Language Models Implement Simple Word2Vec-style Vector Arithmetic

Anonymous Author(s)
Affiliation
Address
email

Abstract

A primary criticism towards language models (LMs) is their inscrutability. This paper presents evidence that, despite their size and complexity, LMs sometimes exploit a computational mechanism familiar from traditional word embeddings: the use of simple vector arithmetic in order to encode abstract relations (e.g., *Poland:Warsaw::China:Beijing*). We investigate a range of language model sizes (from 124M parameters to 176B parameters) in an in-context learning setting, and find that for a variety of tasks (involving capital cities, upper-casing, and past-tensing), a key part of the mechanism reduces to a simple linear update applied by the feedforward networks. We further show that this mechanism is specific to tasks that require retrieval from pretraining memory, rather than retrieval from local context. Our results contribute to a growing body of work on the mechanistic interpretability of LLMs, and offer reason to be optimistic that, despite the massive and non-linear nature of the models, the strategies they ultimately use to solve tasks can sometimes reduce to familiar and even intuitive algorithms.

1 Intro

The growing capabilities of large language models (LLMs) have led to an equally growing interest in understanding how such models work under the hood. Such understanding is critical for ensuring that LLMs are reliable and trustworthy once deployed. Recent work (often now referred to as "mechanistic interpretability") has contributed to this understanding by reverse-engineering the data structures and algorithms that are implicitly encoded in the model's weights, e.g., by identifying detailed circuits [Wang et al., 2022, Elhage et al., 2021, Olsson et al., 2022] or by identifying mechanisms for factual storage and retrieval which support intervention and editing [Geva et al., 2021b, Li et al., 2022, Meng et al., 2022a,c, Dai et al., 2022].

Here, we contribute to this growing body of work by analyzing how LLMs recall information during in-context learning. Specifically, we observe that the mechanism that LLMs use in order to retrieve certain facts (e.g., mapping a country to its capital city) bears a striking resemblance to the type of vector arithmetic operations associated with LLMs' simpler, static word-embedding predecessors. That is, early word embeddings such as word2vec [Mikolov et al., 2013] famously supported factual recall via linear vector arithmetic–e.g., there existed some vector that, when added to the vector for any country would produce the vector for its capital. Modern LLMs are based on a complex transformer architecture [Vaswani et al., 2017] which produces contextualized word embeddings [Peters et al., 2018, Devlin et al., 2019] connected via multiple non-linearities. Despite this, we find that LLMs implement a very similar vector-addition mechanism which plays an important role in a number of in-context-learning tasks.

We study this phenomenon in three tasks–involving recalling capital cities, uppercasing tokens, and past-tensing verbs. Our key findings are:

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

- We find evidence of a **distinct processing signature** in the forward pass which characterizes this mechanism. That is, if models need to perform the `get_capital(x)` function, which takes an argument x and yields an answer y, they must first surface the argument x in earlier layers which enables them to apply the function and yield y as the final output (Figure 2). This signature generalizes across models and tasks, but appears to become sharper as models increase in size.
- We take a closer look at GPT2-Medium, and find that the vector arithmetic mechanism is implemented by mid-to-late layer feedforward networks (FFNs) in a way that is **modular and supports intervention**. That is, FFNs construct a vector that is not specific to context or argument, such that the same vector which produces *Warsaw* given *Poland* in one context can be dropped into an unrelated context to produce *Beijing* given *China*.
- We demonstrate that this mechanism is **specific to recalling information from pretraining memory**. For settings in which the correct answer can be retrieved from the prompt, this mechanism does not appear to play any role, and FFNs can be ablated entirely with relatively minimal performance degradation. Thus, we present new evidence supporting the claim that FFNs and attention specialize for different roles, with FFNs supporting factual recall and attention copying and pasting from local context.

Taken together, our results offer new insights about one component of the complex algorithms that underlie in-context learning. The simplicity of the mechanism, in itself surprising, raises the possibility that other apparently complicated behaviors may be supported by a sequence of simple operations under the hood. Moreover, our results suggest a distinct processing signature and hint at a method for intervention. These ideas could support future work on detecting and preventing unwanted behavior by LLMs at runtime.

2 Methods

In decoder-only transformer language models [Vaswani et al., 2017], a sentence is processed one word at a time, from left to right. The token at the current timestep is passed into the input of the model in order to predict the next, and so on. In this paper, we focus on the transformations that the next-token prediction undergoes in order to predict the next word. At each layer, an attention module and feed-forward network (FFN) module apply subsequent updates to this representation. Consider the FFN update at layer i, where x_i is the current next-token representation. The update applied by the FFN here is calculated as FFN$(\vec{x_i}) = \vec{o_i}, \;\; \vec{x_{i+1}} = \vec{x_i} + \vec{o_i}$ where $\vec{x_{i+1}}$ is the updated token for the next layer. Note that due to the *residual connection*, the output vector $\vec{o_i}$ is added to the input. $\vec{x}$ is updated this way by the attention and FFNs until the end of the model, where the token is decoded into the vocab space with the language modeling head E: softmax$(E\vec{x})$. From start to end, x is only updated by additive updates, and because of this, is said to form a *residual stream* [Elhage et al., 2021]. Thus, the token representation x_i represents all of the additions made into the residual stream up to layer i.

