RNNs for tasks that require a network to make use of information distant from the current point of processing. Despite having access to the entire preceding sequence, the information encoded in hidden states tends to be fairly local, more relevant to the most recent parts of the input sequence and recent decisions. . .

One reason for the inability of RNNs to carry forward critical information is that the hidden layers, and, by extension, the weights that determine the values in the hid- den layer, are being asked to perform two tasks simultaneously: provide information useful for the current decision, and updating and carrying forward information required for future decisions. . . .

To address these issues, more complex network architectures have been designed to explicitly manage the task of maintaining relevant context over time, by enabling the network to learn to forget information that is no longer needed and to remember information required for decisions still to come.

The most commonly used such extension to RNNs is the long short-term mem- ory (LSTM) network. LSTMs divide the context management problem into two subproblems: removing information no longer needed from the context, and adding information likely to be needed for later de- cision making.

Jurafsky and Martin, Sec 9.5, pg 14

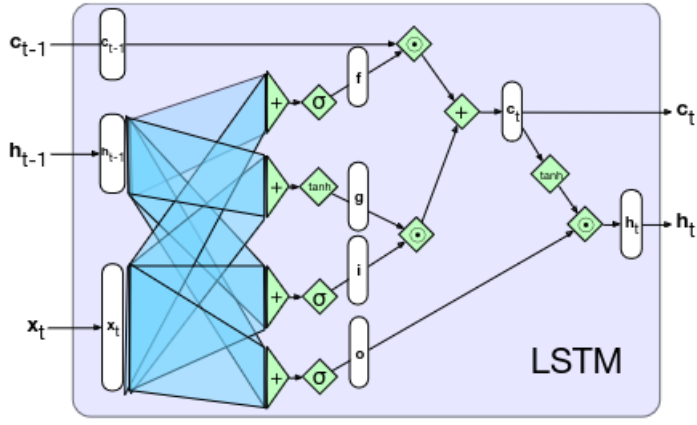

Figure 9.13 A single LSTM unit displayed as a computation graph. The inputs to each unit consists of the current input, x, the previous hidden state, h_{t-1}, and the previous context, c_{t-1}. The outputs are a new hidden state, h_t and an updated context, c_t.

Jurafsky and Martin, Sec 9.5, pg 16

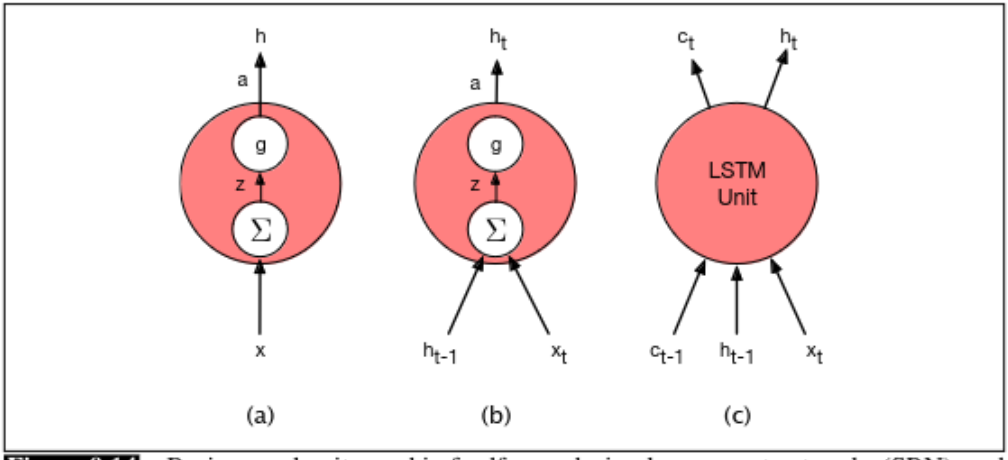

Figure 9.14 Basic neural units used in feedforward, simple recurrent networks (SRN), and long short-term memory (LSTM).

Jurafsky and Martin, Sec 9.5, pg 16

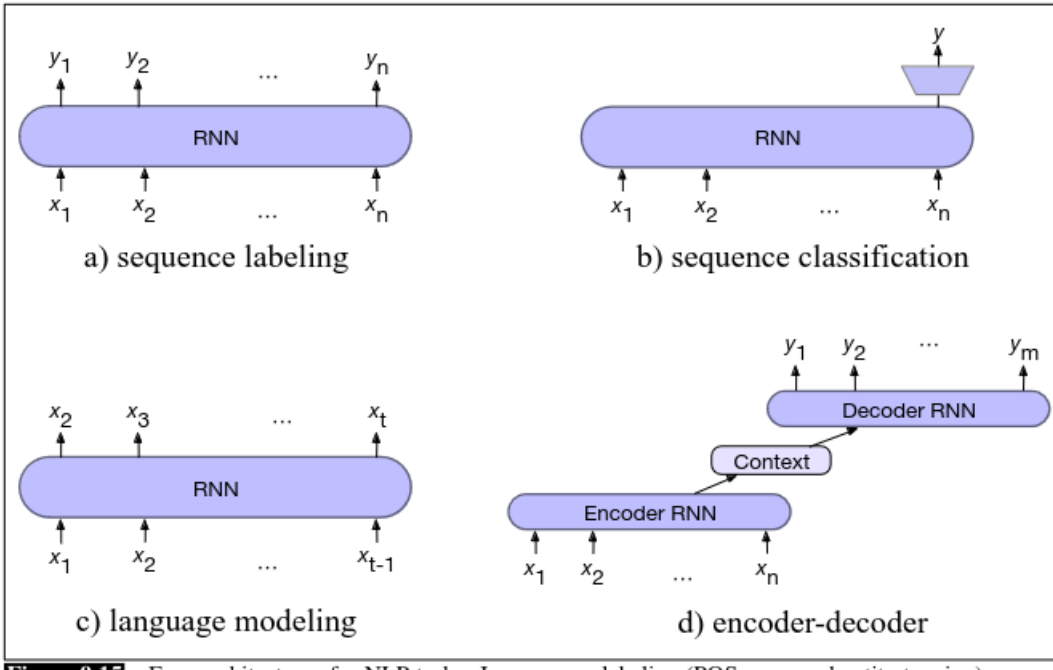

Figure 9.15 Four architectures for NLP tasks. In sequence labeling (POS or named entity tagging) we map

Coming up:

- Read J&M chapter 9 (Mon, Nov 20)
- Take neural nets/RNNs quiz (Tues, Nov 21)
- Read J&M chapter 13 (Mon, Nov 27)
- Finish stylometry assignment (Fri, Dec 1)

Remaining things for this semester (not yet posted to Canvas, due dates not set)

- Word2Vec programming assignment
- A reading on large language models and/or text generation (not from J&M)
- Three or fewer more quizzes
- Two short reflections on ethical or social questions