Vector Arithmetic in Concept and Token Subspaces

Sheridan Feucht **Byron Wallace** **David Bau***
Northeastern University
{feucht.s,b.wallace,d.bau}@northeastern.edu

Abstract

In order to predict the next token, LLMs must represent semantic and surface-level information about the current word. Previous work identified two types of attention heads that disentangle this information: (i) Concept induction heads, which copy word meanings, and (ii) Token induction heads, which copy literal token representations [2]. We show that these heads can be used to identify subspaces of model activations that exhibit coherent semantic structure in Llama-2-7b. Specifically, when we transform hidden states using the attention weights of concept heads, we are able to more accurately perform parallelogram arithmetic [4, 5] on the resulting hidden states, e.g., showing that *Athens* – *Greece* + *China* = *Beijing*. This transformation allows for much higher nearest-neighbor accuracy (80%) than direct use of raw hidden states (47%). Analogously, we show that token heads allow for transformations that reveal surface-level word information in hidden states, allowing for operations like *coding* – *code* + *dance* = *dancing*.

1 Introduction

Consider how an LLM might model the word *boat*. A boat is a type of vehicle that floats on water, can be powered by sails or engines, and generally carries one or more people. But there are many other important facts about this word: It is an English word that is all lowercase; it starts with the letter 'b'; it rhymes with (and looks like) *coat*, and it is a singular common noun referring to a broad category. If we wish to analyze the relationship between the word *boat* and the word *water*, we must focus on the semantics of these words, discarding all of the other information that an LLM might encode. On the other hand, if we are trying to find a word that rhymes with *boat*, its meaning may not be particularly helpful to know.

In the original `word2vec` paper, Mikolov et al. [4, 5] embed words in a manner that allows for parallelogram-like vector arithmetic: they claim that their embedding space is structured such that *man* is to *woman* as *king* is to *queen*. However, we find that their approach is only somewhat effective for raw Llama-2-7b hidden states [7] (Section 3). We hypothesize that these apparently poor results observed using a naive approach may be attributed to "interference" from irrelevant information in model activations. In other words, we posit that much of the information packed into LLM hidden states has nothing to do with semantics, and that `word2vec` arithmetic is only effective if performed in a semantic subspace of model activations.

By using the weights of concept induction heads from [2], we isolate a lower-dimensional space of Llama-2-7b activations for which, e.g., the representation of `king` - `man` + `woman` ≈ `queen`. We also find that we can use token induction heads to perform parallelogram arithmetic for surface-level tasks, like identifying the first letter in a word, with much higher accuracy than using raw hidden states. This suggests that concept and token induction heads from [2] exhibit rich structure in their outputs, operating in subspaces of model activations that represent different facets of words.

*Paper website at `https://arithmetic.baulab.info`.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Mechanistic Interpretability.

2 Method: Concept and Token Lens

In recent work, Feucht et al. [2] identify two types of attention heads responsible for copying text in-context: token induction heads (first described in [1]), which copy exact tokens, and concept induction heads, which copy "fuzzy" word meanings. As these attention heads are responsible for copying previous words seen in-context, their value and output weights can be naturally viewed as transformations that extract semantic and token-level information from any given hidden state. Feucht et al. [2] use this insight to develop a "concept lens" that visualizes semantic information in hidden states. We repurpose their approach to derive general concept and token transformations that reveal meaningful structure in hidden states, in the sense that arithmetic in the resultant space accords with intuitive analogies.

Let d be Llama-2-7b's hidden dimension and m the dimension of a single head. We rely on a key insight from Elhage et al. [1]: the value and output projections for a particular head h at layer l, $V_{(l,h)} \in \mathbb{R}^{(m,d)}$ and $O_{(l,h)} \in \mathbb{R}^{(d,m)}$ respectively, are solely responsible for whatever information a head writes into the residual stream. Specifically, they point out that the product of these two matrices $O_{(l,h)}V_{(l,h)}$ is a low-rank $d \times d$ matrix (at most rank m) that determines the effect of head (l, h) on the residual stream. In other words, multiplying a hidden state x_l by this matrix extracts whatever information within x_l that this head typically contributes to the residual stream.