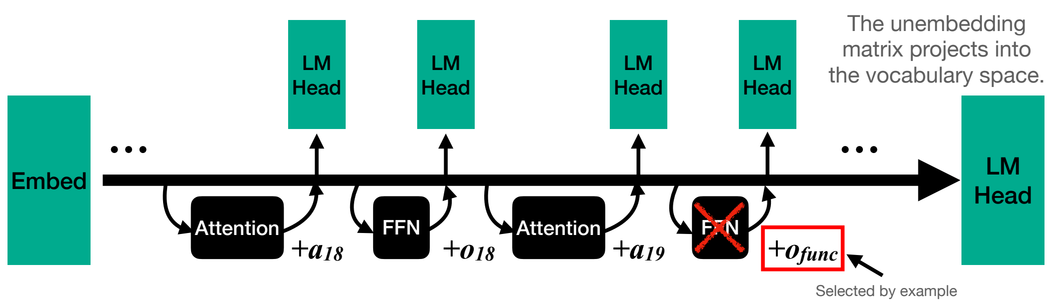

Figure 1: When decoding the next word, additive updates are made through the residual connections of each attention/FFN sub-layer. To decode the running prediction at every layer, the pre-trained language modeling head is applied at various points in each layer as in Geva et al. [2022a], nostalgebraist [2020]. The $\vec{o}$ vector interventions we make (§4.1) are illustrated by removing one or more FFN sub-layers, and replacing their updates with pre-defined vectors extracted from other examples.

2.1 Early Decoding

A key insight from the residual stream perspective is that we can decode the next token prediction with the LM head before it reaches the final layer. This effectively allows for "print statements" throughout the model's processing. The intuition behind this technique is that LMs incrementally update the token representation $\vec{x}$ to build and refine an encoding of the vocabulary distribution. This technique was initially introduced in nostalgebraist [2020] as the logit lens, and Geva et al. [2022b] show that LMs do in fact refine the output distribution over the course of the model. Figure 1 illustrates the process we use to decode hidden states into the vocabulary space, in which the hidden state at each layer is decoded with the pre-trained language modeling head E. After decoding into the vocabulary space, we apply a softmax to get a probability distribution over all tokens. When we decode at some layer, we say that the most likely token in the resulting vocab distribution is currently being represented in the residual stream. We examine several in-context learning tasks to understand how the answers to these problems are discovered by a model over the course of the forward pass.

2.2 Tasks

Can we understand the subprocesses underlying how LMs solve simple problems? We apply early decoding to suite of in-context learning tasks to explore the transformations the next token prediction undergoes in order to predict the answer.

World Capitals Our World Capitals task requires the model to retrieve the capital city for various states and countries in a few-shot setting. The dataset we use contains 248 countries and territories. A one-shot example is shown below:

"Q: What is the capital of France?
A: Paris
Q: What is the capital of Poland?
A:___" Expected Answer: " Warsaw"

Reasoning about Colored Objects We focus on a subset of 200 of the reasoning about colored objects dataset prompts (henceforth, the colored objects dataset) from BIG-Bench [Srivastava et al., 2022], which gives the model a list of colored common objects and require to simply state the color of a query object. For the purposes of this paper, we focus only on one aspect of this task–the model's ability to output the final answer in the correct format.[1]

"Q: On the floor, I see a silver keychain, a red pair of sunglasses, a gold sheet of paper, a black dog leash, and a blue cat toy. What color is the keychain?
A: Silver
Q: On the table, you see a brown sheet of paper, a red fidget spinner, a blue pair of sunglasses, a teal dog leash, and a gold cup. What color is the sheet of paper?
A:___" Expected answer: " Brown"

Past Tense Verb Mapping Lastly, we examine whether a language model can accurately recognize a pattern and predict the past tense form of a verb given its present tense. The dataset used is the combination of the regular and irregular partitions of the past tense linguistic mapping task in BIG-Bench [Srivastava et al., 2022]. After filtering verbs in which the present and past tense forms start with the same token, we have a total of 1,567 verbs. An example one-shot example is given below:

"Today I abandon. Yesterday I abandoned. Today I abolish. Yesterday I___" Expected answer: " abolished"

2.3 Models

We experiment exclusively on decoder-only transformer LMs across various sizes and pre-training corpora. When not specified, results in figures are from GPT2-medium. We also include results portraying the stages of processing signatures in the residual streams of the small, large, and extra large variants [Radford et al.], the 6B parameter GPT-J model [Wang and Komatsuzaki, 2021], and the 176B BLOOM model [Scao et al., 2022], either in the main paper or in the Appendix.

