Chapter 9 Sequence Processing with Recurrent Networks

When we comprehend and produce spoken language, we are processing continuous input streams of indefifinite length. And even when dealing with written text we normally process it sequentially, even though we in principle have arbitrary access to all the elements at once.

In contrast, the machine learning approaches we've studied for sentiment analysis and other classifification tasks do not have this temporal nature.

- **recurrent neural networks**：a class of networks designed to address these challenges by dealing directly with the temporal aspect of language, allowing us to handle variable length inputs without the use of arbitrary fifixed-sized windows, and providing the means to capture and exploit the temporal nature of language.

Simple Recurrent Neural Networks

- **Elman Networks/simple recurrent networks**：As with ordinary feedforward networks, an input vector representing the current input element

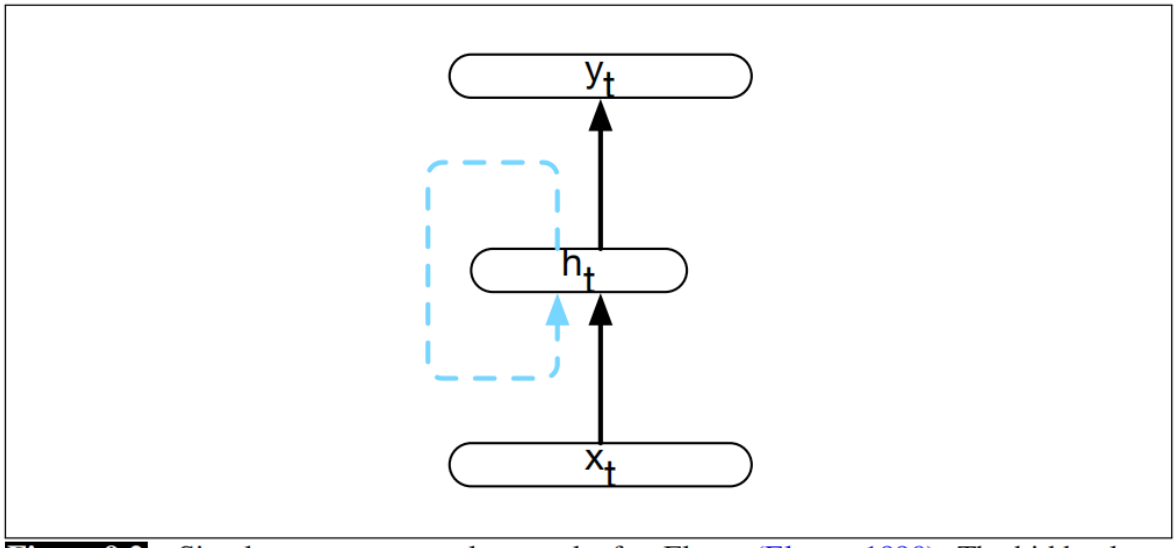

Figure 9.2 Simple recurrent neural network after Elman (Elman, 1990). The hidden layer includes a recurrent connection as part of its input. That is, the activation value of the hidden layer depends on the current input as well as the activation value of the hidden layer from the previous time step.

The hidden layer from the previous time step provides a form of memory, or context, that encodes earlier processing and informs the decisions to be made at later points in time.

Inference in Simple RNNs

Forward inference (mapping a sequence of inputs to a sequence of outputs) in an RNN is nearly identical to what we've already seen with feedforward networks.

function FORWARDRNN(x, *network*) **returns** output sequence y

$h_0 \leftarrow 0$
for $i \leftarrow 1$ **to** LENGTH(x) **do**
  $h_i \leftarrow g(U\, h_{i-1} + W\, x_i)$
  $y_i \leftarrow f(V\ \ h_i)$
return y

Figure 9.4 Forward inference in a simple recurrent network. The matrices U, V and W are shared across time, while new values for h and y are calculated with each time step.

Training

As with feedforward networks, we'll use a training set, a loss function, and backpropagation to obtain the gradients needed to adjust the weights in these recurrent networks.

