GPT-2 Through the Lens of Vector Symbolic Architectures

Johannes Knittel
Harvard University

Tushaar Gangavarapu
Cornell University

Hendrik Strobelt
IBM Research

Hanspeter Pfister
Harvard University

Abstract

Understanding the general priniciples behind transformer models remains a complex endeavor. Experiments with probing and disentangling features using sparse autoencoders (SAE) suggest that these models might manage linear features embedded as directions in the residual stream. This paper explores the resemblance between decoder-only transformer architecture and vector symbolic architectures (VSA) and presents experiments indicating that GPT-2 uses mechanisms involving nearly orthogonal vector bundling and binding operations similar to VSA for computation and communication between layers. It further shows that these principles help explain a significant portion of the actual neural weights.

1 Introduction

Understanding the way transformer models [28] work remains a complex task, impacting AI interpretability, safety, and alignment, among other areas. Successes in using linear probing to steer models [2, 18] and principal component analysis experiments on hidden embeddings [18] indicate that these models might handle linear features embedded as directions in the residual stream, which is also known as linear representation hypothesis [9, 23, 15]. Recently, sparse autoencoders have been trained for a number of related model architectures based on the idea that hidden embeddings can be broken down into a combination of sparse feature vectors [4]. One way how a linear additive model might function is by using nearly orthogonal vectors instead of arbitrary (or strictly orthogonal) directions. In high-dimensional spaces, we can have many more nearly orthogonal vectors than strictly orthogonal ones [9]. Vector symbolic architectures (VSA) [16], also known as hyperdimensional computing, leverage this principle for model construction. For example, we can create 'bag-of-concept' vectors by summing nearly orthogonal vectors, so that individual concept vectors maintain a higher dot product with the summed vector, effectively modeling an OR conjunction. The residual stream of transformers could represent such a collection of nearly orthogonal concept vectors to form a distributed code. However, current evidence is limited on whether production models like GPT effectively use such feature vectors and how. A better understanding of the inner mechanics has many implications on model interpretability and the usefulness of training sparse autoencoders, as well as on architectural design decisions. This paper explores the resemblance between decoder-only transformer architectures and vector symbolic architectures and presents experiments indicating that GPT-2 [24] uses mechanisms involving nearly orthogonal vector bundling and binding operations similar to VSA. It further shows that these principles help explain a significant portion of the actual weights in the MLP layers.

2nd Workshop on Attributing Model Behavior at Scale (ATTRIB) at NeurIPS 2024.

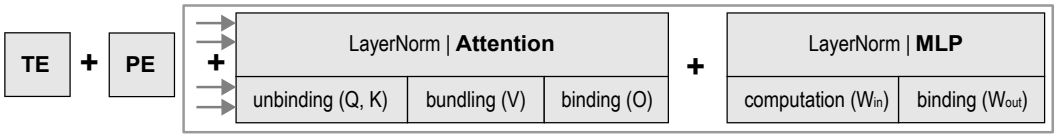

Figure 1: For each layer, the output of the attention and the feedforward network (MLP) are added to the residual stream of GPT-2, which could be interpreted as a combination of (un)binding and bundling operations using nearly orthogonal vectors.

2 Interpreting GPT-2 Through Bundlings and Bindings

In high dimensional spaces, we can have many more nearly orthogonal vectors than dimensions [9]. For an n-dimensional space, we cannot find more than n distinct vectors such that they are orthogonal to each other, i.e., the inner product (dot product for real spaces) of any vector pair is exactly zero. However, we can find much larger sets of vectors such that the dot product of any vector to any other vector is nearly zero or small. One can think about a unit sphere in which the surface is divided into small patches such that the centroids of these areas are sufficiently far away, corresponding to a large enough angle (i.e., small enough dot product). We can imagine that there are many more such patches than the number of dimensions, depending on our dot product threshold. Bundling and binding of such nearly orthogonal *concept vectors* are important principles of vector symbolic architectures [11], but similar ideas are used in many other techniques, e.g., in vector-based linear models [20].

Bundling refers to the bagging of vectors to create composite concept vectors. Imagine we have three nearly orthogonal vectors c_1, c_2, c_3, each representing a specific (binary) concept. We can bundle c_1 and c_2 into a concept vector $c_{1,2} = c_1 + c_2$ that has directional similarity to both vectors c_1 and c_2 but not to c_3. We can efficiently test whether a vector c_j is similar to either c_1 or c_1: $\langle c_{1,2}, c_j \rangle \geq \lambda$, essentially corresponding to a single perceptron with a ReLU activation function.

