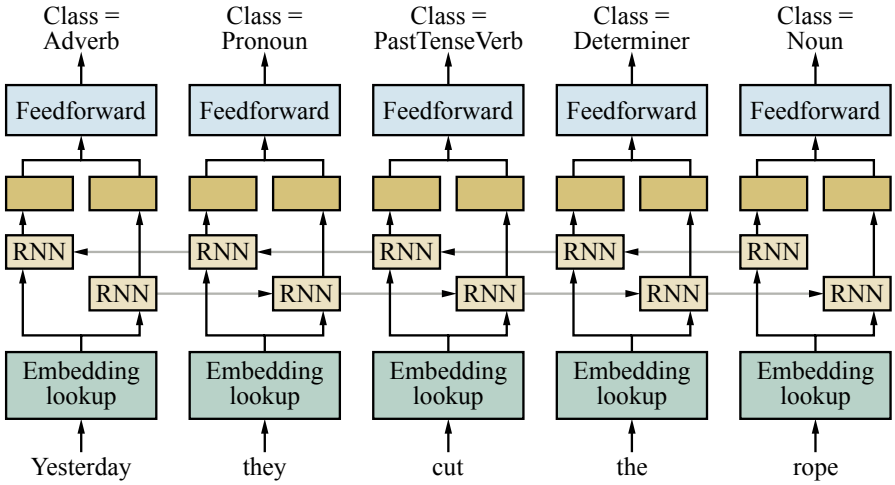

Figure 24.5 A bidirectional RNN network for POS tagging.

words offset by 1. That is, for the training text "hello world," the first input $\mathbf{x}_1$ is the word embedding for "hello" and the first output $\mathbf{y}_1$ is the word embedding for "world." We are training the model to predict the next word, and expecting that in order to do so it will use the hidden layer to represent useful information. As explained in Section 21.6.1 we compute the difference between the observed output and the actual output computed by the network, and back-propagate through time, taking care to keep the weights the same for all time steps.

Once the model has been trained, we can use it to generate random text. We give the model an initial input word $\mathbf{x}_1$, from which it will produce an output $\mathbf{y}_1$ which is a softmax probability distribution over words. We sample a single word from the distribution, record the word as the output for time t, and feed it back in as the next input word $\mathbf{x}_2$. We repeat for as long as desired. In sampling from $\mathbf{y}_1$ we have a choice: we could always take the most likely word; we could sample according to the probability of each word; or we could oversample the less-likely words, in order to inject more variety into the generated output. The sampling weight is a hyperparameter of the model.

Here is an example of random text generated by an RNN model trained on Shakespeare's works (Karpathy, 2015):

> *Marry, and will, my lord, to weep in such a one were prettiest;*
> *Yet now I was adopted heir*
> *Of the world's lamentable day,*
> *To watch the next way with his father with his face?*

24.2.2 Classification with recurrent neural networks

It is also possible to use RNNs for other language tasks, such as part of speech tagging or coreference resolution. In both cases the input and hidden layers will be the same, but for a POS tagger the output will be a softmax distribution over POS tags, and for coreference resolution it will be a softmax distribution over the possible antecedents. For example, when the network gets to the input **him** in "*Eduardo told me that Miguel was very sick so I took* **him** *to the hospital*" it should output a high probability for "*Miguel*."