- **Backpropagation Through Time**: In the fifirst pass, we perform forward inference, computing h_t, y_t, and accumulating the loss at each step in time, saving the value of the hidden layer at each step for use at the next time step. In the second phase, we process the sequence in reverse, computing the required error terms gradients as we go, computing and saving the error term for use in the hidden layer for each step backward in time.

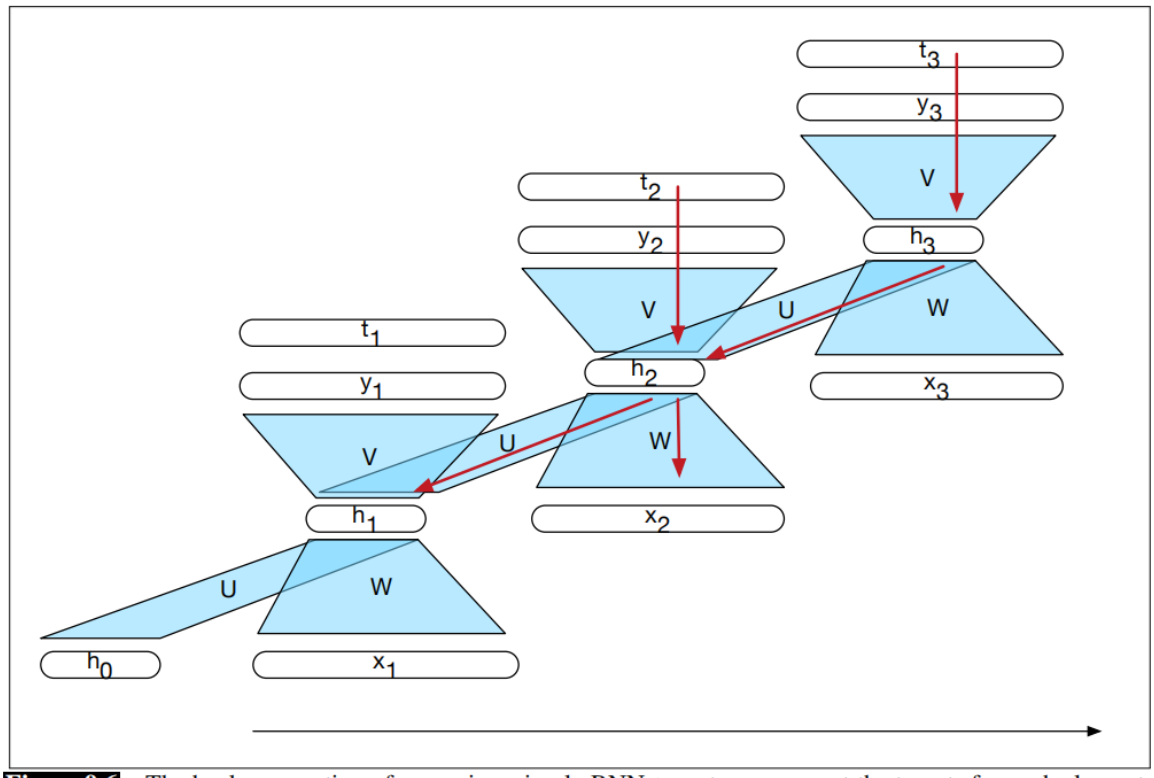

Figure 9.6 The backpropagation of errors in a simple RNN t_i vectors represent the targets for each element of the sequence from the training data. The red arrows illustrate the flow of backpropagated errors required to calculate the gradients for U, V and W at time 2. The two incoming arrows converging on h_2 signal that these errors need to be summed.

Unrolled Networks as Computation Graphs

In such an approach, we provide a template that speci-fifies the basic structure of the network, including all the necessary parameters for the input, output, and hidden layers, the weight matrices, as well as the activation and output functions to be used.

Applications of Recurrent Neural Networks

Recurrent neural networks have proven to be an effective approach to language modeling, sequence labeling tasks such as part-of-speech tagging, as well as sequence classifification tasks such as sentiment analysis and topic classifification.

Recurrent Neural Language Models

Given a fifixed preceding context, both attempt to predict the next word in a sequence.