In addition to bundling, we can also perform a *binding* operation by multiplying a suitable matrix M with the concept vector. For instance, this allows us to model a specific order of concepts: $cm = Mc_1 + c_2$ is similar to c_2, Mc_1, and $M^{-1}c_1$, but not to c_1 (if $M \neq I$). M could be used here to model that c_1 refers to the first concept of the pair (c_1, c_2). This binding still works even if the matrix is not orthogonal but composed of nearly orthogonal vectors, i.e., $M^\intercal M$ will lead to a nearly diagonal matrix (see Appendix A).

Binding transformations can be interpreted in two ways. We can regard them as binding operators of (possibly previously bundled) concept vectors. We can also think of them as a construction of a new concept vector based on binary input features by adding those nearly orthogonal vectors (the rows of the matrix) where the corresponding input dimension is non-negative, with positive and negative signs as indications of the (binary) direction of the concept. From a purely technical standpoint, both interpretations are valid. The latter interpretation might be more helpful if we can assign semantics to specific dimensions of the input vectors (monosemanticity).

We can further build hierarchical concept vectors through bindings [30]. For instance, if we start with a set of concepts that could apply to people in general, we might first bundle the present concepts for two persons, respectively, which we can then bind with two different matrices M_1, M_2. If we want to compare whether the first person in two vectors are similar, we can unbind those vectors using the inverse of M_1 and then take the dot product. In other words, binding can be interpreted as a packaging step, whereas the unbinding process can be used to 'focus' on a specific package. We will later outline the similarities of this interpretation to causal self attention.

Throughout this paper, we will focus on GPT-2 [24] as a specific model architecture and instance of decoder-only transformers. The flow through the hidden embeddings of transformers is often referred to as residual stream since the outputs of the attention and MLP layers are always added to the previous outputs. This has motivated the training of sparse autoencoders (SAEs) on those hidden representations for disentangling the embeddings into a sum of sparse feature vectors, assuming that the residual stream represents many different concepts in superposition [4]. Both the successful training of SAEs and experiments on finding linear representations of concepts in the embedding space are hints that transformers might store and process features using directions rather than with highly dense (and somewhat arbitrary) distributed codes. However, sufficiently large SAEs could theoretically approximate any embedding (e.g., by dedicating one 'feature' for every observed training input). Further, the sole observation of linear separability for a selection of concepts does not

automatically imply that trained transformers effectively employ nearly orthogonal concept vectors and perform operations that we can interpret as matrix bindings. On the other hand, the model architecture bears similarities with vector symbolic architectures. We outline in this section how transformers, and GPT-2 in particular, could be interpreted with bundling and binding mechanisms closely related to VSAs. Figure 1 depicts a simplified architecture of GPT-2 through the lens of VSA.

2.1 Word Embeddings

The word embeddings of GPT-2 are nearly orthogonal to each other, so we could interpret them as concept vectors. They are not pure or random concept vecors, though, since related tokens do point to similar directions (e.g., pairs of different versions of the 'the' token have high dot products). An important design decision of GPT is that the unembedding matrix at the end is just the transpose of the embedding matrix. This means that the layers need to add values to the residual stream such that the combined magnitude of those additions outweigh the initial word embedding by far. The final step in predicting the actual next token is essentially a determination which direction is the strongest since we take the dot product with every word embedding vector and then assign model probabilities through the softmax function.

2.2 Attention Layer

The causal self attention mechanism in GPT-2 first computes query, key, and value vectors v_q, v_k, v_v by linearly transforming the current residual input x with square matrices W_Q, W_K, W_V (we will ignore biases for now). Let us first consider the case with just one attention head for simplicity. Then, the output of the attention added to the stream is determined by taking a weighted sum of all v_v vectors of the current and previous positions, followed by a linear projection W_O. The weights of the sum are based on the dot product between the query and respective value vectors, normalized with the softmax function. If we assume that our learned matrices W_Q, W_K, W_V are nearly orthogonal, we could interpret this process as a sequence of unbinding operations (W_Q, W_K, W_V), followed by the bundling of a selection of such unbound vectors v_v based on their 'relevance', followed by the binding of the resulting sum vector through W_O.

In other words, the attention head focuses on a specific package of possibly bundled vectors by rotating the input space appropriately. This package could be different for the current position (query) compared to previous positions (key). It then computes how similar the concepts are for the specific package. The softmax function effectively acts as a (dynamic) dot product thresholding since only those products that are sufficiently close to the maximum attention value will lead to non-zero coefficients when taking the weighted sum of the value vectors. The resulting output is bound again to obtain unique concept vectors for the specific attention layer.

