	finetune	test perplexity
(Merity et al., 2017)	no	58.8
baseline	no	58.8 ± 0.3
sparse LSTM	no	$\mathbf{57.9 \pm 0.3}$
(Merity et al., 2017)	yes	57.3
baseline	yes	$\mathbf{56.6 \pm 0.2}$
sparse LSTM	yes	57.0 ± 0.2

Table 1: Language modeling for PTB (mean $\pm$ stdev).

$i_s = h_s = 1725$). *Dense* and *sparse* variants have the same number of parameters for $N = 3$ and $\gamma = 0.555$. These values are obtained by identifying both expressions. Note that the equality in model parameters for the dense and sparse case holds only approximately due to rounding errors in (γi_s) and (h_s/N).

Figure 1 displays $\mathbf{W}_{hh}$ and $\mathbf{W}_{hi}$ for the middle layer, which has close to 11M parameters out of the total of 24M in the whole model. A dense model with hidden size $h = 1725$ would require 46M parameters, with 24M in the middle LSTM alone.

Given the strong hyperparameter dependence of the AWD-LSTM model, and the known issues in objectively evaluating language models (Melis et al., 2017), we decided to keep all hyperparameters (i.e., dropout rates and optimization scheme) as in the implementation from Merity et al. (2017)[5], including the weight dropping with $p = 0.5$ in the sparse $\mathbf{W}_{hh}$ matrices. Table 1 shows the test perplexity on a processed version (Mikolov et al., 2010) of the Penn Treebank (PTB) (Marcus et al., 1993), both with and without the 'finetune' step[6], displaying mean and standard deviation over 5 different runs. Without finetuning, the sparse model consistently performs around 1 perplexity point better, whereas after finetuning, the original remains slightly better, although less consistently so over different random seeds. We observed that the sparse model overfits more strongly than the baseline, especially during the finetune step. We hypothesize that the regularization effect of *a priori* limiting interactions between dimensions does not compensate for the increased expressiveness of the model due to the larger hidden state size. Further experimentation, with tuned hyperparameters, is needed to determine the actual benefits of predefined sparseness, in terms of model size, resulting perplexity, and sensitivity to the choice of hyperparameters.

[5]Our implementation extends https://github.com/salesforce/awd-lstm-lm.

[6]The 'finetune' step indicates hot-starting the Averaged Stochastic Gradient Descent optimization once more, after convergence in the initial optimization step (Merity et al., 2017).

4 Sparse Word Embeddings

Given a vocabulary with V words, we want to construct vector representations of length k for each word such that the total number of parameters needed (i.e., non-zero entries), is smaller than $k\,V$. We introduce one way to do this based on word frequencies (Section), and present part-of-speech tagging experiments (Section).

4.1 Word-Frequency based Embedding Size

Predefined sparseness in word embeddings amounts to deciding which positions in the word embedding matrix $\mathbf{E} \in \mathbb{R}^{V \times k}$ should be fixed to zero, prior to training. We define the fraction of trainable entries in $\mathbf{E}$ as the embedding density δ_E. We hypothesize that rare words can be represented with fewer parameters than frequent words, since they only appear in very specific contexts. This will be investigated experimentally in Section . Word occurrence frequencies have a typical Zipfian nature (Manning et al., 2008), with many rare and few highly frequent terms. Thus, representing the long tail of rare terms with short embeddings should greatly reduce memory requirements.

In the case of a low desired embedding density δ_E, we want to save on the rare words, in terms of assigning trainable parameters, and focus on the fewer more popular words. An exponential decay in the number of words that are assigned longer representations is one possible way to implement this. In other words, we propose to have the number of words that receive a trainable parameter at dimension j decrease with a factor α^j ($\alpha \in]0, 1]$). For a given fraction δ_E, the parameter α can be determined from requiring the total number of non-zero embedding parameters to amount to a given fraction δ_E of all parameters:

$$\text{\# embedding params.} = \sum_{j=0}^{k-1} \alpha^j V = \delta_E \, k \, V$$

and numerically solving for α.

Figure 2 gives examples of embedding matrices with varying δ_E. For a vocabulary of 44k terms