As described in [2], to build a *concept lens* $L_{C_k} \in \mathbb{R}^{(d,d)}$ that reads from all of the concept induction head subspaces simultaneously, we combine the weights from the top-k concept induction heads C_k. We calculate the sum of the top-k concept OV matrices:

$$L_{C_k} = \sum_{(l,h) \in C_k} O_{(l,h)} V_{(l,h)}. \tag{1}$$

If all attention heads in C_k are in the same layer, $L_{C_k} x_l$ is mathematically equivalent to taking the sum of the outputs of those attention heads. However, we also allow for summation of heads across layers, which was empirically effective in prior work [6], possibly because transformer representations are interchangeable in intermediate layers [3]. We can repeat this process using the top-k token induction heads T_k to obtain a *token lens*, which reveals information about the written form of a word.

3 Parallelogram Arithmetic

3.1 Approach

We test the assertion made by Mikolov et al. [4] that embeddings should exhibit parallelogram-like structure: in other words, we test whether `man` – `woman` = `king` – `queen`. Figures 1a and 1b illustrate our approach. We use data from Mikolov et al. [4] and Todd et al. [6], which consists of tuples of words in some relation to each other. For every possible pair of tuples, we want to evaluate whether the difference between one tuple is equal to the difference between another; i.e., for (`Athens`, `Greece`) and (`Beijing`, `China`), we want to evaluate whether `Athens` – `Greece` = `Beijing` – `China`. In general, we notate this as $a - b = a' - b'$ for a pair of tuples (a, b) and (a', b').

To obtain embeddings for each word w, we first pass that word through the model in a clean run, obtaining a single vector w_ℓ by taking its last token representation at a particular layer ℓ. We then transform this hidden state using some $d \times d$ matrix L to obtain Lw_ℓ. In the **raw** setting, we do not transform w_ℓ at all, so $L = I_d$. In the **concept** setting, we use $L = L_{C_k}$, in the **token** setting, we use $L = L_{T_k}$, and as a baseline, we use $L = L_{all}$, which is the sum of **all** attention head OV matrices. Finally, to see whether $a_\ell - b_\ell = a'_\ell - b'_\ell$ in the subspace mapped to by L, we calculate $La_\ell - Lb_\ell + Lb'_\ell$ and evaluate if La'_ℓ is its nearest neighbor among all possible words in this task.

Passing a word to a model on its own can be ambiguous, so we choose a prefix for each task that can be prepended to all words in that task (Table 1). This sequence is constant across all words for which we perform vector arithmetic. See Appendix A for results for all tasks with and without prefixes.

3.2 Results

Figure 1 shows nearest-neighbor accuracy for selected tasks. While all settings achieve accuracies above random chance (represented by the dotted gray line), concept and token lenses allow for much

