Measuring Feature Sparsity in Language Models

Mingyang Deng[*]
MIT

Lucas Tao[*]
Stanford University

Joe Benton[†]
University of Oxford

Abstract

Recent works have proposed that activations in language models can be modelled as sparse linear combinations of vectors corresponding to features of input text. Under this assumption, these works aimed to reconstruct feature directions using sparse coding. We develop metrics to assess the success of these sparse coding techniques and test the validity of the linearity and sparsity assumptions. We show our metrics can predict the level of sparsity on synthetic sparse linear activations, and can distinguish between sparse linear data and several other distributions. We use our metrics to measure levels of sparsity in several language models. We find evidence that language model activations can be accurately modelled by sparse linear combinations of features, significantly more so than control datasets. We also show that model activations appear to be sparsest in the first and final layers.

1 Introduction

Over the past decade, neural networks have demonstrated remarkable performance in many domains, including natural language processing [11, 35], computer vision [45], protein structure modelling [23] and complex strategy games [41, 5]. However, our capacity to interpret these models lags far behind. Being able to reliably and automatically determine the features used by such models – that is, the properties of the model inputs that the model extracts and uses for its predictions – would be a major step forward in our ability to interpret and safely deploy such models.

Many methods have been proposed for extracting model features, including feature visualization [19, 31], saliency maps [42] and feature importance scores [25]. Feature extraction for language models is less well developed, though recent techniques can locate facts within models [26] and show how models perform certain linguistic operations [21]. Nevertheless, most approaches to language model interpretability require human input at key stages and scale poorly to extracting many features.

One obstacle is *polysemantic neurons* in language model MLP layers – neurons that respond to multiple unrelated features [32, 17]. One hypothesised explanation for polysemanticity is that in the absence of co-occuring features, models use one neuron to represent several features without loss of accuracy. Toy models for this phenomenon, known as superposition, have been proposed [17, 39]. These models make two key assumptions: the *linear representation hypothesis*, that activations in neural networks can be decomposed as a linear combination of vectors corresponding to individual features, and the *sparsity hypothesis*, that only a small number of features are active in any input [17].

Under these assumptions, decomposing language model activations into components corresponding to different features is an instance of sparse coding [34, 17, 40]. Recent works have applied sparse coding to automatically decompose language model activations into sparse linear combinations of interpretable feature vectors [48, 15]. However, these methods have assessed their decompositions by how easy it is to find plausible descriptions of the set of inputs on which a particular found feature is active. Such metrics can indicate how successful these methods are at finding interpretable feature directions, but say little about the accuracy of the underlying assumptions of linearity and sparsity.

[*]Equal contribution
[†]Corresponding author, benton@stats.ox.ac.uk

NeurIPS 2023 Workshop on Socially Responsible Language Modelling Research

The main contribution of this work is to introduce more rigorous and quantitative ways of measuring the success of sparse coding methods applied to language models. This allows us to properly assess the extent to which activations can accurately be modelled as sparse linear combinations of feature vectors, and see how the level of sparsity depends on properties of the model. In summary, we:

- Propose novel metrics for measuring the success of sparse coding on neural network activations, and demonstrate their robustness on synthetic activations;
- Use our metrics to provide quantitative evidence that neural network activations can be accurately modelled as sparse linear combinations of feature vectors, supporting claims in earlier works [48, 15];
- Provide a more thorough analysis of the success of sparse coding as compared to previous works, including studying the relative sparsity of various model types and across layers.

We provide a more thorough review of related literature in Appendix A.

2 Background

Formally, the *linear representation hypothesis* can be interpreted in the following way. At a fixed layer of a given neural network, we assume that the activations represent m different ground-truth features $\mathcal{F}_1, \ldots, \mathcal{F}_m$, each encoded by a feature vector $\mathbf{f}_1, \ldots, \mathbf{f}_m \in \mathbb{R}^d$. An input containing features $\mathcal{F}_{i_1}, \ldots, \mathcal{F}_{i_k}$ should be represented by an activation vector which is a linear combination of $\mathbf{f}_{i_1}, \ldots, \mathbf{f}_{i_k}$. Conversely, any activation $\mathbf{x} \in \mathbb{R}^d$ should be approximately decomposable as a linear sum $\mathbf{x} \approx \alpha_{i_1}\mathbf{f}_{i_1} + \cdots + \alpha_{i_k}\mathbf{f}_{i_k}$ where $\mathcal{F}_{i_1}, \ldots, \mathcal{F}_{i_k}$ are the features present in the input producing activation $\mathbf{x}$ and $\alpha_{i_1}, \ldots, \alpha_{i_k}$ are non-negative feature coefficients. We call m the *dictionary size* and d the *embedding size*. For notational convenience we combine the feature vectors into a matrix $\Phi \in \mathbb{R}^{d\times m}$ whose columns are $\mathbf{f}_1, \ldots, \mathbf{f}_m$.

The *sparsity hypothesis* says that a typical activation $\mathbf{x}$ only requires a small number of features to approximately represent it as a linear combination of those features, that is, we can find $\alpha_{i_1}, \ldots, \alpha_{i_k}$ such that $\mathbf{x} \approx \alpha_{i_1}\mathbf{f}_{i_1} + \cdots + \alpha_{i_k}\mathbf{f}_{i_k}$ and $k \ll d$ [2, 17]. In practice, we expect that most features of an input should be active on only relatively few inputs, making sparsity a natural assumption [2, 17].

In our language model setting, we observe a set of intermediate activations $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$ from a given layer. Our task is to reconstruct the feature vectors $\mathbf{f}_1, \ldots, \mathbf{f}_m$ corresponding to the ground-truth features. If the set of activations is fixed, we may combine them into a matrix $X \in \mathbb{R}^{d\times n}$ with columns $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$ and formulate our problem as finding $\Phi \in \mathbb{R}^{d\times m}$ and $\alpha \in \mathbb{R}^{m\times n}$ such that $X = \Phi\alpha + \varepsilon$ where $\alpha \succeq 0$ is sparse and ε is small, typically in L^2 norm. (Alternatively, we may consider our activations to be drawn from some activation distribution; see Appendix B for details.)

For a fixed matrix X of activations, the sparse coding problem is frequently solved by minimizing

$$\mathcal{L}(\Phi; \alpha) = \frac{1}{n}\left(\|X - \Phi\alpha\|_2^2 + \lambda\|\alpha\|_1\right), \tag{1}$$

which we call the the *sparse coding* objective, over $\alpha \in \mathbb{R}^{m\times n}$ and $\Phi \in \mathbb{R}^{d\times m}$, where the L^1 norm is viewed as a continuous relaxation of the L^0 sparsity norm and λ is a hyperparameter controlling the degree of sparsity regularization. We constrain the columns of Φ to have L^2 norm 1 and minimise $\mathcal{L}(\Phi)$ using an iterative optimization procedure similar to that of [4, 48], as described in Appendix C.

