LING530F: Deep Learning for Natural Language Processing (DL-NLP)

Muhammad Abdul-Mageed

muhammad.mageed@ubc.ca

Natural Language Processing Lab

The University of British Columbia

Table of Contents

RNN With a Single Output in the End

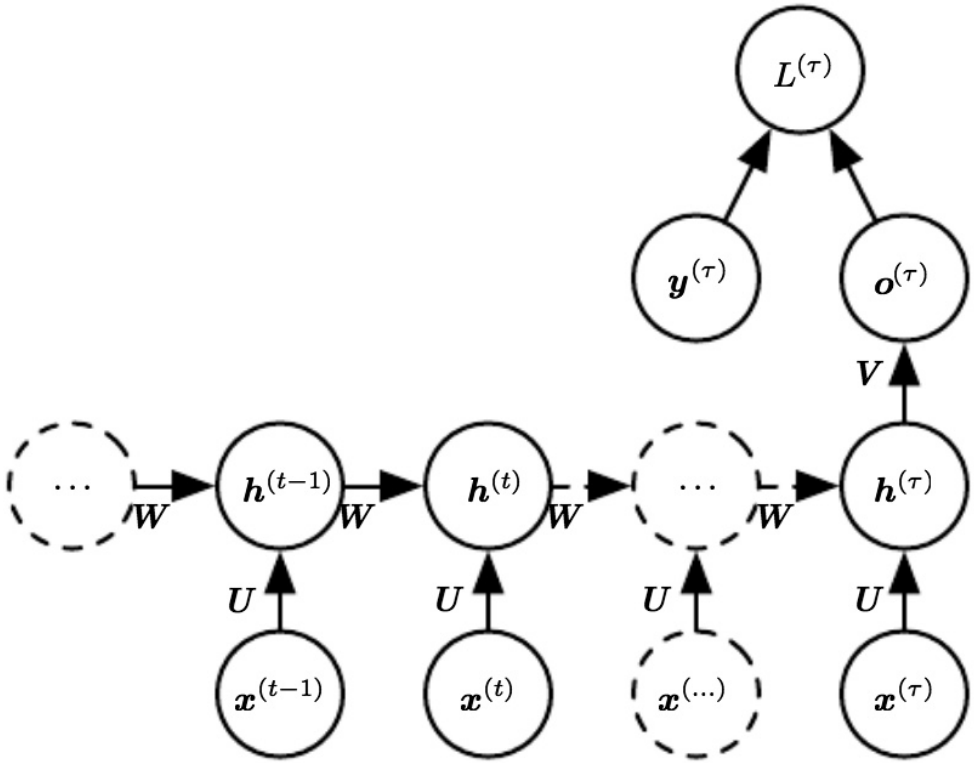

Figure: An RNN with a single output in the end. This network can be used, for example, for speaker, dialect, gender, or sentiment identification when cast as text classification

RNN With Output Recurrence

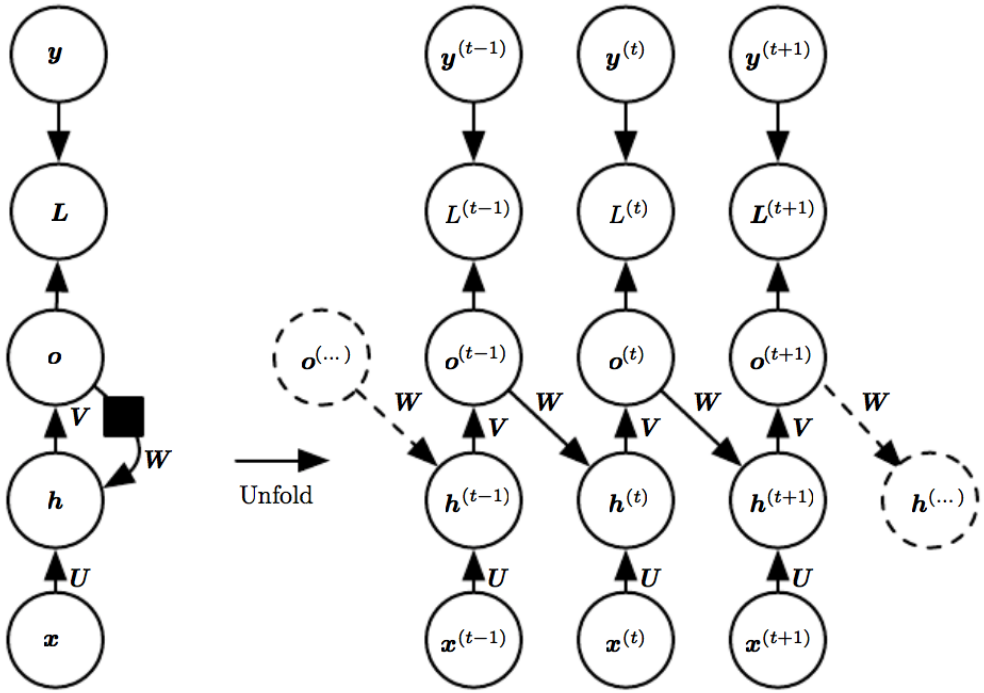

Figure: This network is less powerful than one with full recurrence, yet can be parallelized, with the gradient for each step t computed in isolation. There is no need to compute the output for the previous time step first, because the **training set provides the ideal value of that output** $\rightarrow$ teacher forcing. [From Goodfellow et al., 2016]