[1]The reason for this is that most of the results in this paper were originally observed as incidental findings while studying the Reasoning about Colored Objects task more generally. We thus zoom in on this one component for the purposes of the mechanism studied here, acknowledging that the full task involves many other steps that will no doubt involve other types of mechanisms.

3 Stages of Processing in Predicting the Next Token

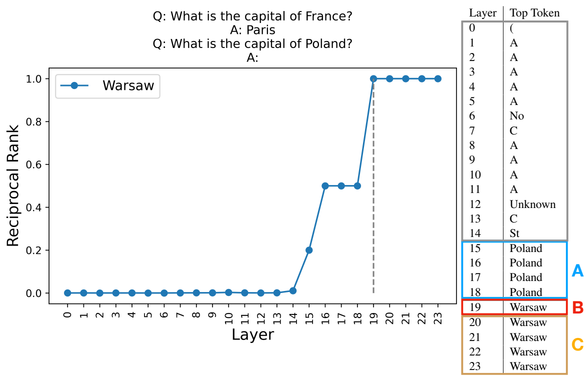

Figure 2: We can decode the running next-token prediction in an in-context learning task to reveal functionally distinct stages of processing. The blue box (A) shows where the model prepares an argument for transformation, the red box (B) shows the function application phase during which the argument is transformed (here with the `capital_of` function, and the yellow box (C) shows a saturation event, in which the model has found the answer, and stops updating the prediction.

First, we use the early decoding method in order to investigate how the processing proceeds over the course of a forward pass to the model. Each task requires the model to infer some relation to recall some fact, e.g., retrieving the capital of Poland. In these experiments, we see several discrete stages of processing that the next token undergoes before reaching the final answer. These states together provide evidence that the models "apply" the relevant functions (e.g., `get_capital`) abruptly at some mid-late layer to retrieve the answer. Moreover, in these cases, the model prepares the argument to this function in the layers prior to that in which the function is applied.

In Figure 2 we illustrate an example of the stages we observe across models. For the first several layers, we see no movement on the words of interest. Then, during **Argument Formation**, the model first represents the argument to the desired relation in the residual stream. This means that the top token in the vocabulary distribution at some intermediate layer(s) is the subject the question inquires about (e.g., the x, in `get_capital`(x)). During **Function Application** we find that the model abruptly switches from the argument to the output of the function (the y, in `get_capital`(x) $=$ y). We find that function application is typically applied by the FFN update at that layer to the residual stream. This is done by adding the output vector $\vec{o}$ of the FFN to the residual stream representation, thus transforming it with an additive update. We study these $\vec{o}$ vectors in detail in Section 4. Finally, the model enters **Saturation**[2], where the model recognizes it has solved the next token, and ceases updating the token representation for the remaining layers.

The trend can be characterized by an X-shaped pattern of the argument and final output tokens when plotting the ranks of the argument(x) and output (y) tokens. We refer to this behavior as argument-function processing. Figure 3 shows that this same processing signature can be observed consistently across tasks and models. Moreover, it appears to become more prominent as the models increase in size. Interestingly, despite large differences in number of layers and overall size, models tend to undergo this process at similar points proportionally in the model.

4 Implementation of Context-Independent Functions in FFN Updates

The above results on processing signature suggest that the models “apply” a function about 2/3rds of the way through the network with the addition of an FFN update. Here, we investigate the mechanism via which that function is applied more closely. Specifically, focusing on GPT2-Medium[3], we show that we can force the encoded function to be applied to new arguments in new contexts by isolating the responsible FFN output vector and then dropping into a forward pass on a new input.

[2] Saturation events are described in Geva et al. [2022a] where detection of such events is used to “early-exit” out of the forward pass

[3] We focus on one model because manual analysis was required in order to determine how to perform the intervention. See Appendix for results on GPT-J and Section 7 for discussion.

