1 Appendix

1.1 NNLM Training

We use the Penn Treebank corpus (Marcus et al., 1993) to train a neural language model using these embedding models. The data is split into train, validation and test sets, with a vocabulary of 10000. We process the text to only include words that occur within the lexicon of our pretrained embedding models (≈ 7300 words), replacing the other words with the $\langle$PAD$\rangle$ token. The model architecture and training closely follow that of the medium-sized LSTM model presented by Zaremba et al. (2014) which we build using *Tensorflow*. The network consists of two LSTM layers with hidden states of size 650, with a fully-connected softmax output and word embedding look-up table input. The models are trained using stochastic gradient descent with a learning rate of 1.0 for 6 epochs, with the learning rate decreasing at a rate of 1.2 for up to 39 epochs of training. The LSTMs are initialized with a uniform random distribution between $[-0.05, 0.05]$, and are unrolled for 35 time steps, and train on contiguous sequences of text, passing the hidden state to the proceeding batch with the batch size set to 20. A dropout rate of 0.5 was applied to the embedding vector, and outputs of both LSTM layers during training. As proposed previously by Gulordava et al. (2018), we add a linear projection layer between the final LSTM layer and the softmax output, which increases the dimensionality to the appropriate length.

1.2 Bias Analysis

In this paper, we omitted the bias terms from the output embedding layer as in previous work, which occurs in many cases when training NNLMs with tied embeddings. Since we are using the final layer of the network to find some locally-optimal vectors based on the output embeddings, we include the bias terms when optimising for our AM vectors, as they convey information about some prior knowledge concerning the input. In the case of NNLMs, it would make sense for these bias terms to capture frequency patterns in the distribution of words. It is well known that many types of semantic models learn word co-occurrence information and are heavily affected by word frequency (Wartena, 2014; Wendlandt et al., 2018).

To test our hypothesis, we measure the Pearson correlation between the bias terms and the natural log counts of the 20000 highest occurring words

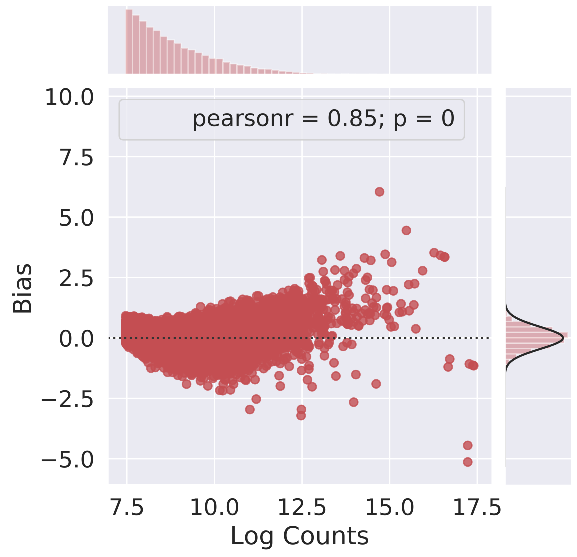

Figure 1: Plot of bias terms against log of the counts for each word in the vocabulary (the first 20000 words).

in the one billion word news dataset (Chelba et al., 2013). We present our results in figure 1. There is a strong linear correlation between the bias terms and the log counts with most bias terms close to zero. While these results make a strong case for the relationship between word frequency and bias term, it doesn't fully explain all the information they capture during training. Generally, computing the bias term isn't as computationally intensive as the actual embeddings themself, though many leave this bias vector out of their networks classification layer when weight tying. Furthermore, while it probably doesn't greatly affect the model, the weight update to the input embeddings doesn't apply to the bias term during training. This may cause a shift when computing the class posterior using the output embedding and bias. Precomputing the bias terms using a naive frequency-based approach with a scaling factor could save time and improve results, whilst being decoupled from the weight updates. We leave this for future work, though we would not expect a large change in performance between trained bias and precomputed bias.

1.3 Bias Information in AM Embeddings

We also performed additional analyses to understand why the AM embeddings outperform the output embeddings of the *J-LM* model on so many of these tasks. To test whether these locally-optimal representations encode the bias information during the training procedure, we use a linear regression to predict the value of the bias term using the em-

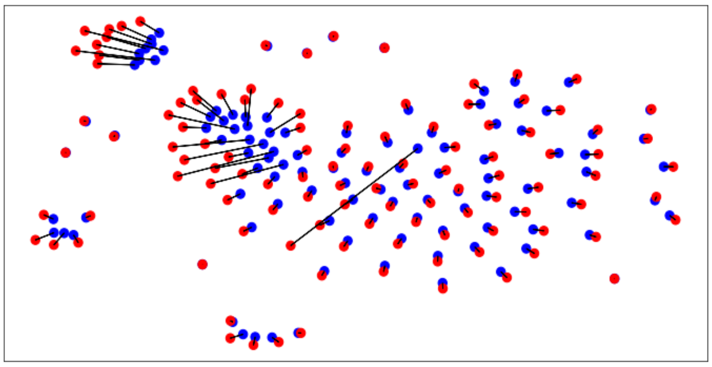

Figure 2: Two-dimensional TSNE scatter plot with arrows between output embeddings (blue) and AM embeddings (red).

Vocab Splits	Train (90%)	Test (10%)
Output Embeddings	0.290	0.301
AM Embeddings	0.257	0.195
Concatenated	0.218	0.182

Table 1: Mean squared error scores of linear regression models trained to predict bias term for each embedding model.

beddings as input. We split the 20000 words into a training and testing set, which contain 90% of the words and 10% of the words, respectively. We train three regression models, one using the output embeddings, one using the AM embeddings, and one which concatenates both embeddings together. If the bias is captured by the AM embeddings, then we would expect them to be much better at predicting the bias than the original weight matrix, which does not include the bias term. The results are presented in Table 1. As we can see, the results strongly indicate that these AM representations are better at predicting the bias. This suggests that the AM embeddings perturb the original embeddings in some way that includes information from the bias term, which we demonstrated in the previous section relates to word frequency information.

1.4 Visualising the Distribution

For visual analysis, we use TSNE to reduce the dimensionality of both the output embeddings and the AM embeddings down from 1024 to just 2 dimensions. Here, TSNE is preferred as it preserves the information about the geometric distance between vectors. To aid interpretation of the visualisation, we selected words that represent a set of distinct yet semantically similar concepts. For our analysis, we chose four-digit numbers greater than or equal to 1900, since these should correspond quite often to years and thus have a strong semantic association. The results are displayed in Figure 2, where arrows connect the output embeddings in blue, and the AM embeddings in red. We see that many of the points become further spread out as a consequence of training objective, but overall the clusters tend to stay close together. Finding the optimal class implies that the network must also minimise the probability of incorrect classes, which requires the vector representations to be more linearly separable. While our analysis provides ample evidence as to why these AM embeddings perform well on downstream prediction tasks and language modelling, it doesn't fully explain the performance on intrinsic evaluation benchmarks.

References

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Kristina Gulordava, Laura Aina, and Gemma Boleda. 2018. How to represent a word and predict it, too: Improving tied architectures for language modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2936–2941, Brussels, Belgium. Association for Computational Linguistics.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank.

Christian Wartena. 2014. On the effect of word frequency on distributional similarity.

Laura Wendlandt, Jonathan K Kummerfeld, and Rada Mihalcea. 2018. Factors influencing the surprising instability of word embeddings. *arXiv preprint arXiv:1804.09692*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.