For our metrics defined later, it will be convenient to define $\alpha(\Phi) = \arg\min_{\alpha\in\mathbb{R}^{m\times n}} \mathcal{L}(\Phi; \alpha)$, i.e. the optimal choice of α for a given Φ, and $\mathcal{L}(\Phi) = \mathcal{L}(\Phi, \alpha(\Phi))$, i.e. the value of the sparse coding objective for a given Φ, assuming an optimal choice of α. In practice, we approximate $\alpha(\Phi)$ using the same optimization procedure described in Appendix C for minimizing (1).

3 Metrics for Success of Sparse Dictionary Learning

For a sparsity metric to be meaningful, it should be invariant under scaling all activations by the same factor, continuous with respect to the activations (since we expect activations not to be an exact linear combination of feature vectors, so our metric should be robust to small perturbations) and ideally intuitive. For data which is genuinely generated using a sparse linear mechanism with an average of k fully active features, we'd like our metric to be approximately k. We consider four metrics, the first two of which have been previously considered [40], and two which are novel as far as we are aware.

Non-zero entries: The most natural metric of success is the average number of non-zero entries in the coefficient vector α, which we denote $\mathcal{N}_0(\Phi) = \frac{1}{n}\|\alpha(\Phi)\|_0$. Though $\mathcal{N}_0(\Phi)$ is intuitive and invariant under scaling, we find that it is not robust in practice.

Final loss value: Second, we consider using $\mathcal{L}(\Phi)$ for the final value of Φ [40]. Though this metric is continuous, it is not invariant under scaling and so does not give useful comparisons across different models or layers within a model.

Average coefficient norm: Third, we consider $\mathcal{S}_p(\Phi) = \|\alpha(\Phi)\|_p^p / \|\alpha(\Phi)\|_\infty^p$ for each $p > 0$. This corresponds to the average L^p norm of the coefficient vector, normalized by the average maximum coefficient. This is scale invariant and, in the case where all feature coefficients are either zero or equal to the same positive value, $\mathcal{S}_p(\Phi)$ corresponds to the average number of features present. In practice, we typically use $p = 1$ for simplicity and robustness.

Normalized loss: We define the normalized loss as $\mathcal{L}_{\text{norm}}(\Phi) = \mathcal{L}(\Phi)/(\lambda\|\alpha(\Phi)\|_\infty)$. This is similar to the final loss value, but normalized by the average magnitude of the feature coefficients, restoring invariance to scaling. Intuitively, if we perfectly reconstruct the activations so that $\mathbf{x} = \Phi\alpha$ always, then $\mathcal{L}(\Phi) = \lambda\|\alpha(\Phi)\|_1$ and so $\mathcal{L}_{\text{norm}}(\Phi) = \mathcal{S}_1(\Phi)$. In practice, reconstruction is somewhat imperfect, in which case $\mathcal{L}_{\text{norm}}$ also includes a penalty for the unreconstructed part. This makes it more stable over a range of hyperparameters.

3.1 Experimental Verification of Metrics

We now test the effectiveness of the four proposed metrics by evaluating them on various synthetic datasets where we know the true level of sparsity. First, we test how well our metrics can predict the true level of sparsity for sparse linear data. To do this, we generate synthetic activation distributions that satisfy the sparsity and linearity hypotheses with varying average numbers a of active features. Details of the generating process are given in Appendix D. We then decompose our synthetic activations using sparse coding and observe how well our metrics predict the true sparsity.

We plot the true sparsity a against each metric in Fig. 1(a). We see that for metric values less than 20, the normalized loss and average coefficient norm closely approximate the true sparsity, while plateauing for greater values. This suggests that these metrics can reliably approximate the correct sparsity level for low sparsities. Meanwhile, number of non-zero features overestimates the true level of sparsity. The final loss is not scale-invariant, so it cannot be compared to the true sparsity.

Second, we test how well our metrics can distinguish between sparse linear data and other datasets. We construct three sparse linear datasets and three non-sparse linear datasets (details in Appendix D),

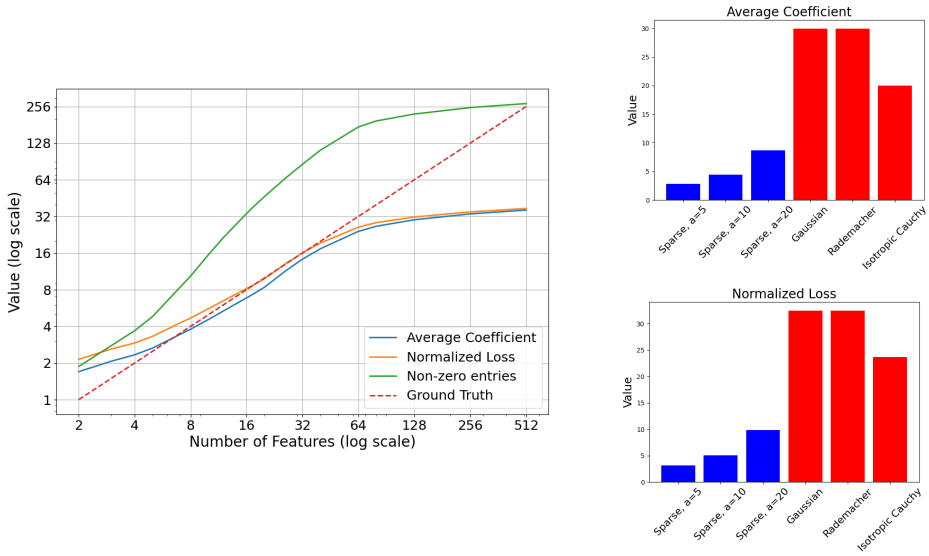

Figure 1: (a) Metric values compared to true sparsity level for synthetic data. (b) Metric values for sparse linear data (blue) compared to non-sparse linear data (red).

decompose each of the datasets using sparse coding and apply our metrics. The results for average coefficient and normalized loss are shown in Fig. 1(b). We see that the average coefficient norm and normalized loss perform very similarly, and both clearly distinguish between the sparse linear data and the other datasets. The results for non-zero entries and final loss are shown in Appendix D; both perform less well and so we focus on average coefficient norm and normalized loss from now on.

We also consider ablations of both experiments where we vary embedding size, dictionary size, noise level and number of ground truth features. We find our results are robust to changes in noise level, dictionary size and number of ground-truth features, and relatively robust to changes in embedding size, provided the level of sparsity is not too close to the embedding size. Performance may deteriorate if the number of ground-truth features is too much larger than the embedding dimension, since then our linear sparse data starts to look approximately Gaussian. For details, see Appendix E.

4 Demonstrating Sparsity in Language Model Activations

Now that we have demonstrated that average coefficient norm and normalized loss can reliably distinguish between sparse linear activations and some other classes of distributions, we use them to study sparsity in language model activations. In this section, we focus on the normalized loss, as it is more robust in practice; we present results using average coefficient norm in the appendices.