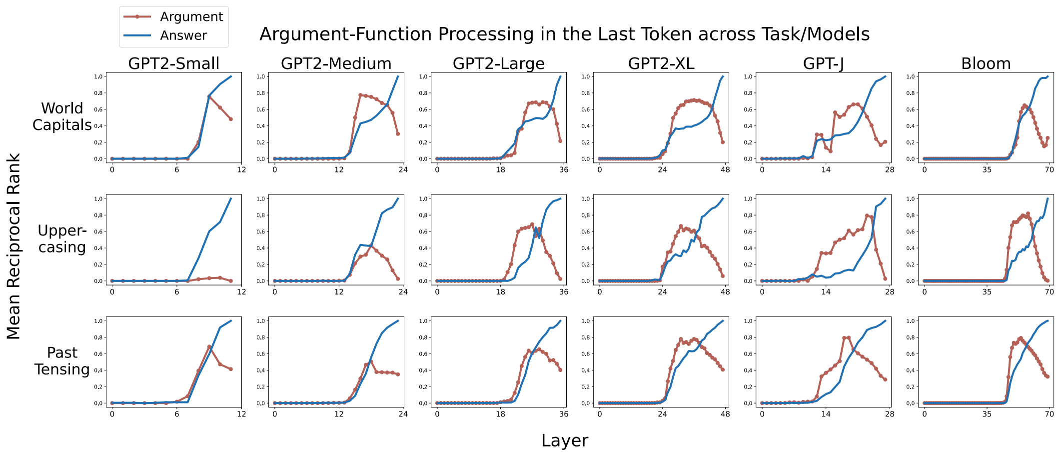

Figure 3: Argument formation and function application is characterized by a promotion of the argument (red) followed by it being replaced with the answer token (blue), forming an X when plotting reciprocal ranks. Across the three tasks we evaluate, we see that most of the models exhibit these traces, and despite the major differences in model depths, the stages occur at similar points in the models. Data shown is filtered by examples in which the models got the correct answer.

4.1 $\vec{o}$ Vector Interventions

Consider the example in Figure 2. At layer 18, the residual stream ($\vec{x_{18}}$) is in argument formation, and represents the " Poland" token. At the end of layer 19, a function is applied, transforming $\vec{x_{19}}$ into the answer token " Warsaw. As discussed in the previous section, we can isolate the function application in this case to FFN 19; let $\tilde{x_{19}}$ represent the residual stream after the attention update, but before the FFN update at layer 19 (which still represents Poland). Recall that the update made by FFN 19 is written FFN$_{19}(\tilde{x_{19}}) = \vec{o_{19}}$ and $\vec{x_{19}} = \tilde{x_{19}} + \vec{o_{19}}$. We find that $\vec{o_{19}}$ will apply the `get_capital` function regardless of the content of $\tilde{x_{19}}$. For example, if we add $\vec{o_{19}}$ to some $\tilde{x}$ which represents the " China" token, it will transform into " Beijing". Thus we refer to $\vec{o_{19}}$ as $\vec{o_{city}}$ since it retrieves the capital cities of locations stored in the residual stream. We locate such $\vec{o}$ vectors in the uppercasing and past tense mapping tasks in the examples given in Section 2.2, which we refer to as $\vec{o_{upper}}$ and $\vec{o_{past}}$, respectively.[4]

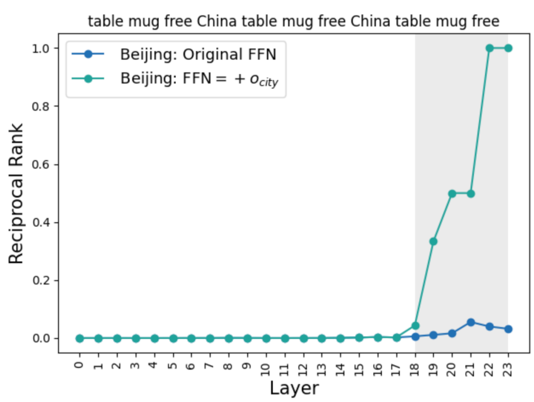

Figure 4: The gray area indicates layers where FFN intervention was performed. We find that even if the input context is nonsense (repeating pattern of "table mug free China"), if we can use "China" as an argument in the residual stream, the $\vec{o_{city}}$ vector has the effect of promoting the correct capital city.

We test whether these updates have the same effect, and thus implement the same function, as they do in the original contexts from which they were extracted, which would imply a systematic structure in the internal embedding space the LM leverages. To do so, we replace entire FFN layers with these vectors and run new inputs through the intervened model.[5]

[4] In Appendix A, we extend these results to GPT-J, for which the same procedure leads to strong effects on uppercasing, but smaller overall positive effects on capital cities and past tensing (see Section 7).

[5] Which FFNs to replace is a hyperparameter; we find that replacing layers 18-23 in GPT2-Medium leads to good results. It also appears necessary to replace multiple FFNs at a time. See additional experiments in Appendix D. In summary, it is likely that the $\vec{o}$ vectors are added over the course of several layers, consistent