Production models have more than one attention head, but this requires only minor modifications to this outlined analogy. Each attention head may be interpreted as a quantized unbinding / binding operation, that is, each head focuses on a specific package but only cares about the first few dimensions of the resulting transformation (the remaining dimensions are implicitly set to zero). Using this interpretation, attention heads therefore copy (and repackage) concepts into the current stream, sourced from specific packages of those tokens that have conceptual similarities in specific areas.

2.3 LayerNorm and Biases

In GPT-2, the attention and MLP layers first apply LayerNorm before their processing (the residual stream itself is not modified, though). LayerNorm essentially centers and normalizes the input by subtracting the mean and then dividing it by the square root of the variance. It then scales the result with channel-specific learnt weights and adds a bias to each channel. We can merge the latter step (scaling and translation) into the subsequent weights and biases of the respective layer [8]. The centering and normalization depends on the actual input, though. While the normalization does not impact the angle between vectors, this is not true for translations. Hence, accessing concepts in later layers only works if the typically observed input vectors and intermediate embeddings have a mean near zero so that distortions across layers are minimized.

The biases in W_Q, W_K only add a constant factor to the dot product, which we can ignore. The biases of W_V can be merged into the biases of W_O as the softmax attention values always add up to

one. The biases of W_O, W_{out} (attention, MLP blocks) are constant additions to the residual stream. Given that they are nearly orthogonal to the word embeddings, they could serve as additional helpers for 'cleaning' the initial input by adding vectors that gradually decrease the share of the initial input on the final result without changing the output prediction.

2.4 Processing of Concepts

The MLP blocks in GPT-2 are composed of two projections. The first projects the current embedding to an intermediate vector four times the size of the hidden dimension, followed by a nonlinear activation function (ReLU or similar). The second and final projection transforms these results back to the original hidden dimension (without nonlinearities). A single neuron in the first MLP layer could perform a simple check of whether the current stream contains a specific concept by simply performing a weighted summation in which the weights resemble the nearly orthogonal concept vector. The nonlinearity enables more advanced boolean modeling. An OR conjunction would be modeled by a less negative bias (presence of at least one concept contained in the weights already triggers activation), whereas an AND conjunction would require a more negative bias of said neuron (presence of two or more concepts that make up the weights are needed). We can model the NOT operation by subtracting the respective concept vector from the weights so that they point to the opposite direction. The second layer could store the result of this operation as a new concept using a nearly orthogonal vector. With carefully selected thresholds (biases), the MLP block in transformers could therefore model boolean functions on and using nearly orthogonal concept vectors.

If individual neurons represent individual concepts, we would severely limit the number of concepts that are effectively added after each MLP block. However, as Vaintrob et al. [27] demonstrated for binary inputs, such processing can also result in a vector that resembles the AND conjunction of many more concepts in superposition. If we can find small sets of concepts that rarely co-occur, we can assign OR conjunctions to neurons such that they trigger whenever at least one of the concepts in the set is present. Later layers can check whether one or more concepts are actually present (AND) by testing whether all nodes containing one of said concepts have fired (i.e., all related concept vectors are present in the stream).

3 Experiments

The previously outlined framework is one possibility how different layers in transformers could communicate with each other and compute functions that ultimately lead to next token predictions. We performed several experiments on GPT-2 (small) to gather evidence on the suitability of this conceptual framework for the interpretation of the inner workings. We used the TransformerLens [21] library to extract weights and activations from GPT-2 small. We disabled all in-built post-processing steps except for the merging of value biases into the projection biases of the attention output projection W_O and the incorporation of the LayerNorm scaling and biases into the respective subsequent weights and biases (MLP blocks, attention blocks, and unembedding matrix). We relied on the FAISS library [6] for an efficient implementation of dot product nearest neighbor search (HNSW algorithm).

3.1 Near Orthogonality of Vectors and Matrices

Near orthogonality of vectors and matrices are an important assumption of concept vectors and binding operations. We computed the matrix product $M^\intercal M$ for word embeddings, the attention matrices, and the output projections of the MLP blocks. Appendix B shows some results. In all cases, we observed strong diagonals with minimal artifacts, suggesting that these matrices are indeed composed of nearly orthogonal vectors. The output biases of the attention and MLP blocks are nearly orthogonal to the word embeddings. The magnitude of the attention output bias in the last layer is significantly higher, suggesting that this is a final cleaning step to ensure that the initial token embedding does not play a role anymore in the unembedding step. The mean of the word embeddings and intermediate hidden embeddings for a number of prompts are near zero (< 0.05), suggesting low distortions of the angles of directions across layers.