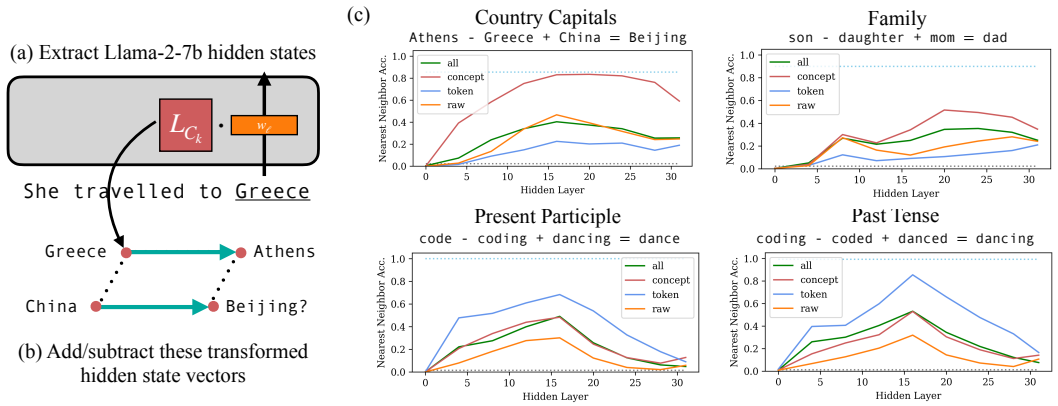

Figure 1: `word2vec`-style vector arithmetic is more accurate when working in subspaces from [2] instead of using raw hidden states. (a) To extract embeddings for a word, we prefix with a constant phrase (e.g. "She travelled to") and save the last token representation of the word at a chosen layer ℓ. To extract conceptual or token information from this vector, we multiply by concept and token lenses L_{C_k} and L_{T_k} respectively (Section 2). (b) Using a vector from a separate context to represent each word, we measure whether `Athens` – `Greece` + `China` has `Beijing` as its top nearest neighbor. (c) For semantic tasks like capital cities and gender-based family words, doing vector arithmetic in the subspace of the top-k concept heads (red) is more effective than using raw hidden states (orange), the top-k token heads (blue), or the sum of all attention head OV matrices (green). On the other hand, the subspace read by the top-k token heads is most effective for grammatical tasks that involve changing the spelling of a word (e.g., `code` $\rightarrow$ `coding`). For comparison, dotted gray lines represent random chance, whereas dotted light blue represents Llama-2-7b's 5-shot ICL accuracy for this task. We use $k = 80$, as found in [2].

more accurate vector arithmetic. In the case of capital cities, this arithmetic is just as good as the model's accuracy when asked to complete the task in an ICL setting with 5 shots (light blue dotted line). Oddly, this approach is less effective for tasks that seem simpler, like present participles of verbs. Errors in these cases are difficult to interpret, as the incorrect nearest neighbor is often one of the operands in the original expression.

Figure 2 shows results for 14 tasks from the original `word2vec` paper [4]. Concept lens is more effective for semantic tasks, whereas token lens does well for tasks that contain surface-level word variations (e.g., `quick` $\rightarrow$ `quickly`). Pluralizing nouns ("gram8-plural") can be done in both concept space and in token space (by adding 's' to a word), but pluralizing verbs can only be done in token space ("gram9-plural-verbs"), possibly because the latter mostly has to do with verb agreement, not word meaning. See Appendix A for more tasks from Todd et al. [6].

3.3 Effective Rank of Concept and Token Subspaces

Although the OV matrix for a single attention head is at most rank m with $m < d$, our transformations L_{C_k} and L_{T_k} are full-rank when $k = 80$, as shown empirically in Figure 3a. This means that our transformations for Figure 1 do not actually project activations onto a strict concept or token subspace. However, we hypothesize that we do not need to use all d dimensions to perform vector arithmetic for these tasks. To test this, we set all singular values below the top-r values to zero for L_{C_k}, L_{T_k}, and L_{all}, sweeping across values of r (Figure 3b). We choose the best layer for each task from Figure 1 and analyze whether reducing the rank of L damages performance. As Figure 3c shows, reducing the rank of L does not damage performance for tasks from Section 3, indicating that these transformations, in effect, project activations onto a lower-dimensional subspace.

4 Conclusion

We combine attention weights from previously-discovered components to obtain low-rank transformations that reveal token and concept information in Llama-2-7b, suggesting that understanding the geometry of LLM activations requires a precise formulation of *what information* we want to analyze.