- **autoregressive generation**: the word generated at the each time step is conditioned on the word generated by the network at the previous step.
  - To begin, sample the fifirst word in the output from the softmax distribution that
    results from using the beginning of sentence marker, ~~, as the fifirst input.~~
  - ~~Use the word embedding for that fifirst word as the input to the network at the next time step, and then sample the next word in the same fashion.~~
  - ~~Continue generating until the end of sentence marker,~~, is sampled or a fifixed length limit is reached.

Sequence Labeling

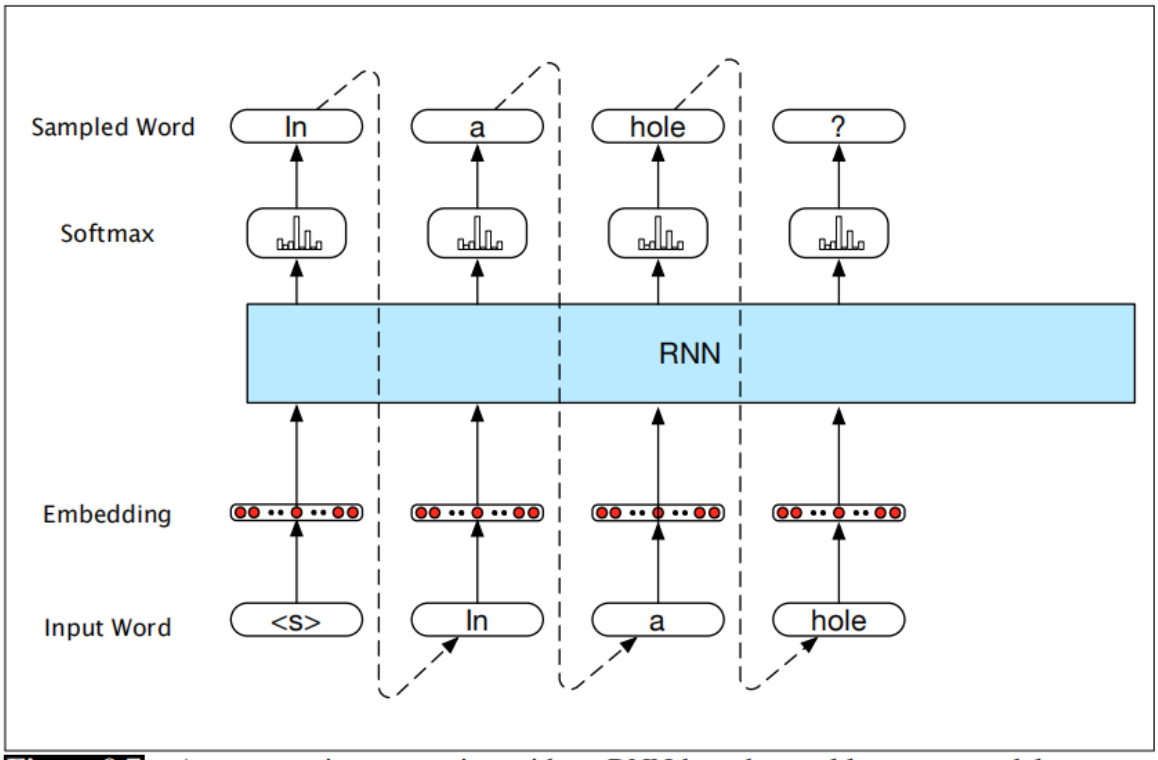

Figure 9.7 Autoregressive generation with an RNN-based neural language model.

To use sequence labeling for a span-recognition problem, we'll use a technique called **IOB encoding** (Ramshaw and Marcus, 1995).

Yet another application of sequence labeling is to the problem of **structure prediction**. Here the task is to take an input sequence and produce some kind of structured output, such as a parse tree or meaning representation.

Viterbi and Conditional Random Fields (CRFs)

We can then use a variant of the Viterbi algorithm to select the most likely tag sequence.

RNNs for Sequence Classifification

To apply RNNs in this setting, the text to be classifified is passed through the RNN a word at a time generating a new hidden layer at each time step.

And the training regimen that uses the loss from a downstream application to adjust the weights all the way through the network is referred to as **end-to-end training**.