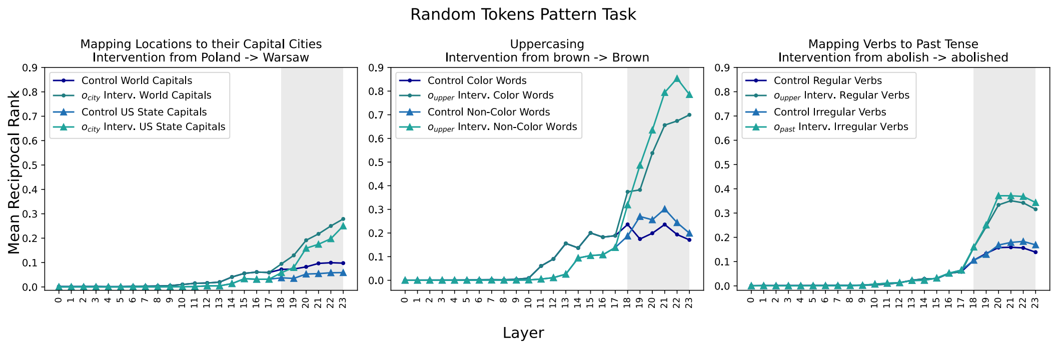

Figure 5: We intervene on GPT2-Medium's forward pass while it is predicting the completion of a pattern. The control indicates normal model execution, while the gray boxes indicate which FFNs are replaced with our selected $\vec{o}$ vectors. We can see a significant increase in the reciprocal rank of the output of the function implemented by the $\vec{o}$ vector used even though the context is completely absent of any indication of the original task.

Data: We are interested in whether the captured o vectors can be applied in a novel context, in particular, to a context that is otherwise devoid of cues as to the function of interest. Thus, we synthesize a new dataset where each entry is a string of three random tokens (with leading spaces) followed by a token x which represents a potential argument to the function of interest. For example, in experiments involving o_{city}, we might include a sequence such as `table mug free China table mug free China table mug free`. This input primes the model to produce "China" at the top of the residual stream, but provides no cues that the capital city is relevant, and thus allows us to isolate the effect of o_{city} in promoting "Beijing" in the residual stream. In addition to the original categories, we also include an "out-of-domain" dataset for each task: US states and capitals, 100 non-color words, and 128 irregular verbs. These additional data test the sensitivity of the $\vec{o}$ vectors to different types of arguments.

Results: Figure 4 shows results for a single example. Here, we see that "Beijing" is promoted all the way to the top of the distribution solely due to the injection of $\vec{o_{city}}$ into the forward pass. Figure 5 shows that this pattern holds in aggregate. In all settings, we see that the outputs of the intended functions are strongly promoted by adding the corresponding $\vec{o}$ vectors. By the last layer, for world and state capitals, the mean reciprocal rank of the target city name across all examples improves from roughly the 10th to the 4th-highest ranked word and 20th and 3rd-ranked words respectively.

We also see the promotion of the proper past tense verbs by $\vec{o_{past}}$. The reciprocal ranks improve similarly for both regular (approx. 7th to 3rd rank) and irregular verbs (approx. 6th to 3rd), indicating that the relationship between tenses is encoded similarly by the model for these two types. $\vec{o_{upper}}$ promotes the capitalized version of the test token almost every time, although the target word starts at a higher rank (on average, rank 5). These results together show that regardless of the surrounding context, and regardless of the argument to which it is applied, $\vec{o}$ vectors consistently apply the expected functions. Since each vector was originally extracted from the model's processing of a single naturalistic input, this generalizability suggests **significant structure and cross-context abstraction within the learned embedding-space**.

Common Errors: While the above trend clearly holds on the aggregate, the intervention is not perfect for individual cases. The most common error is that the intervention has no real effect. In the in-domain (out-domain) settings, this occurred in about 37% (20%) of capital cities, 4% (5%) on uppercasing, and 19% (22%) for past tensing. We believe the rate is so much higher for world capitals because the model did not have a strong association between certain country-capital pairs from pretraining, e.g, for less frequently mentioned countries. Typically, in these cases, the top token remains the argument, but sometimes becomes some random other city, for example, predicting the capital of Armenia is Vienna. We also find that the way tokenization splits the argument and target

with the idea that residual connections encourage each layer to move gradually towards a point of lower loss [Jastrzebski et al., 2017].

words affects the ability of the $\vec{o}$ vector to work and is another source of errors. This is discussed further in Appendix E.

5 The Role of FFNs in Out-of-Context Retrieval

So far, we have shown that FFN output vectors can encode functions that transfer across contexts. Here we investigate whether the mechanism we identify applies in general to associations of this type, or rather if such functionality can be implemented by the attention mechanism instead. Shared among the tasks we study is the requirement to recall a token that does not appear in the given context (abstractive tasks). In this section we show that mid-higher layer FFNs are crucial for this process. When the answer to the question *does* appear in context (extractive tasks), we find that ablating a subset of FFNs has a comparatively minor effect on performance, indicating that they are relatively modular and there is a learned division of labor within the model. This observation holds across the decoder-only LMs tested in this paper, but is particularly salient in the larger/deeper networks. This breakdown is consistent with previous work finding that FFNs store facts learned from pre-training [Geva et al., 2021a, Meng et al., 2022b,c] and attention heads copy from the previous context [Wang et al., Olsson et al., 2022].

5.1 Abstractive vs. Extractive Tasks

Extractive Tasks: Extractive tasks are those in which the exact tokens required to answer a prompt can be found in the input context. These tasks can thus be solved by parsing the local context alone, and thus do not necessarily require the model to apply a function of the type we have focused on in this paper (e.g., a function like `get_capital`).

