practice, the computational savings are typically not worth the effort because the computation of the output probabilities is only one part of the total computation in the neural language model. For example, suppose there are l fully connected hidden layers of width n_h. Let n_b be the weighted average of the number of bits required to identify a word, with the weighting given by the frequency of these words. In this example, the number of operations needed to compute the hidden activations grows as as $O(ln_h^2)$ while the output computations grow as $O(n_h n_b)$. As long as $n_b \leq ln_h$, we can reduce computation more by shrinking n_h than by shrinking n_b. Indeed, n_b is often small. Because the size of the vocabulary rarely exceeds a million words and $\log_2(10^6) \approx 20$, it is possible to reduce n_b to about 20, but n_h is often much larger, around 10^3 or more. Rather than carefully optimizing a tree with a branching factor of 2, one can instead define a tree with depth two and a branching factor of $\sqrt{|\mathbb{V}|}$. Such a tree corresponds to simply defining a set of mutually exclusive word classes. The simple approach based on a tree of depth two captures most of the computational benefit of the hierarchical strategy.

One question that remains somewhat open is how to best define these word classes, or how to define the word hierarchy in general. Early work used existing hierarchies (Morin and Bengio, 2005) but the hierarchy can also be learned, ideally jointly with the neural language model. Learning the hierarchy is difficult. An exact optimization of the log-likelihood appears intractable because the choice of a word hierarchy is a discrete one, not amenable to gradient-based optimization. However, one could use discrete optimization to approximately optimize the partition of words into word classes.

An important advantage of the hierarchical softmax is that it brings computational benefits both at training time and at test time, if at test time we want to compute the probability of specific words.

Of course, computing the probability of all $|\mathbb{V}|$ words will remain expensive even with the hierarchical softmax. Another important operation is selecting the most likely word in a given context. Unfortunately the tree structure does not provide an efficient and exact solution to this problem.

A disadvantage is that in practice the hierarchical softmax tends to give worse test results than sampling-based methods we will describe next. This may be due to a poor choice of word classes.

12.4.3.3 Importance Sampling

One way to speed up the training of neural language models is to avoid explicitly computing the contribution of the gradient from all of the words that do not appear