Table 1: Explaining neurons by comparing their weights with sums of token embeddings.

Neuron	Vector	Cos. Sim.
0-12	`CNN` + `Cold` + `NBC` + ... + `lying` + ... + `TRUMP` + ...	0.61
0-186	`remarked` + `commented` + `complained` + `said` + ...	0.63
0-192	`being` + `Being` + `Having` + `be` + `Be` + `beings` + ...	0.65
0-205	`selections` + `eclectic` + `selection` + `assortment` + ...	0.58
0-247	`resulted` + `based` + `Given` + `highlighted` + ...	0.62

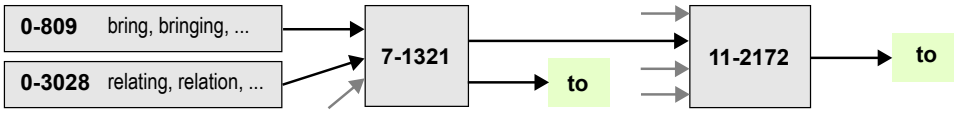

Figure 2: Circuit discovery using similarities between W_{out} and W_{in} vectors based on greedily obtained concept vectors. The two neurons in the first layer each focus on a set of tokens. The neuron 7-1321 seems to model a union of those sets, whose output is similar to the input of the depicted neuron in the last layer and the *to* unembedding. The last one has many inbound connections to previous outputs and seems to boost the concept of the *to* vector.

3.2 Processing of Concepts

Earlier experiments indicated that some neurons in GPT-2 focus on very specific tokens, such as *the*. Assuming that MLP blocks perform boolean operations on concept vectors and that word embeddings are concept vectors, we would expect some neurons to have weights representing either a single word vector or a composite vector made of several word vectors (e.g., the sum of all word embeddings that are variations of *the*). To test this hypothesis, we conducted experiments.

Specifically, we aimed to determine if some neural weights could be explained by *simple* bundled vectors that are merely sums of word embeddings. For each possible input vector (word embedding), we compiled a list of the most similar neurons. That is, we looked for neurons in the first feedforward layer of the MLP block whose input weights have a reasonably high dot product with the centered candidate vector. For each neuron and its list of candidates, we then greedily assembled the bundled vector by including those vectors (ranked by similarity) that lead to an increase of the cosine similarity between the bundled vector and the respective neural weights. We discarded vectors with a cosine similarity of less than 0.05 with the neural weight vector. We also discarded those with less than 0.1 if they did not significantly increase the overall cosine similarity by more than 0.04. These thresholds represent hyperparameters and were conservatively chosen to obtain sets of highly relevant vectors. We enforced a minimal similarity to the target vector and only considered unweighted sums since any vector can be trivially represented by a weighted sum of sufficiently many nearly orthogonal vectors.

With this simple approach, we achieve a similarity of 0.5 or higher for more than 80% of the neurons in the first layer, and at least 0.3 for more than 95% (this includes attributions to attention heads we discuss in the next section). For instance, our greedily obtained vector for neuron 1844 in the first layer has a high cosine similarity of 0.69 with the actual weights of that neuron. The list contains many first names, most of them male sounding (' Chris', ' Kevin', ' Jeff') with some exceptions (' Rebecca'). Neuronpedia explains this as *first names of people*. Neuron 20 (similarity of 0.73) focuses on tokens somehow semantically related to 'maintaining', including 'Nevertheless' and 'Storage' (Neuronpedia says *verbs related to maintaining or keeping something*). Table 1 lists more examples. It is important to note that we inferred these neuron explanations directly from the weights rather than through activation observations and correlations, or by training an autoencoder or probing classifier.

3.3 Processing Circuits

Using the analogy of concept vectors and matrix bindings, we would expect that the MLP blocks perform (boolean) functions on concept vectors sourced from either the token embeddings, previous results from MLP blocks (information from computations), or attention blocks (information from other tokens). The results are then written back to the stream via matrix binding. The sum of

all written concept vectors (including those from the attention blocks) should exhibit the highest directional similarity with the token embedding that is up next. This led us to investigate whether neural weights could be explained based on W_O vectors (output projection vectors of the attention layer) and W_{out} vectors (final projection in MLP block). This would support the hypothesis that GPT-2 communicates via near-orthogonal vectors between layers. This communication either aims to bring the stream closer to the predicted next token or serves as an input for subsequent circuits. Analogous to the previous experiment, we tried to explain neural weights with simple bundled vectors. For this set of experiments, we included output projection vectors of the attention layer W_O and of the MLP block W_{out}, in addition to the word embeddings.