4.1 Embedding Layers

First, we use our metrics to assess sparsity in the embedding layers of transformer language models. We pick three classes of models to test on: BERT (Tiny, Mini, Small and Medium) [43, 7], TinyStories (1M, 3M and 33M) [16], and GPT-Neo/GPT-2 [9, 37]. We use the token embeddings as our set of activations X, apply sparse coding, and measure the sparsity of the resulting decomposition using normalized loss. See Appendix F for experimental details and results for average coefficient.

The normalized loss values we obtain are displayed in Fig. 2, where we have also plotted the sparsity value achieved for a standard Gaussian distribution for reference. We observe that for all models the normalized loss of the token embeddings is much lower than that for a Gaussian, as we would expect if the linearity and sparsity hypotheses held for the embeddings.

Note also that if we compare the normalized loss values in Fig. 2 to the results in Section 3.1 and Appendix E, we see they are within the range where normalized loss correctly predicts ground-truth sparsity on synthetic sparse linear data. This suggests that the normalized loss values in Fig. 2 are a good approximation to the true sparsity level, assuming the linearity and sparsity hypotheses.

Second, we observe that the number of active features increases as the embedding size increases, but more slowly. This matches the intuition that the model can disentangle more semantic meanings in a

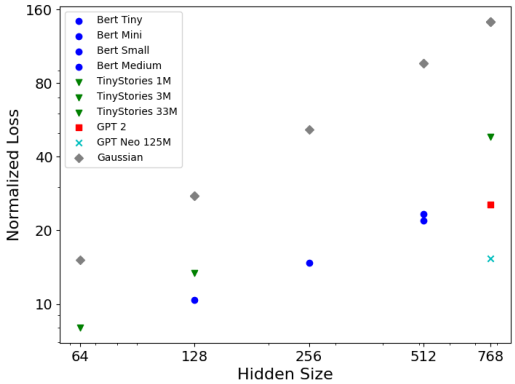

Figure 2: Embedding size versus normalized loss for token embeddings of three different classes of language models.

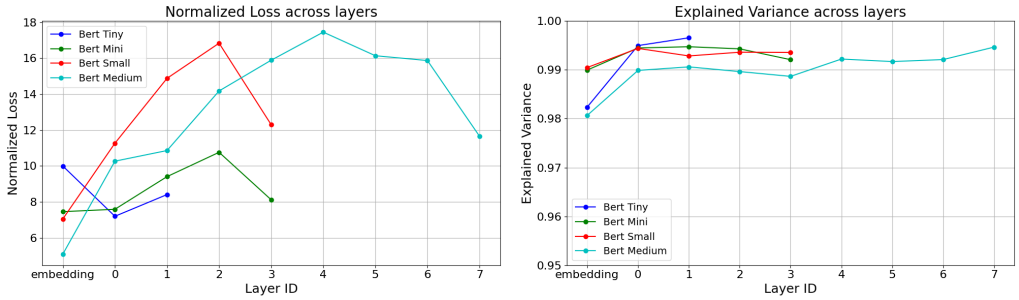

Figure 3: (a) Sparsity of activations by layer for four BERT models. (b) Percentage of activation variance explained by our sparse decomposition for each layer and model.

higher dimensional space. We also observe that the BERT architecture leads to slightly more sparse representations than the TinyStories architecture.

As observed in previous work, the feature decompositions produced by our sparse coding method do frequently correspond to natural human interpretations. In Appendix G, we provide examples where our feature decomposition splits tokens into multiple semantically meaningful components, and indicate that the feature directions that we find may be more interpretable than arbitrary directions in embedding space, corroborating the findings of Yun et al. [48] and Cunningham et al. [15].

4.2 Later Layers

Next, we explore how the level of sparsity changes across layers of a language model. We apply our sparse coding method to all layers of four BERT models (Tiny, Mini, Small and Medium), with activations generated by running the models on Wikipedia abstracts, and plot the resulting normalized loss metric. See Appendix H for experimental details and results using average coefficient.

The results are plotted in Fig. 3. We find that all layers are sparse according to the normalized loss. In addition, there is a general trend that the embedding layer is the most sparse, with sparsity first decreasing as layer depth increases, before becoming more sparse again in the latest layers. This is consistent with the intuition that the model may initially be detecting a larger number of higher-level features in deeper layers, leading to a reduction in sparsity as layer index increases, before having to extract just the key features for the next token prediction, leading to a subsequent decrease in features stored in the final layers and corresponding increase in the sparsity level.

5 Social Impacts

In this work, we have developed novel metrics for measuring the success of sparse coding, we have demonstrated that our metrics can accurately predict the level of sparsity for synthetic models of sparse linear activations, and we have applied these metrics to measure sparsity in language models. We show that activations across various model architectures and all layers were more sparse than several control datasets, providing evidence for the linearity and sparsity hypotheses, and find that activation sparsity is greatest in the first and final layers of a model.

Increasing the interpretability and transparency of language models is a central tool for ensuring that their deployment is safe and broadly beneficial. Determining the features that these models are relying on to make their predictions is a key step in rendering them interpretable. Sparse coding is one particularly promising avenue for feature extraction, since it has demonstrated initial success and runs automatically and so has the potential to scale well to large models.

To improve sparse coding for feature extraction, it is important to be able to reliably assess the performance of current methods and the accuracy of the assumptions on which they rely. We hope that by providing better quantitative ways to assess feature sparsity and superposition within language models, we can encourage more detailed and precise studies of these methods, including more rigorous tests of the linearity and sparsity hypotheses, further investigations into the factors that affect the degree of superposition within models, and improved methods to disentangle superposed features.

References

[1] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[2] S. Arora, Y. Li, Y. Liang, T. Ma, and A. Risteski. Linear Algebraic Structure of Word Senses, with Applications to Polysemy. *Transactions of the Association for Computational Linguistics*, 6:483–495, 2018.

[3] S. Bach, A. Binder, G. Montavon, F. Klauschen, K. R. Müller, and W. Samek. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLOS ONE*, 10, 2015.

[4] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[5] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, et al. Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv preprint arXiv:1912.06680*, 2019.

[6] D. Berthelot, C. Raffel, A. Roy, and I. Goodfellow. Understanding and Improving Interpolation in Autoencoders via an Adversarial Regularizer. In *International Conference on Learning Representations*, 2019.

[7] P. Bhargava, A. Drozd, and A. Rogers. Generalization in NLI: Ways (Not) To Go Beyond Simple Heuristics. In *EMNLP 2021 Workshop on Insights from Negative Results*, 2021.

[8] S. Bills, N. Cammarata, D. Mossing, H. Tillman, L. Gao, G. Goh, I. Sutskever, J. Leike, J. Wu, and W. Saunders. Language models can explain neurons in language models. *OpenAI Blog*, 2023. URL `https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html`.