Abstractive Tasks: Are those in which the answer to a prompt is not given in the input context and must be retrieved from pretraining memory. Our results suggest this is done primarily through argument-function processing, requiring function application through (typically) FFN updates as described in Section 3.

We provide examples with their associated GPT2-Medium layerwise decodings in Figure 6. We expect that the argument formation and function application stages of processing occur primarily in abstractive tasks. Indeed, in Appendix A, we show that the characteristic argument-answer X pattern disappears on extractive inputs. We hypothesize that applying out-of-context transformations to the predicted token representation is one of the primary functions of FFNs in the mid-to-late layers, and that removing them should only have a major effect on tasks that require out-of-context retrieval.

	Top Tokens per Layer	
	Abstractive Task	Extractive Task
Layer	Q: What is the capital of Somalia? A: Mogadishu Q: What is the capital of Poland? A:	The capital of Somalia is Mogadishu. The capital of Poland is Warsaw. Q: What is the capital of Somalia? A: Mogadishu Q: What is the capital of Poland? A:
...	...	...
14	St	St
15	Poland	St
16	Poland	Warsaw
17	Poland	Warsaw
18	Poland	Warsaw
19	Warsaw	Warsaw
20	Warsaw	Warsaw
21	Warsaw	Warsaw
22	Warsaw	Warsaw
23	Warsaw	Warsaw

Figure 6: The abstractive task undergoes argument formation (blue) and function application (red), while the extractive task immediately saturates (yellow).

5.2 Effect of Ablating FFNs

Data: Consider the example shown in Section 2.2 demonstrating the $o_{\vec{upper}}$ function. By providing the answer to the in-context example as " Silver", we make the task abstractive by requiring the in-context token " brown" to be transformed to " Brown" in the test example. However, if we provide the in-context label as " silver", the task becomes extractive, as the expected answer becomes " brown". We create an extractive version of this dataset by lowercasing the example answer. All data is presented to the model with a single example (one-shot). Notice that the abstractive and extractive examples only differ by a single character and are thus minimally different.

We repeat this experiment on the world capitals task by adding the prefix "The capital of A is B. The capital of C is D" to each input. Notice, however, that since the answer is provided explicitly, the task is much easier for the models in the extractive case.

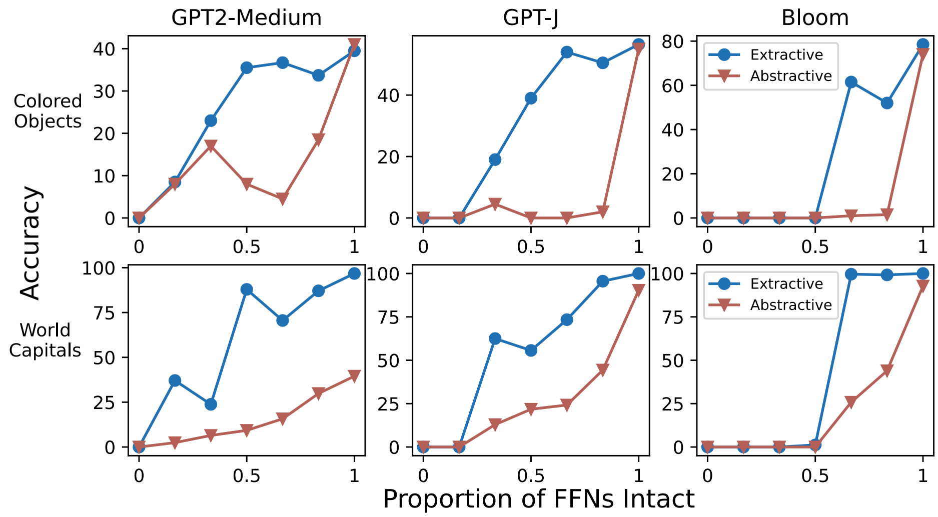

Figure 7: Removing FFNs negatively affects performance when the task is abstractive: the in-context label is an out-of-context transformation of the in-context prompt (e.g., " silver" in context, answer given as " Silver"). In comparison, on the extractive dataset, performance is robust to a large proportion of FFNs being removed. Other models tested are shown in Appendix B

Procedure: We run the one-shot extractive and abstractive datasets on the full models, and then repeatedly remove an additional 1/6th of all FFNs from the top down (e.g., in 24 layer GPT2-Medium: removing the 20-24th FFNs, then the 15-24th, etc.).