Teacher Forcing

Teacher Forcing

- Models with recurrent connections from their output back to the model hidden state can be trained with **teacher forcing**
- Gold labels are fed from the training data
- **Consider example of POS tagging** where gold labels at each time step are used, rather than using model predictions of previous words
- Although teacher forcing can help us avoid full recurrence as with BPTT, teacher forcing can also be combined with BPTT (where the hidden states are a function of previous time steps)

Closed Loop

Network in Closed Loop

- In **closed loop** mode, the network outputs are fed back as inputs.
- This could be problematic if the labels/fed-back inputs at this test time are quite different freom those during training.
- **Solution I**: Mix teacher forcing with free-running inputs, by starting with predicting the gold target a number of steps in the future then using the network's own predictions after a few steps
- **Solution II**: Use a mixture of generated and actual values as inputs
- This exploits a **curriculum learning** strategy to gradually use more of the generated values as input

Drawing Samples from RNN

Deciding End of Sequence

- The network needs a mechanism to know **where to stop** (i.e., determine the length of the sequence.
- **Approach 1: Add an extra** $< end >$ **symbol to the vocabulary**, where generation stops after the symbol is produced.
- **Approach 2: Add an extra Bernoulli output**, usually a sigmoid unit trained with the cross-entropy loss to maximize the log-probability of the correct prediction as to whether the sequence ends or continues at each time step.
- **Approach 2** is **more general** than **approach 1**, as it can be added to any RNN (rather than an RNN that generates a sequence).
- See Goodfellow et al., 2016 (p. 384) for a third approach based on simply augmenting the network with a counter at each time step.

From Fixed-Length x to Sequences of Y

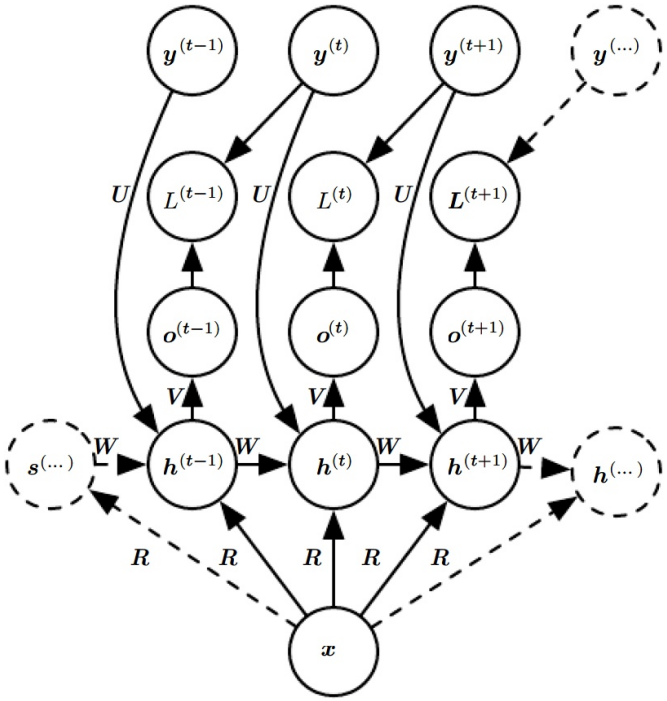

Figure 10.9: An RNN that maps a fixed-length vector $\boldsymbol{x}$ into a distribution over sequences $\mathbf{Y}$. This RNN is appropriate for tasks such as image captioning, where a single image is used as input to a model that then produces a sequence of words describing the image. Each element $\boldsymbol{y}^{(t)}$ of the observed output sequence serves both as input (for the current time step) and, during training, as target (for the previous time step).

Figure: A weight matrix $\mathbf{R}$ that was absent from the model with only a sequence of y values is introduced. The product $\mathbf{x}^T\mathbf{R}$ is added as an additional input to hidden units at each time step.

