words, where, arguably, it is the least useful. This disadvantage has stimulated the exploration of alternative methods to deal with high-dimensional outputs, described below.

12.4.3.2 Hierarchical Softmax

A classical approach (Goodman, 2001) to reducing the computational burden of high-dimensional output layers over large vocabulary sets $\mathbb{V}$ is to decompose probabilities hierarchically. Instead of necessitating a number of computations proportional to $|\mathbb{V}|$ (and also proportional to the number of hidden units, n_h), the $|\mathbb{V}|$ factor can be reduced to as low as $\log |\mathbb{V}|$. Bengio (2002) and Morin and Bengio (2005) introduced this factorized approach to the context of neural language models.

One can think of this hierarchy as building categories of words, then categories of categories of words, then categories of categories of categories of words, etc. These nested categories form a tree, with words at the leaves. In a balanced tree, the tree has depth $O(\log |\mathbb{V}|)$. The probability of a choosing a word is given by the product of the probabilities of choosing the branch leading to that word at every node on a path from the root of the tree to the leaf containing the word. Figure 12.4 illustrates a simple example. Mnih and Hinton (2009) also describe how to use multiple paths to identify a single word in order to better model words that have multiple meanings. Computing the probability of a word then involves summation over all of the paths that lead to that word.

To predict the conditional probabilities required at each node of the tree, we typically use a logistic regression model at each node of the tree, and provide the same context C as input to all of these models. Because the correct output is encoded in the training set, we can use supervised learning to train the logistic regression models. This is typically done using a standard cross-entropy loss, corresponding to maximizing the log-likelihood of the correct sequence of decisions.

Because the output log-likelihood can be computed efficiently (as low as $\log |\mathbb{V}|$ rather than $|\mathbb{V}|$), its gradients may also be computed efficiently. This includes not only the gradient with respect to the output parameters but also the gradients with respect to the hidden layer activations.

It is possible but usually not practical to optimize the tree structure to minimize the expected number of computations. Tools from information theory specify how to choose the optimal binary code given the relative frequencies of the words. To do so, we could structure the tree so that the number of bits associated with a word is approximately equal to the logarithm of the frequency of that word. However, in