We were able to reconstruct bundled vectors with a cosine similarity of at least 0.3 with neural input weights in 30-50% of neurons within the middle layers, as well as the majority of neurons in both the first and last layer. The majority of bundlings were composed of up to 40 vectors. For some neurons, our simple circuit discovery strategy explained the weights nearly perfectly. For instance, neuron 2537 in layer 1 can be explained by the non-activation of neurons 2977 and 1993 in layer 0, showing a similarity of 0.91. Neuron 1073 in layer 1 seems to be dependent on the presence of concept 11 in head 1.4 and the absence of concepts 31 in head 1.2 and 1 in head 1.10, with a weight similarity of 0.72. Figure 2 illustrates a circuit that seems to increase the likelihood of predicting the word *to* following certain words like *bring* or *relating*. When forcing the output of neuron 7-1321 to zero, the logit for the token *to* decreases for the prompt *In relation* but not for the prompt *I was listening*. Neuron 12 in the first layer appears to trigger on a specific set of tokens including names of major news outlets (e.g., CNN) and the word *TRUMP*. Using our strategy, we were unable to identify other neurons significantly influenced by the output of this neuron. However, we observed a cosine similarity of 0.07 with the token *Greenwald*. Although this seems low, it is crucial to remember that tokens potentially encompass a broad array of concepts. The model may have overfitted to news articles related to Glenn Greenwald and the US election.

The first and last layers have strong similarities to token embeddings. The residual stream in higher layers already points toward likely tokens that should be predicted next. One possible explanation is that the last layer therefore weakens or confirms current predictions by also taking into account which tokens are likely to be predicted next. It is important to note that we employed a greedy method for identifying related vectors and thus circuits, which assumes a certain degree of monosemanticity in individual neurons and attention dimensions. Our objective was to gather evidence of matrix binding and the processing of nearly orthogonal vectors, rather than to identify all communication channels comprehensively.

4 Related Work

Linear representations in models and embeddings have a long history in research, with Word2Vec [19] popularizing the idea of performing simple arithmetic on word embeddings. Numerous probing experiments [1] have been performed to investigate linear representations in large language models [26, 25, 12, 17, 2, 18]. Hernandez et al. [14] found evidence that relational decoding in large language models can often be approximated with an affine transformation. Wattenberg and Vegas [30] discuss relational composition in context of sparse autoencoders, noting that matrix bindings bear similarities with the attention process. The field of mechanistic interpretability [22] is concerned with reverse engineering and explaining (in parts) the inner workings of models, for instance, through discovering and describing circuits [13, 29], or by trying to understand whether certain behavior can be tracked down to individual neurons or is represented in a more distributed way [5, 3, 7, 9, 12].

5 Discussion and Conclusion

Our experiments indicate that GPT-2 effectively learns mechanisms that are similar to concepts of vector symbolic architectures. The different blocks seem to communicate by writing and reading nearly orthogonal vectors from the residual stream. We could also show that at least some neurons implement simple boolean functions as outlined in Section 2. This has several implications on model interpretability and architectural design decisions. The presence of bundling and binding circuits strengthens the validity of training sparse autoencoders for model disentanglement. However, such nearly orthogonal vectors may not necessarily map to interpretable features or independently understandable features (decomposability theory [9]). They may be part of what Engels et al. describe

as multidimensional features [10]. Our findings may also inform future architectural decisions. For instance, it may help to understand which architectural components promote or hinder the learning of bundling and binding operations, possibly leading to better model architectures. A better understanding of the inner workings also helps to hypothesize for which type of tasks GPT-like models work better or worse.

We want to emphasize that our findings are preliminary and our method has several limitations. We only performed experiments on GPT-2 small and further experiments are needed to confirm that these behaviors also exist in related and more recent models. We deliberately focused on very simple techniques and avoided the use of sparse autoencoders in our experiments to mitigate confounding factors and uncertainties introduced by additional modeling.

Acknowledgments and Disclosure of Funding

This work was partially supported by the Harvard Data Science Initiative Postdoctoral Fellowship.

References

[1] Guillaume Alain and Yoshua Bengio. *Understanding intermediate layers using linear classifier probes*. 2017.

[2] Andy Arditi et al. *Refusal in language models is mediated by a single direction*. arXiv: 2406.11717 [cs.LG]. 2024.

[3] David Bau et al. "Understanding the role of individual units in a deep neural network". In: *Proceedings of the National Academy of Sciences* 117.48 (2020), pp. 30071–30078. DOI: 10.1073/pnas.1907375117.

[4] Trenton Bricken et al. "Towards monosemanticity: Decomposing language models with dictionary learning". In: *Transformer Circuits Thread* (2023).