[9] S. Black, G. Leo, P. Wang, C. Leahy, and S. Biderman. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow, 2021.

[10] P. Bojanowski, A. Joulin, D. Lopez Paz, and A. Szlam. Optimizing the Latent Space of Generative Networks. In *International Conference on Machine Learning*, 2018.

[11] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, et al. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, 2020.

[12] C. Burns, H. Ye, D. Klein, and J. Steinhardt. Discovering Latent Knowledge in Language Models Without Supervision. In *International Conference on Learning Representations*, 2023.

[13] L. Chan, A. Garriga-Alonso, N. Goldowsky-Dill, R. Greenblatt, J. Nitishinskaya, A. Radhakrishnan, B. Shlegeris, and N. Thomas. Causal scrubbing: A method for rigorously testing interpretability hypotheses. *AI Alignment Forum*, 2022. URL `https://www.alignmentforum.org/posts/JvZhhzycHu2Yd57RN/causal-scrubbing-a-method-for-rigorously-testing`.

[14] A. Conmy, A. N. Mavor-Parker, A. Lynch, S. Heimersheim, and A. Garriga-Alonso. Towards Automated Circuit Discovery for Mechanistic Interpretability. *arXiv preprint arXiv:2304.14997*, 2023.

[15] H. Cunningham, A. Ewart, L. Riggs, R. Huben, and L. Sharkey. Sparse Autoencoders Find Highly Interpretable Features in Language Models. *arXiv preprint arXiv:2309.08600*, 2023.

[16] R. Eldan and Y. Li. TinyStories: How Small Can Language Models Be and Still Speak Coherent English? *arXiv preprint arXiv:2305.07759*, 2023.

[17] N. Elhage, T. Hume, C. Olsson, N. Schiefer, T. Henighan, S. Kravec, Z. Hatfield-Dodds, R. Lasenby, D. Drain, C. Chen, R. Grosse, S. McCandlish, J. Kaplan, D. Amodei, M. Wattenberg, and C. Olah. Toy Models of Superposition. *Transformer Circuits Thread*, 2022.

[18] K. Engan, S. O. Aase, and J. H. Husoy. Method of optimal directions for frame design. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1999.

[19] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

[20] M. Faruqui, Y. Tsvetkov, D. Yogatama, C. Dyer, and N. A. Smith. Sparse Overcomplete Word Vector Representations. *Proceedings of Association for Computational Linguistics*, pages 1491–1500, 2015.

[21] A. Geiger, H. Lu, T. Icard, and C. Potts. Causal Abstractions of Neural Networks. In *Advances in Neural Information Processing Systems*, 2021.

[22] G. Ilharco, M. T. Ribeiro, M. Wortsman, S. Gururangan, L. Schmidt, H. Hajishirzi, and A. Farhadi. Editing Models with Task Arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

[23] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. Kohl, A. J. Ballard, A. Cowie, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

[24] H. Lee, A. B. Rajat, R. Andrew, and Y. Ng. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems*, 2006.

[25] S. M. Lundberg, P. G. Allen, and S.-I. Lee. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems*, 2017.

[26] K. Meng, D. Bau, A. Andonian, and Y. Belinkov. Locating and Editing Factual Associations in GPT. In *Advances in Neural Information Processing Systems*, 2022.

[27] T. Mikolov, W. T. Yih, and G. Zweig. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013.

[28] B. Millidge and S. Black. The Singular Value Decompositions of Transformer Weight Matrices are Highly Interpretable. *AI Alignment Forum*, 2022. URL `https://www.alignmentforum.org/posts/mkbGjzxD8d8XqKHzA/the-singular-value-decompositions-of-transformer-weight`.

[29] N. Nanda, L. Chan, T. Lieberum, J. Smith, and J. Steinhardt. Progress measures for grokking via mechanistic interpretability. In *International Conference on Learning Representations*, 2023.

[30] T. Oikarinen and T.-W. Weng. CLIP-Dissect: Automatic Description of Neuron Representations in Deep Vision Networks. In *International Conference on Learning Representations*, 2023.

[31] C. Olah, A. Mordvintsev, and L. Schubert. Feature Visualization. *Distill*, 2017. URL `https://distill.pub/2017/feature-visualization/`.

[32] C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, and S. Carter. Zoom In: An Introduction to Circuits. *Distill*, 2020. URL `https://distill.pub/2020/circuits/zoom-in/`.

[33] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

[34] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, 1997.

[35] OpenAI. GPT-4 Technical Report. *arxiv preprint arXiv:2303.08774*, 2023.

[36] T. Pham Van, T. M. Nguyen, N. N. Tran, H. V. Nguyen, L. B. Doan, H. Q. Dao, T. T. Minh, H. Quoc Viet, T. Liem, and H. Noi. Interpreting the Latent Space of Generative Adversarial Networks using Supervised Learning. In *International Conference on Advanced Computing and Applications*, 2020.

[37] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language Models are Unsupervised Multitask Learners, 2019.

[38] S. Ravfogel, M. Twiton, Y. Goldberg, and R. Cotterell. Linear Adversarial Concept Erasure. In *International Conference on Machine Learning*, 2022.

[39] A. Scherlis, K. Sachan, A. S. Jermyn, J. Benton, J. Steinhardt, and B. Shlegeris. Polysemanticity and Capacity in Neural Networks. *arXiv preprint arXiv:2210.01892*, 2022.

[40] L. Sharkey, D. Braun, and B. Millidge. Taking features out of superposition with sparse autoencoders. *AI Alignment Forum*, 2022. URL `https://www.alignmentforum.org/posts/z6QQJbtpkEAX3Aojj/interim-research-report-taking-features-out-of-superposition`.

[41] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[42] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *International Conference on Learning Representations*, 2014.

[43] I. Turc, M.-W. Chang, K. Lee, and K. Toutanova. Well-Read Students Learn Better: The Impact of Student Initialization on Knowledge Distillation. *CoRR*, 2019.

[44] A. M. Turner, L. Thiergart, D. Udell, G. Leech, U. Mini, and M. MacDiarmid. Activation Addition: Steering Language Models Without Optimization. *arXiv preprint arXiv:2308.10248*, 2023.

[45] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[46] K. Wang, A. Variengien, A. Conmy, B. Shlegeris, and J. Steinhardt. Interpretability in the Wild: a Circuit for Indirect Object Identification in GPT-2 small. In *International Conference on Learning Representations*, 2023.

[47] K. Yin and G. Neubig. Interpreting Language Models with Contrastive Explanations. In *Conference on Empirical Methods in Natural Language Processing*, 2022.

[48] Z. Yun, Y. Chen, B. A. Olshausen, and Y. Lecun. Transformer visualization via dictionary learning: contextualized embedding as a linear superposition of transformer factors. In *DeeLIO 2021 Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, 2021.