Bidirectional RNN

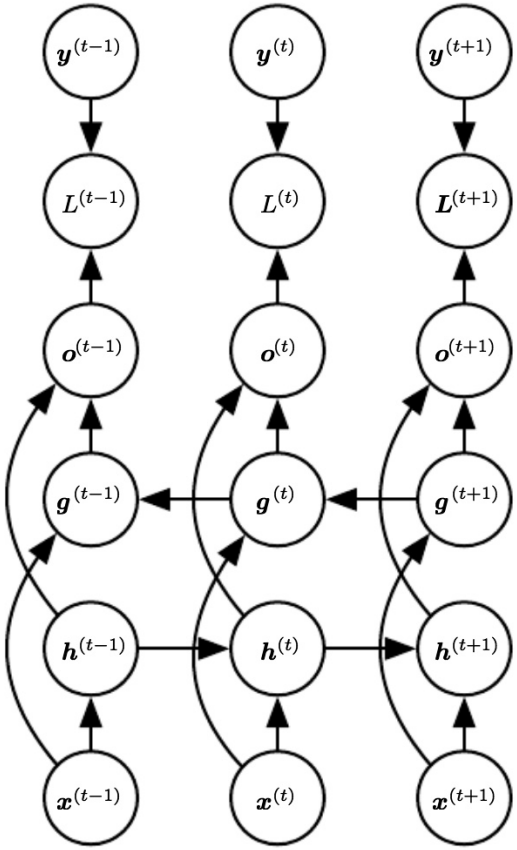

Figure: A BiRNN can be used for tasks like speech recognition, POS tagging, and handwriting recognition. Additionally, can be used for many **sequence-to-sequence** tasks (see next slide . . .)

Encoder-Decoder Sequence-to-Sequence Architectures

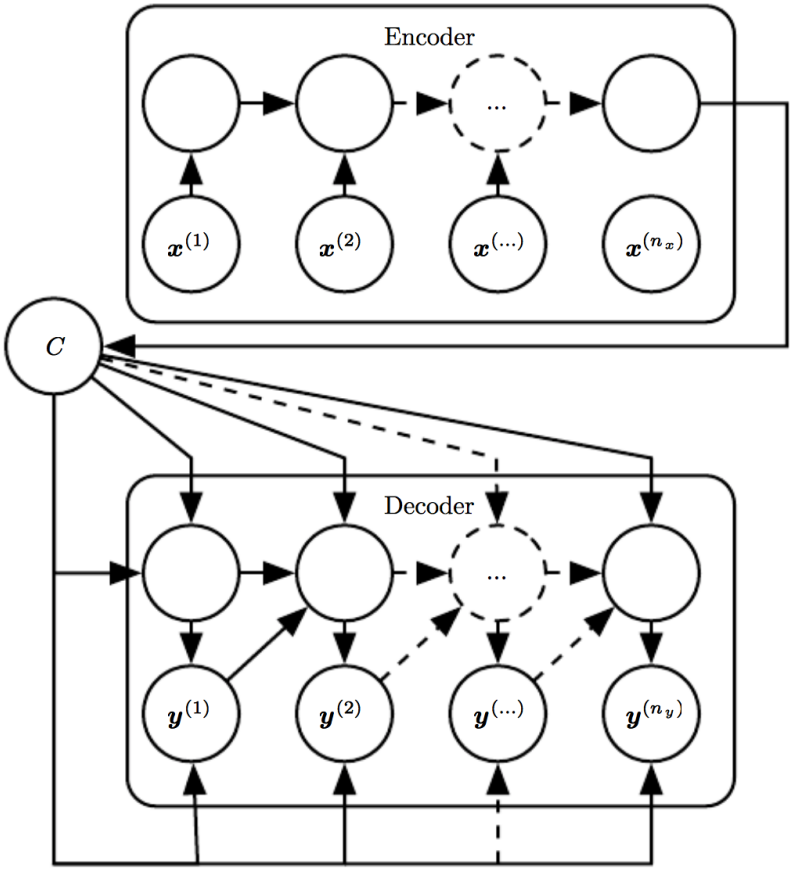

Figure: An seq2seq RNN. The final hidden state of the encoder RNN is used to compute a generally fixed-size a semantic summary C of the input sequence and is given as input to the encoder RNN. [From Goodfellow et al., 2016)

Encoder-Decoder Design

Design Details

- C might be a vector or sequence of vectors that summarize the input sequence.
- The **encoder** processes the input sequence $(\mathbf{x}^{(1)}, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n_x)})$, emitting a fixed-length context C as a simple function of its last hidden state.
- A **decoder** network is conditioned on C to generate the output sequence $(\mathbf{y}^{(1)}, \mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(n_y)})$.
- The two RNNs are trained jointly to maximize the average of $\log (\mathbf{y}^{(1)}, \mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(n_y)}) \mid (\mathbf{x}^{(1)}, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n_x)})$.
- **Length of the encoder and decoder can vary** from each other, unlike work done before Cho et al. (2014) and Sutskever et al. (2014) introduced these architectures.

Connecting Encoder to Decoder

How Writer & Reader Are Connected

- If the context C is a vector, then the **decoder RNN is a vector-to-sequence RNN**.
- At least **2 ways for a vector-to-sequence RNN to receive input**:
  - **Input** can be provided as the initial state of the RNN
  - **Input** can be connected to the hidden units at each time step

Limitation

- The context C output by encoder can have **too small dimension to summarize a long sequence**
- Bahdanau et al. (2015) proposed to make a **variable length sequence** (in MT work), and introduced an **attention mechanism**.