[5] Fahim Dalvi et al. *What Is One Grain of Sand in the Desert? Analyzing Individual Neurons in Deep NLP Models*. arXiv:1812.09355 [cs]. Dec. 2018. DOI: 10.48550/arXiv.1812.09355.

[6] Matthijs Douze et al. "The Faiss library". In: (2024). arXiv: 2401.08281 [cs.LG].

[7] Nadir Durrani et al. "Analyzing Individual Neurons in Pre-trained Language Models". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Bonnie Webber et al. Online: Association for Computational Linguistics, Nov. 2020, pp. 4865–4880. DOI: 10.18653/v1/2020.emnlp-main.395.

[8] Nelson Elhage et al. "A mathematical framework for transformer circuits". In: *Transformer Circuits Thread* (2021).

[9] Nelson Elhage et al. *Toy Models of Superposition*. arXiv:2209.10652 [cs]. Sept. 2022. DOI: 10.48550/arXiv.2209.10652.

[10] Joshua Engels et al. "Not all language model features are linear". In: *arXiv preprint arXiv:2405.14860* (2024).

[11] Stephen I. Gallant and T. Wendy Okaywe. "Representing objects, relations, and sequences". In: *Neural Comput.* 25.8 (Aug. 2013), pp. 2038–2078. ISSN: 0899-7667. DOI: 10.1162/NECO_a_00467.

[12] Wes Gurnee et al. *Finding Neurons in a Haystack: Case Studies with Sparse Probing*. arXiv:2305.01610 [cs]. June 2023. DOI: 10.48550/arXiv.2305.01610.

[13] Michael Hanna, Ollie Liu, and Alexandre Variengien. "How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model". In: *Thirty-seventh conference on neural information processing systems*. 2023.

[14] Evan Hernandez et al. "Linearity of relation decoding in transformer language models". In: *The twelfth international conference on learning representations*. 2024.

[15] Yibo Jiang et al. "On the origins of linear representations in large language models". In: *Forty-first international conference on machine learning*. 2024.

[16] Denis Kleyko et al. "Vector Symbolic Architectures as a Computing Framework for Emerging Hardware". In: *Proceedings of the IEEE* 110.10 (Oct. 2022), pp. 1538–1571. ISSN: 0018-9219, 1558-2256. DOI: 10.1109/JPROC.2022.3209104.

[17] Kenneth Li et al. "Emergent world representations: Exploring a sequence model trained on a synthetic task". In: *The eleventh international conference on learning representations*. 2023.

[18] Samuel Marks and Max Tegmark. *The Geometry of Truth: Emergent Linear Structure in Large Language Model Representations of True/False Datasets*. arXiv:2310.06824 [cs]. Aug. 2024. DOI: 10.48550/arXiv.2310.06824.

[19] Tomas Mikolov et al. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems* 26 (2013).

[20] Jeff Mitchell and Mirella Lapata. "Vector-based models of semantic composition". In: *Proceedings of ACL-08: HLT*. Ed. by Johanna D. Moore et al. Columbus, Ohio: Association for Computational Linguistics, June 2008, pp. 236–244.

[21] Neel Nanda and Joseph Bloom. *TransformerLens*. 2022.

[22] Chris Olah et al. "Zoom in: An introduction to circuits". In: *Distill* (2020). DOI: 10.23915/distill.00024.001.

[23] Kiho Park, Yo Joong Choe, and Victor Veitch. *The Linear Representation Hypothesis and the Geometry of Large Language Models*. arXiv:2311.03658 [cs, stat]. July 2024. DOI: 10.48550/arXiv.2311.03658.

[24] Alec Radford et al. "Language models are unsupervised multitask learners". In: ().

[25] Ian Tenney, Dipanjan Das, and Ellie Pavlick. "BERT rediscovers the classical NLP pipeline". In: *Proceedings of the 57th annual meeting of the association for computational linguistics*. 2019, pp. 4593–4601.

[26] Ian Tenney et al. "What do you learn from context? Probing for sentence structure in contextualized word representations". In: *International conference on learning representations*. 2019.

[27] Dmitry Vaintrob, Jake Mendel, and Kaarel Haenni. "Toward A Mathematical Framework for Computation in Superposition". en. In: (Jan. 2024).

[28] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

[29] Kevin Ro Wang et al. "Interpretability in the wild: a circuit for indirect object identification in GPT-2 small". In: *The eleventh international conference on learning representations*. 2023.

[30] Martin Wattenberg and Fernanda B Viégas. "Relational composition in neural networks: a survey and call to action". In: *arXiv preprint arXiv:2407.14662* (2024).