Results: Our results are shown in Figure 7. Despite the fact that the inputs in the abstractive and extractive datasets only slightly differ (by a single character in the colored objects case) we find that performance plummets on the abstractive task as FFNs are ablated, while accuracy on the extractive task drops much more slowly. For example, even after 24 FFN sublayers are removed from Bloom (totaling 39B parameters) extractive task accuracy for the colored objects dataset drops 17% from the full model's performance, while abstractive accuracy drops 73% (down to 1% accuracy). The case is similar across model sizes and pretraining corpora; we include results on additional models in Appendix B. This indicates that we can isolate the effect of locating and retrieving out of context tokens in this setting to the FFNs. Additionally, because the model retains reasonably strong performance compared to using the full model, we do not find convincing evidence that the later layer FFNs are contributing to the extractive task performance, supporting the idea of modularity within the network.

6 Related Work

Recent work has contributed to understanding language models by studying the role of different modules in the transformer architecture in language modeling. In particular, the attention layers [Olsson et al., 2022, Kobayashi et al., 2020, Wang et al.] and more notably for this work, the FFN modules, which are frequently associated with factual recall and knowledge storage [Geva et al., 2021a, Meng et al., 2022a,c]. Although how language models store and use knowledge has been studied more generally as well [Petroni et al., 2019, Cao et al., 2021, Dai et al., 2022, Bouraoui et al., 2019, Burns et al., 2022, Dalvi et al., 2022, Da et al., 2021] as well as in static embeddings [Dufter et al., 2021]. Recent work in mechanistic interpretability aims to fully reverse engineer how LMs perform some behaviors. Our work builds on the finding that FFN layers promote concepts in the vocabulary space [Geva et al., 2022a] by breaking down the process the model uses to do this in context. Bansal et al. [2022] perform ablation studies to test the importance of attention and FFN layers on in-context learning tasks, here we offer an explanation for their role in some cases. Other work analyze information flow within an LM to study how representations are built through the layers [Voita et al., 2019, Tenney et al., 2019] and show distinct points of processing in the model. We also follow this approach, but our analysis focuses on interpreting how models use individual updates within the forward pass, rather than probing for what information is encoded and potentially used to make predictions. Ilharco et al. [2023] show that vector arithmetic can be performed with the weights of finetuned models to compose tasks, similar to how $\vec{o}$ vectors can induce functions in the activation space of the model.

7 Discussion

In this work, we describe a mechanism that is partially responsible for LMs ability to recall factual associations. We conceptualize these recalls as the application of some function (e.g., `get_capital(`x`) = `y and find that the next-token prediction goes through several discrete stages of processing in which the prediction first represents the argument x (e.g., Poland) before applying that function with an additive update to get the final answer y (Warsaw). A core challenge in interpreting neural networks is determining whether the information attributed to certain model components is actually used for that purpose during inference [Hase and Bansal, 2022, Leavitt and Morcos, 2020]. While previous work has implicated FFNs in recalling factual associations [Geva et al., 2022a, Meng et al., 2022a], we show through intervention experiments that we can manipulate the information flowing through the model during these stages. Specifically, we show that it is possible to capture the output vector of an FFN from a single forward pass on a single in-context learning example, and that the captured vector can be used to apply the same function to new arguments (e.g., other countries) in totally different contexts. This process provides a surprisingly simple explanation for the internal subprocesses used by LMs to recall factual associations and resembles vector arithmetic observed in static word embeddings. Our findings invite future work aimed at understanding why, and under what conditions, LMs learn to use this mechanism when they are capable of solving such tasks using, e.g., adhoc memorization.

A limitation that we observe is that the process for carrying out the $\vec{o}$ intervention depends on hyperparameters which are often model-specific (i.e., the exact stimuli used to extract the intervention, and the layer(s) at which to perform the intervention). We provide our most detailed investigation on GPT2-Medium, which clearly illustrates the phenomenon. Our experiments on stages of processing with GPT-J suggest that the same phenomena is in play, although (as discussed in Section 4 and Appendix A), the procedures we derive for interventions on GPT2-Medium do not transfer perfectly. Specifically, we can strongly reproduce the intervention results on uppercasing for GPT-J; results on the other two tasks are positive but with overall weaker effects. This requirement of model-specific customization is common in similar mechanistic interpretability work, e.g., [Meng et al., 2022a, Wang et al., 2022, Geva et al., 2022b], and a prioritiy in future work must be to identify common patterns across these individual studies which reduce the need to repeat such effort on each new model. That said, in this work and other similar efforts, a single positive example as a proof of concept is often sufficient to advance understanding and spur future work that improves robustness across models.

In the long term, findings like those presented here have implications for improving the trustworthiness of LMs in production. If we can understand how models break down complex problems into simple and predictable subprocesses, we can help more readily audit their behavior. Interpreting the processing signatures of model behaviors might offer an avenue via which to audit and intervene at runtime in order to prevent unwanted behavior. Moreover, understanding which relations FFNs encode could aid work in fact location and editing. Contemporaneous work [Geva et al., 2023] has studied a different mechanism for factual recall in LMs, but it is unclear how and when these mechanisms interact.