A Related Works

Linear representations There is considerable evidence that neural networks learn representations with linear structure: for example, linear operations on Word2Vec embeddings capture semantic meaning [27, 36], and linear interpolation in the latent space of GANs and VAEs can combine the features of multiple datapoints [10, 6]. In language models, several works have demonstrated success using linear techniques for locating locating information within models [26, 12] and editing model behaviors [22, 38, 44], as well as observing linear representations in reverse-engineered circuits [29]. Such observations motivated the introduction of the linear representation hypothesis in [17].

Feature extraction Methods proposed for discovering the features used by neural networks include feature visualization [19, 31, 32], saliency maps [42], feature importance scores [25], and layer-wise relevance propagation [3]. For transformer language models, common techniques include visualizing attention patterns [46, 8], gradient-based contrastive explanations [47], and more recently methods based on sparse coding (see below).

Superposition and polysemanticity The idea that text embeddings can be modelled as a linear superposition of sparse feature vectors was developed by Faruqui et al. [20] and Arora et al. [2]. Olah et al. [32] introduced the notion of polysemanticity in the context of vision models, and hypothesised that polysemantic neurons arise to allow networks to represent more features with a fixed number of neurons. The first theoretical models for polysemanticity and superposition were presented by Elhage et al. [17] and Scherlis et al. [39].

Sparse coding The sparse coding problem was introduced by [33, 34]. A variety of methods for learning the sparse dictionary have been proposed, including the method of optimal directions (MOD) [18], k-SVD [1], and methods using Lagrange duality [24]. Several works have applied sparse coding to language models using techniques based on autoencoders [40, 15], or FISTA [4, 48]. The method we use in this paper is a variant of the iterative optimization method used by [48].

Automated interpretability Recently proposed methods for automating the interpretation of neural networks include using multimodal models to automatically propose interpretations for neurons based on activation patters [30, 8], using causal scrubbing to automatically detect circuits within language models [13, 14], and using singular value decompositions of weight matrices [28]. The previous works most closely related to our current contributions are Yun et al. [48] and Cunningham et al. [15], who both use sparse coding to automatically identify a set of feature directions in activation space. They both measure the success of their methods by inspecting sets of examples where a particular feature is active and aiming to identify the common feature between the examples, using either a human labeller or a language model.

B Setup for an Activation Distribution

In practice, we can typically generate new intermediate activations at will by running our model on new text. Accordingly, it can make sense to imagine that we have an activation distribution from which we can sample, rather than a fixed finite set of given activations $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$. With this setup, our problem amounts to finding $\Phi \in \mathbb{R}^{d \times m}$ such that for an activation $\mathbf{x}$ drawn from this distribution we can find $\alpha \in \mathbb{R}^m$ and ε such that $\mathbf{x} = \Phi\alpha + \varepsilon$, where α is typically sparse and ε is small in expectation.

For convenience, in this paper we will stick to notating the case of a fixed matrix X of activations, but all quantities discussed can be trivially extended to the case where we draw samples $\mathbf{x}$ from the distribution of activations – wherever we sum over rows of X or α, this should instead be replaced by an expectation over the activation distribution.

C Details of Sparse Coding Algorithm

The algorithm used in this paper for minimizing the sparse coding objective $\mathcal{L}(\Phi; \alpha)$ from (1) over $\alpha \in \mathbb{R}^{m \times n}$ and $\Phi \in \mathbb{R}^{d \times m}$ is as follows. It is inspired by the iterative optimization procedure used by Yun et al. [48].

We alternate between the following two steps. First, we minimize the objective $\mathcal{L}(\Phi; \alpha)$ over all $\Phi \in \mathbb{R}^{d \times m}$ using several steps of stochastic gradient descent. Second, we update α using a greedy

procedure. For each column $\mathbf{x}^{(j)}$ of X, we find the current feature vector $\mathbf{f}_i$ that has the largest dot product with $\mathbf{x}$, say $\mathbf{f}_{i_1}$, and set $\alpha_{i_1,j} = \mathbf{x} \cdot \mathbf{f}_{i_1} - \frac{1}{2}\lambda$ (this choice corresponds to projection under the L^1 norm). If $\alpha_{i_1,j} > 0$, then we subtract $\alpha_{i_1,j}\mathbf{f}_{i_1}$ from $\mathbf{x}^{(j)}$, find the feature vector with the next largest dot product and repeat. We do this until $\alpha_{i_k,j} \leq 0$ at some step, at which point we set all remaining $\alpha_{s,j} = 0$. We repeat this for each column of X to find the new matrix $\alpha \in \mathbb{R}^{m \times n}$.

We alternate these two steps, updating Φ and α sequentially until the process converges. In the case where the activations are drawn from a distribution rather than given by a fixed matrix X, we resample the activations at each iteration.

We find empirically that the best choice of λ is approximately 10% of the maximum activation, i.e. $\lambda \approx 0.1\|\alpha\|_\infty$. Where it is computationally feasible, we use an adaptive scheme to set λ, first guessing a reasonable choice of λ, then running our sparse coding procedure once to get a feature coefficient matrix α, then updating $\lambda = 0.1\|\alpha\|_\infty$ and iterating until convergence. In practice, we find that we typically only need a couple of steps to converge.

D Further Details on Metric Verification Experiments

To generate synthetic sparse linear data with a features active on average, we do the following. First, we generate $m = 4d$ feature vectors $\mathbf{f}_1, \ldots, \mathbf{f}_m \in \mathbb{R}^d$ sampled uniformly from the unit sphere. Then, to generate each activation we sample a vector α of feature coefficients by taking each coefficient to be uniform between 0 and 1 with probability $a/4d$ and zero otherwise. We define the proto-activation to be $\hat{\mathbf{x}} = \sum_i \alpha_i \mathbf{f}_i$, so that on average each proto-activation is the sum of a activated vectors. We center the set of proto-activations and finally add Gaussian noise of variance $a\sigma^2/d$ to each proto-activation to get our synthetic activations.

Note that since each active feature is weighted by a factor distributed uniformly between 0 and 1, the expected weighted number of active features will be $a/2$. Hence, we consider the "correct" value of our metric on this dataset to be $a/2$ (rather than a).

In our initial experiments in Section 3.1 we take $d = 256$, $\sigma = 0.1$ and use 16384 datapoints and a dictionary size of $8d$. We consider the effects of varying the embedding size, noise level, number of ground-truth features and dictionary size in Appendix E.

For our three non-sparse linear datasets, we use (i) a Gaussian distribution with identity covariance (not sparse), (ii) an heavy-tailed isotropic distribution constructed by sampling from an isotropic Gaussian and then scaling $\|\mathbf{x}\|_2$ to be Cauchy distributed (heavy-tailed but not sparse), and (iii) a d-dimensional Rademacher distribution (sparse but not well-represented by a sparse linear combination of features). Each distribution is scaled to have identity covariance.