A Bundling and Binding of Concept Vectors

In this paper, we will consider real valued space $\mathbb{R}^n$ with $n \gg 100$. In such high dimensional spaces, we can have many more nearly orthogonal vectors than dimensions [9]. For an n-dimensional space, we cannot find more than n distinct vectors such that they are orthogonal to each other, i.e., the inner product (dot product) of any vector pair is exactly zero. However, we can find much larger sets of vectors such that the dot product of any vector to any other vector is nearly zero or small. One can think about a unit sphere in which the surface is divided into small patches such that the centroids of these areas are sufficiently far away, corresponding to a large enough angle (i.e., small enough dot product). We can imagine that there are many more such patches than the number of dimensions, depending on our dot product threshold.

Imagine we have three big vectors c_1, c_2, c_3, each representing a specific (binary) concept. If these vectors are nearly orthogonal, we can create a compositional concept vector $c_{1,2} = c_1 + c_2$ that has directional similarity to both vectors c_1 and c_2 but not to c_3. In vector symbolic architectures (VSA), this is referred to as bundling or superpositioning. We can test whether a vector c_j is similar to either c_1 or c_1 with a simple dot product and thresholding, i.e., $\langle c_{1,2}\,, c_j \rangle \geq \lambda$. This summation essentially represents an OR conjunction ('bag of concepts' vector). Please note the striking similarity of this test to a single perceptron with a ReLU activation function.

In addition to summation, we can also perform a binding operation by multiplying a suitable invertible matrix M with the concept vector. For instance, this allows us to model a specific order of concepts: $cm = Mc_1 + c_2$ is similar to c_2, Mc_1, and $M^{-1}c_1$, but not to c_1 (if $M \neq I$). M could be used here to model that c_1 refers to the first concept of the pair (c_1, c_2). We can see that if M is chosen well, the dot products are preserved: $\langle Mc_1\,, Mc_2 \rangle = (Mc_1)^\intercal (Mc_2) = c_1^\intercal M^\intercal M c_2$

If M is an orthogonal matrix, this corresponds to a basis transformation, preserving the length and dot products of vectors as it is just a rotation (or reflection) in simpler terms and $M^\intercal M$ is just the identity matrix. Even if the matrix is not orthogonal but composed of nearly orthogonal vectors, $M^\intercal M$ will lead to a nearly diagonal matrix. Looking at the inner product definition, we can see that if the scalars are roughly similar, the angle between vectors does not change much after the binding transformation. We will call such matrices nearly orthogonal matrices given that they preserve the dot product sufficiently well.

Such transformations can be interpreted in two ways. We can regard them as binding operators of (possibly previously bundled) concept vectors. We can also think of them as a construction of a new concept vector based on binary input features by adding those nearly orthogonal vectors (the rows of the matrix) where the corresponding input dimension is non-negative, with positive and negative