8 Conclusion

We contribute to a growing body of work on interpreting how the internal processes of language models (LMs) produce some behavior. On three in-context learning tasks, we observe that the next-token prediction appears to undergo several stages of processing in which LMs represent arguments to functions in their residual streams. This process occurs in models ranging in size from 124M to 176B parameters. On GPT2, We study instances where the additive update is made by the output vectors ($\vec{o}$ vectors) of feed-forward networks (FFNs). We show that for all tasks we test, $\vec{o}$ vectors calculated by the model in the process of solving some task can be extracted and replace the FFN updates of the model to solve novel instances of that task, providing evidence that LMs can learn *self-contained* and context-independent functions from pretraining.

References

Hritik Bansal, Karthik Gopalakrishnan, Saket Dingliwal, Sravan Bodapati, Katrin Kirchhoff, and Dan Roth. Rethinking the Role of Scale for In-Context Learning: An Interpretability-based Case Study at 66 Billion Scale, December 2022. URL http://arxiv.org/abs/2212.09095. arXiv:2212.09095 [cs].

Zied Bouraoui, Jose Camacho-Collados, and Steven Schockaert. Inducing Relational Knowledge from BERT, November 2019. URL https://arxiv.org/abs/1911.12753v1.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering Latent Knowledge in Language Models Without Supervision, December 2022. URL http://arxiv.org/abs/2212.03827. arXiv:2212.03827 [cs].

Boxi Cao, Hongyu Lin, Xianpei Han, Le Sun, Lingyong Yan, Meng Liao, Tong Xue, and Jin Xu. Knowledgeable or Educated Guess? Revisiting Language Models as Knowledge Bases. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1860–1874, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.146. URL https://aclanthology.org/2021.acl-long.146.

Jeff Da, Ronan Le Bras, Ximing Lu, Yejin Choi, and Antoine Bosselut. Analyzing Commonsense Emergence in Few-shot Knowledge Models. September 2021. URL https://openreview.net/forum?id=StHCELh9PVE.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, 2022.

Fahim Dalvi, Abdul Rafae Khan, Firoj Alam, Nadir Durrani, Jia Xu, and Hassan Sajjad. Discovering Latent Concepts Learned in BERT, May 2022. URL http://arxiv.org/abs/2205.07237. arXiv:2205.07237 [cs].

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Philipp Dufter, Nora Kassner, and Hinrich Schütze. Static Embeddings as Efficient Knowledge Bases? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2353–2363, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.186. URL https://aclanthology.org/2021.naacl-main.186.

N Elhage, N Nanda, C Olsson, T Henighan, N Joseph, B Mann, A Askell, Y Bai, A Chen, T Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, 2021a.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer Feed-Forward Layers Are Key-Value Memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic, November 2021b. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.446. URL https://aclanthology.org/2021.emnlp-main.446.

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *arXiv preprint arXiv:2203.14680*, 2022a.

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. Transformer Feed-Forward Layers Build Predictions by Promoting Concepts in the Vocabulary Space, October 2022b. URL http://arxiv.org/abs/2203.14680. arXiv:2203.14680 [cs].

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models, 2023.

Peter Hase and Mohit Bansal. When can models learn from explanations? a formal framework for understanding the roles of explanation data. In *Proceedings of the First Workshop on Learning with Natural Language Supervision*, pages 29–39, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.lnls-1.4. URL https://aclanthology.org/2022.lnls-1.4.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *ICLR*, 2023.

Stanisław Jastrzebski, Devansh Arpit, Nicolas Ballas, Vikas Verma, Tong Che, and Yoshua Bengio. Residual connections encourage iterative inference. In *International Conference on Learning Representations*, 2017.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. Attention is not only a weight: Analyzing transformers with vector norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7057–7075, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.574. URL https://aclanthology.org/2020.emnlp-main.574.

Matthew L Leavitt and Ari Morcos. Towards falsifiable interpretability research. *arXiv preprint arXiv:2010.12016*, 2020.

Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. *arXiv preprint arXiv:2210.13382*, 2022.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and Editing Factual Associations in GPT, October 2022a. URL http://arxiv.org/abs/2202.05262. arXiv:2202.05262 [cs] version: 4.

Kevin Meng, David Bau, Alex J Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. In *Advances in Neural Information Processing Systems*, 2022b.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass editing memory in a transformer. *arXiv preprint arXiv:2210.07229*, 2022c.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

nostalgebraist. interpreting GPT: the logit lens. 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://aclanthology.org/N18-1202.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language Models as Knowledge Bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1250. URL https://aclanthology.org/D19-1250.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

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*, pages 4593–4601, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Elena Voita, Rico Sennrich, and Ivan Titov. The Bottom-up Evolution of Representations in the Transformer: A Study with Machine Translation and Language Modeling Objectives. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4396–4406, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1448. URL https://aclanthology.org/D19-1448.

Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the Wild: a Circuit for Indirect Object Identification in GPT-2 small, November 2022. arXiv:2211.00593 [cs].

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. In *NeurIPS ML Safety Workshop*.