Fig. 4 shows the results of attempting to use non-zero entries and final loss to distinguish between sparse linear data and other datasets. We see that non-zero entries fails to reliably distinguish the heavy-tailed but non-sparse data, while the final loss fails to reliably distinguish the Gaussian data from the sparse linear data. In order to make the final loss values comparable across datasets, we scale all datasets to have mean 0 and the average L_2 norm equal 1.

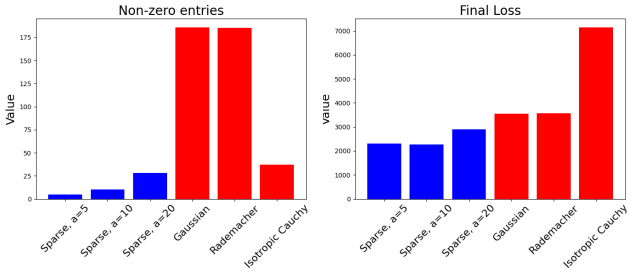

Figure 4: Metric values for sparse linear data (blue) compared to non-sparse linear data (red), with non-zero entries and final loss metrics.

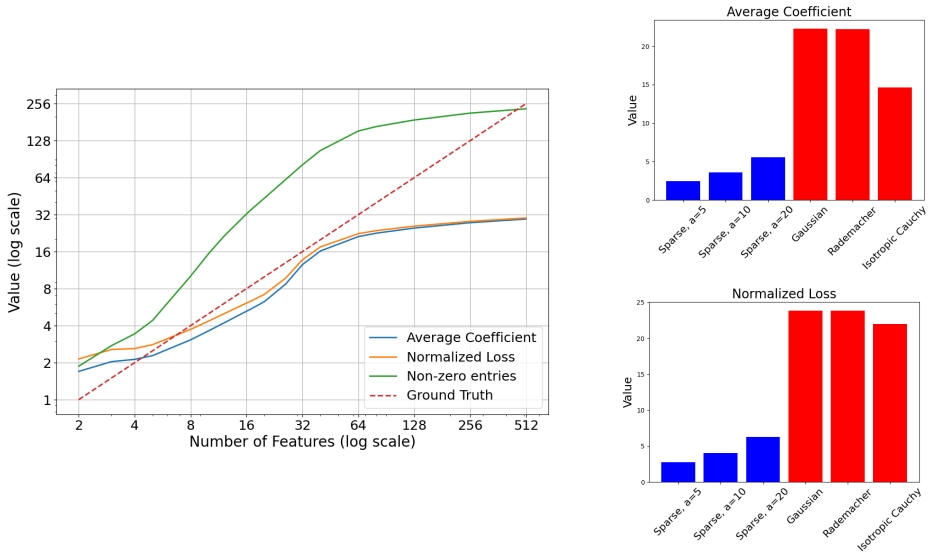

Figure 5: Results of experiment from Section 3.1 with dictionary size of $16d$. (a) Metric values compared to true sparsity level for synthetic data. (b) Metric values for sparse linear data (blue) compared to non-sparse linear data (red).

E Ablations for Metric Verification Experiments

In this section we assess the robustness of the results from Section 3.1 to changes in the parameters of the problem definition. We consider changing the level of noise in the construction of our synthetic dataset σ, the size of the embedding dimension d, the dictionary size, and the number of ground-truth features.

First, we assess the effect of changing the dictionary size. We keep the same embedding dimension $d = 256$, the same number of ground-truth features $4d$ and the same noise level $\sigma = 0.1$ as in the main text, but consider increasing the dictionary size to $16d$. We plot the results of this experiment in Fig. 5. We see that the results are very similar to Fig. 1; the average coefficient and normalized loss metrics continue to distinguish well between the sparse linear data and the other datasets, and both metrics track the ground-truth sparsity well for metric values below approximately 16. We conclude that our methods are relatively robust to dictionary sizes that are misspecified up to a factor of at least 4.

Second, we assess the effect that changing the noise level σ has. We keep the same embedding dimension $d = 256$, the same dictionary size of $8d$ and the same number of ground-truth features at $4d$ as in the main text, but consider decreasing σ to 0.05 or increasing it to 0.2. The results are plotted in Fig. 6. We see that both the average coefficient and the normalized loss metric continue to distinguish well between sparse linear and other data, and both metrics track the ground-truth sparsity well up to a metric value of about 20.

Third, we assess the effect of changing the embedding size. We fix the dictionary size, number of ground-truth features and noise level as in Section 3.1 but consider taking the embedding size to be either 64 or 512. The results are shown in Fig. 7. We continue to achieve good results for larger embedding sizes. For smaller embedding sizes, our metrics are less able to separate data with $a = 20$ and non-sparse data; this is likely because 20 features is a considerable fraction of the total embedding size when $d = 64$, and so data with $a = 20$ is approaching no longer being appreciably sparse. We see that our metrics still separate linear sparse data with $a = 5, 10$ and the other data sets reasonable well.

Finally, we consider changing the number of ground-truth features from $4d$ to $8d$ (while keeping all other parameters of the synthetic data the same). The results of the experiments from Section 3.1 are displayed in Fig. 8. We find that our metrics still accurately track the ground-truth sparsity for metric values up to about 32, and can distinguish between the sparse linear and other datasets.