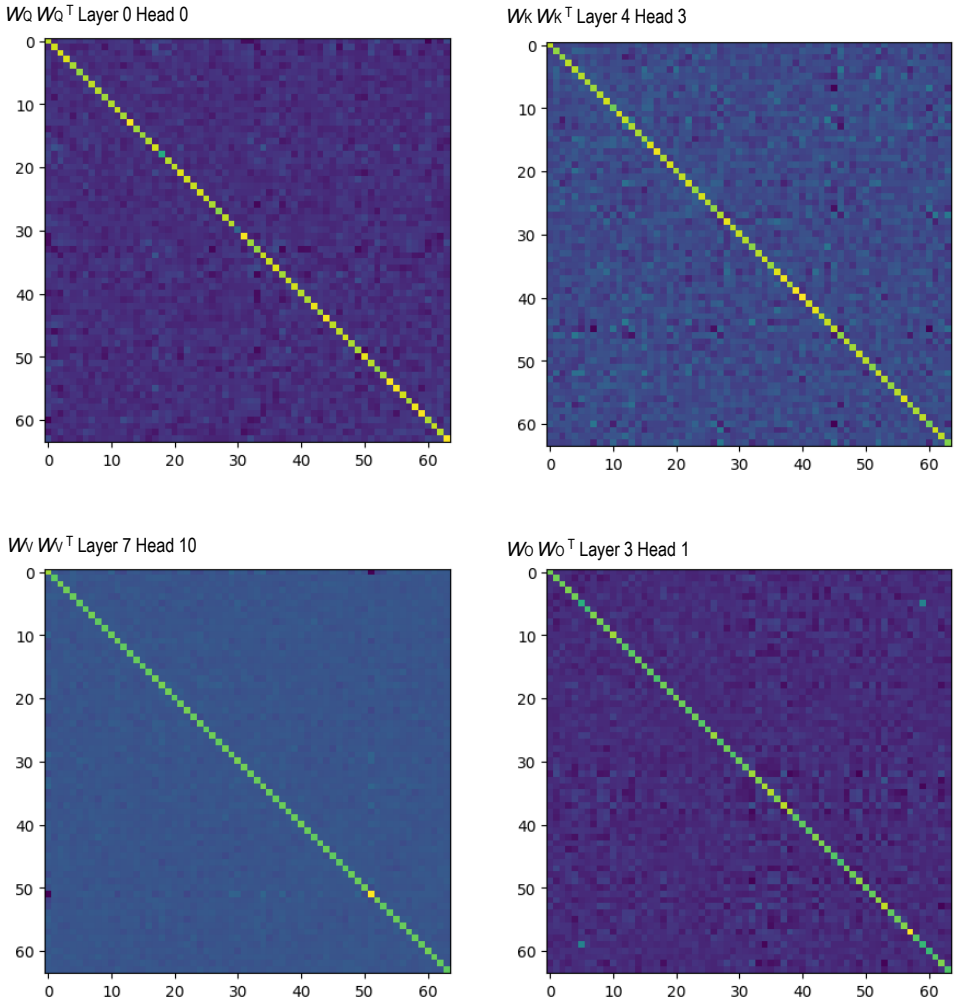

Figure 3: The attention matrices are composed of nearly orthogonal vectors. This is an exemplary selection of attention heads and their corresponding matrix multiplication $MM^\intercal$.

signs as indications of the (binary) direction of the concept. From a purely technical standpoint, both interpretations are valid. The latter interpretation might be more helpful if we can assign semantics to specific dimensions of the input vectors (monosemanticity).

We can build hierarchical concept vectors through bindings [30]. For instance, if we start with a set of concepts that could apply to people in general, we might first take two sums capturing the present concepts of two specific persons, respectively, which we can then bind with two different matrices M_1, M_2. If we want to compare whether the first person in two vectors are similar, we can unbind those vectors using the inverse of M_1 and then take the dot product. In other words, matrix binding can be interpreted as a packaging step, whereas the unbinding process can be used to 'focus' on a specific package. Bundling and binding of nearly orthogonal vectors are important principles of vector symbolic architectures [11], but similar ideas are used in many other techniques, e.g., in vector-based linear models [20].

B Figures

Near orthogonality of vectors and matrices are an important assumption of concept vectors and binding operations. We computed the matrix product $M^\intercal M$ for word embeddings, the attention matrices, and the output projections of the MLP blocks. In all cases, we observed strong diagonals with minimal artifacts, suggesting that these matrices are indeed composed of nearly orthogonal

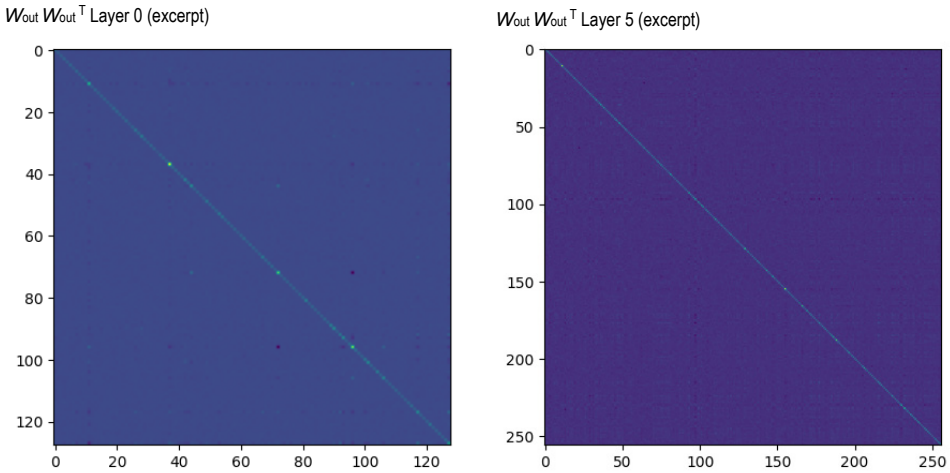

Figure 4: The output projection matrices of the MLP blocks are composed of nearly orthogonal vectors. This is an exemplary selection of output projection matrices for specific layers and the corresponding matrix multiplication $MM^{\intercal}$. We only show the top left cutout of the visualization as each layer contains 3072 neurons.

vectors. Figure 3 and Figure 4 show exemplary visualizations of a selection of attention heads and MLP layers.