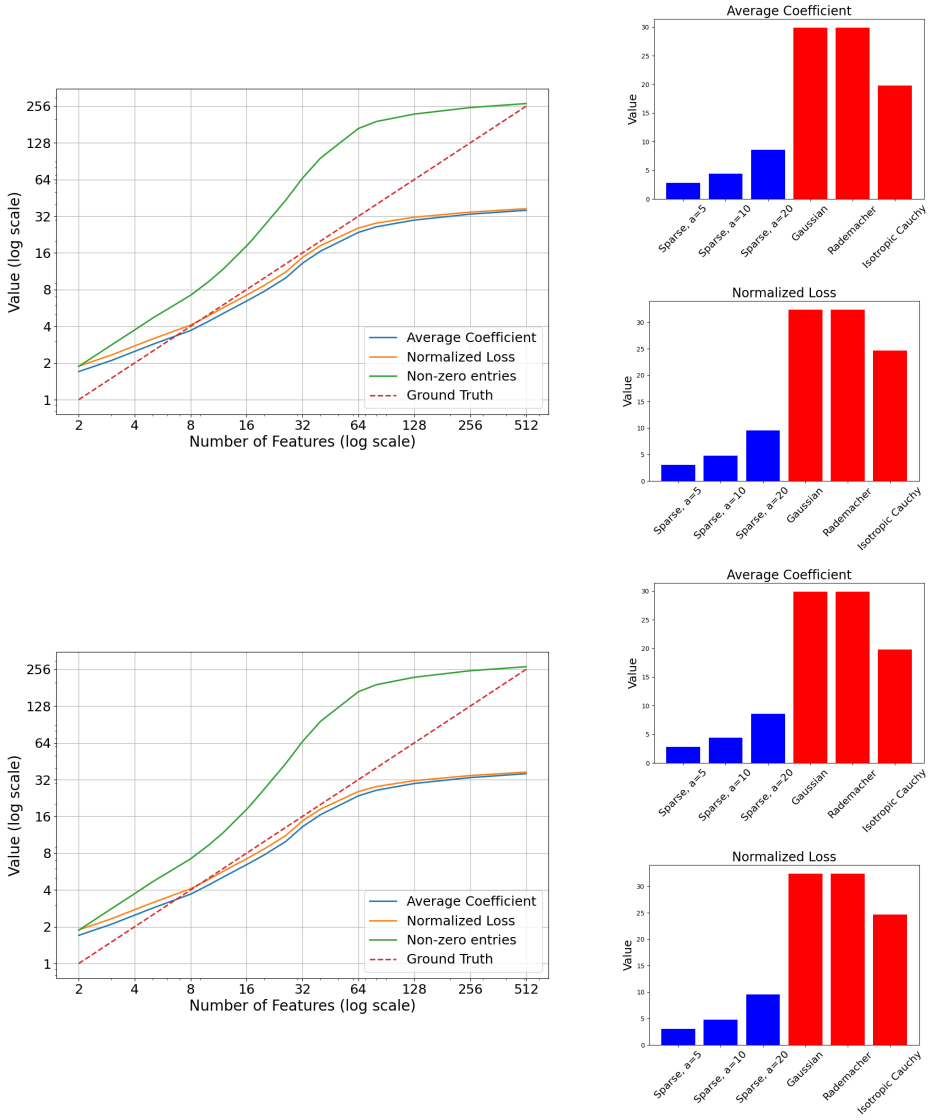

Figure 6: Results of experiment from Section 3.1 with different noise levels: $\sigma = 0.05$ (top) and $\sigma = 0.2$ (bottom). (Left) Metric values compared to true sparsity level for synthetic data. (Right) Metric values for sparse linear data (blue) compared to non-sparse linear data (red).

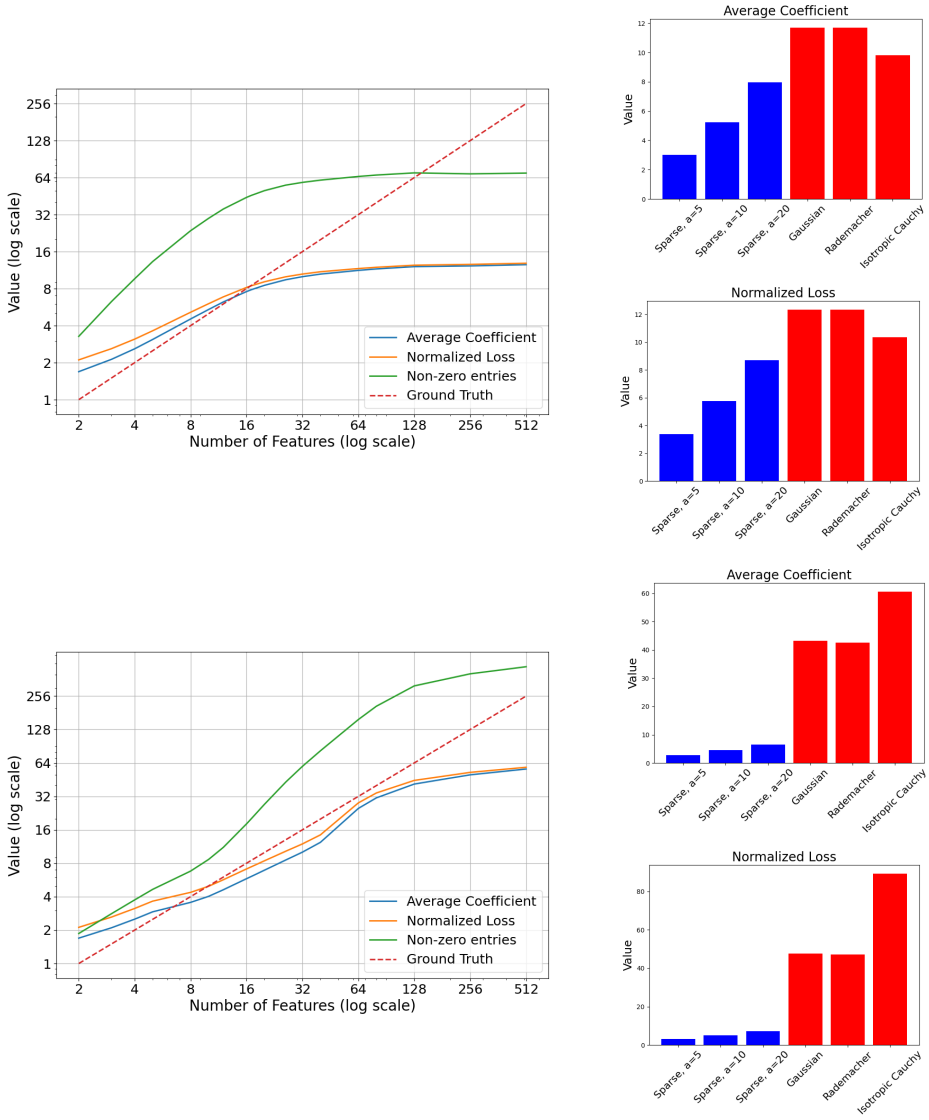

Figure 7: Results of experiment from Section 3.1 with different embedding sizes: 64 (top) and 512 (bottom). (Left) Metric values compared to true sparsity level for synthetic data. (Right) Metric values for sparse linear data (blue) compared to non-sparse linear data (red).

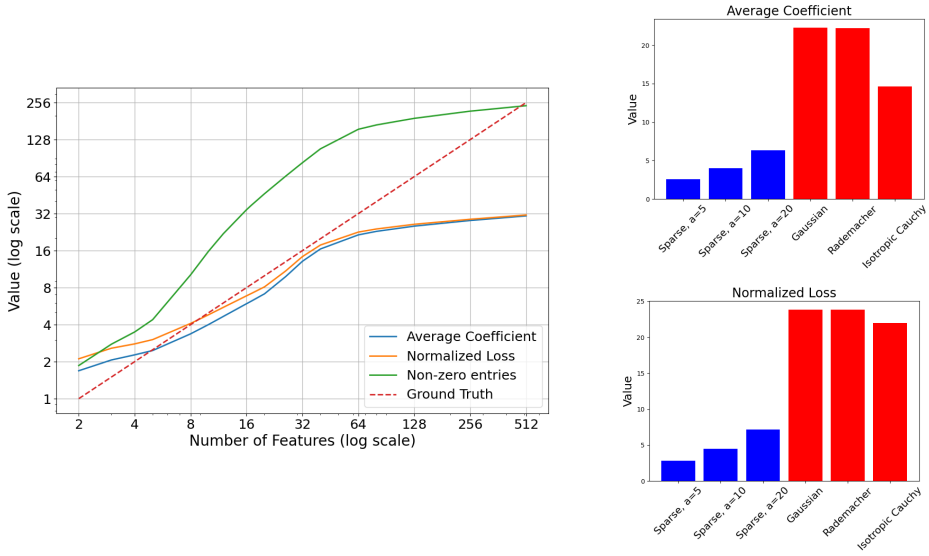

Figure 8: Results of experiment from Section 3.1 with $8d$ ground-truth features. (a) Metric values compared to true sparsity level for synthetic data. (b) Metric values for sparse linear data (blue) compared to non-sparse linear data (red).

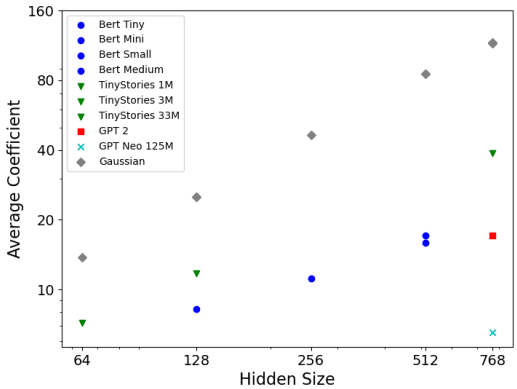

Figure 9: Embedding size versus average coefficient for token embeddings of three different classes of language models.

F Further Details on Language Model Embedding Experiments

For the experiments in Section 4 we use a dictionary size of $8d$. We also center the embedding vectors before applying our sparse coding algorithm. After applying our sparse coding algorithm, the decomposition that we find is able to explain over 90% of the variance in the activations across all models.

In Fig. 9 we show the results of the experiment of Section 4.1 but using the average coefficient metric rather than normalized loss. We find very similar results to those obtained with normalized loss.

G Examples of Interpretable Features in Embedding Space

We provide some initial evidence that the feature directions we find via sparse coding do correspond to human-interpretable features, corroborating the findings of Yun et al. [48], Cunningham et al. [15]. Given the feature vectors we found in our sparse decomposition of the embeddings of GPT-2, we pick a couple of tokens with clear meanings – `season` and `physics` – decompose them into their constituent features, pick the top three features for each with the maximum feature coefficient, and then plot the 20 tokens that maximally activate that feature.

The results are shown in Table 1 and Table 2. We see that for each feature in the decomposition, the maximally activating examples suggest a natural interpretation of that feature, as listed in the third column. For comparison, we also display the 30 words in embedding space which are closest to the embedding direction of `season` in Table 3. We note that this direction is much more polysemantic than the rows in Table 1, suggesting that our sparse coding method is succeeding at disentangling superposed meanings.

H Further Details of Experiments on Later Layers in Language Models

To form our activation distribution, we sampled model inputs from a dataset of 800k sentences from Wikipedia abstracts. For each model and layer, we use a dictionary size of $16d$. We fix $\lambda = 0.1$ during training, and then for decomposing into sparse features we use $\lambda = 0.0027, 0.0037, 0.0047, 0.01$ for BERT Tiny, Mini, Small and Medium respectively. This allows us to ensure that we explain at least 98% of the variance for all models and layers, while avoiding needing an adaptive choice of λ (for which we did not have sufficient computational resources).

In Fig. 10, we show the results of the experiment of Section 4.2 but using the average coefficient metric rather than the normalized loss. The results are qualitatively similar.

Feature	Max Activating examples	Interpretation
Feature 1	`episode`, `Episode`, `episode`, `Episode`, `episodes`, `Season`, `isode`, `isodes`, `Season`, `podcast`, `odcast`, `podcast`, `Podcast`, `season`, `flashbacks`, `Seasons`, `Ep`, `Ep`, `epis`, `season`	TV/Radio shows
Feature 2	`Winter`, `winter`, `Summer`, `summer`, `winter`, `Winter`, `Summer`, `autumn`, `Autumn`, `Spring`, `Spring`, `summers`, `winters`, `spring`, `Fall`, `spring`, `Fall`, `offseason`, `seasonal`, `Halloween`	Yearly seasons
Feature 3	`year`, `Year`, `YEAR`, `year`, `Year`, `decade`, `month`, `years`, `Years`, `Month`, `Years`, `month`, `years`, `yr`, `Month`, `Months`, `months`, `week`, `season`, `months`	Lengths of time

Table 1: Max activating examples for top 3 features in the sparse decomposition of `season`.

Feature	Max Activating examples	Interpretation
Feature 1	Physics, physics, ysics, physic, Math, Math, Chem, math, Chem, particle, chem, chemistry, asm, physicists, math, particles, asms, Chemistry, maths, Phys	Physical sciences
Feature 2	biology, psychology, economics, anthropology, neuroscience, sociology, physiology, biology, astronomy, Biology, chemistry, theology, physics, iology, Economics, Chemistry, Anthropology, ecology, mathematics, ochemistry	Fields of study
Feature 3	interstellar, galactic, galaxies, Interstellar, galaxy, asteroid, Centauri, asteroids, Galactic, astroph, Planetary, Astron, planetary, astronomers, planets, cosmic, spaceship, stellar, Nebula, astronomer	Astronomy

Table 2: Max activating examples for top 3 features in the sparse decomposition of physics.

Token	Closest embeddings
season	season, Season, season, Season, seasons, offseason, preseason, summer, episode, postseason, episodes, Episode, winter, autumn, playoff, Summer, seasonal, podcast, Winter, Ý, Year, subur, Nitrome, StreamerBot, ÃĥÃĤÃĥÃĤÃĥÃĤÃĥÃĤÃĥÃĤÃĥÃĤÃĥÃĤÃĥÃĤÃĥÃĤÃĥ..., ÃĥÃĤÃĥÃĤÃĥÃĤÃĥÃĤ, externalTo, decade, episode, Seasons

Table 3: Closest 30 words to season in embedding space.

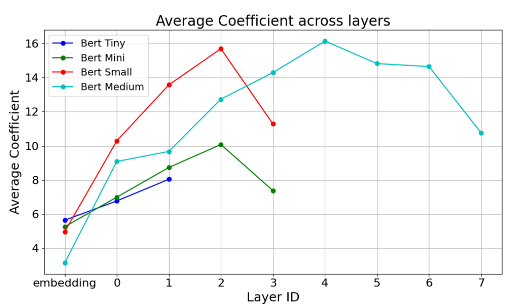

Figure 10: Sparsity of activations by layer for four BERT models, measured using average coefficient norm.