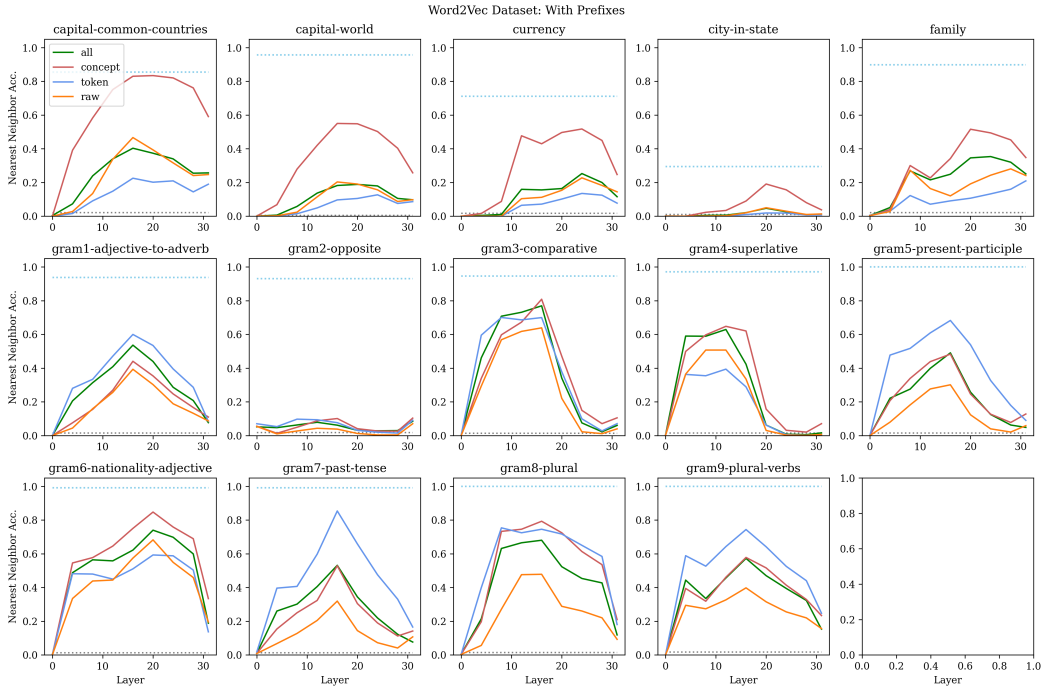

Figure 2: Nearest-neighbor accuracy for all `word2vec` tasks [4] with prefixes for each task in Table 1 (Llama-2-7b). Dotted gray lines indicate guessing accuracy (out of all possible neighbors/words in the dataset). Dotted light blue lines indicate 5-shot ICL accuracy for this task, i.e., the best possible performance this model can have for this task. We do not expect high performance for the "opposite" task due to its cyclic nature: to represent the concept of "opposite," we need `possible` – `impossible` = `impossible` – `possible`, which is incompatible with parallelogram arithmetic. Targeted subspaces are more effective than using all attention heads for most tasks, except for gram1, gram3, and gram4.

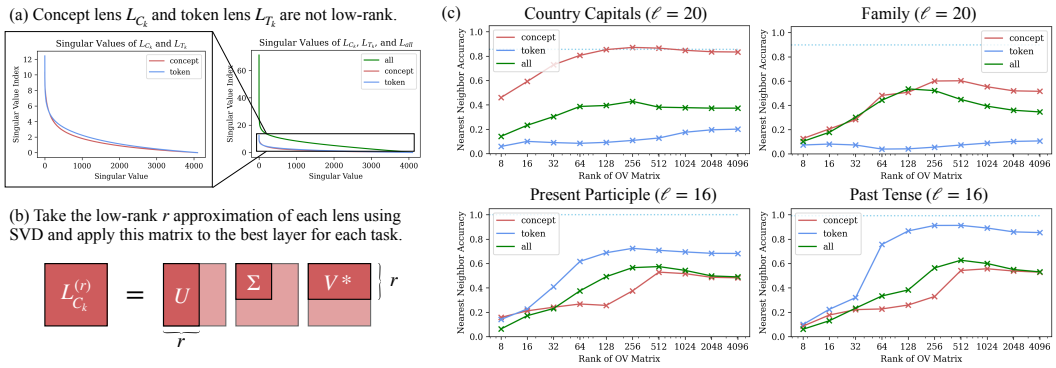

Figure 3: Reducing the rank of L by taking the top-r singular components does not damage nearest-neighbor accuracy. (a) Inspecting the singular values of our concept lens, L_{C_k}, and token lens, L_{T_k}, these transformations appear to be full-rank. (b) Regardless, we take r-rank approximations of these transformations by setting all singular values after the top-r values to zero. (c) We choose the best layer for each task from Figure 1 and reduce the rank of every L in this way. Performance is maintained for ranks as low as $r = 256$. Note that values for $r = 4096$ are the same as results from Figure 1.

References

[1] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

[2] Sheridan Feucht, Eric Todd, Byron Wallace, and David Bau. The dual-route model of induction. In *Second Conference on Language Modeling*, 2025.

[3] Vedang Lad, Jin Hwa Lee, Wes Gurnee, and Max Tegmark. The remarkable robustness of llms: Stages of inference?, 2025.

[4] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[5] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In Lucy Vanderwende, Hal Daumé III, and Katrin Kirchhoff, editors, *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[6] Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. Function vectors in large language models. In *Proceedings of the 2024 International Conference on Learning Representations*, 2024. arXiv:2310.15213.

[7] Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Niko lay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cris tian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melissa Hall Melanie Kambadur, Sharan Narang, Aur'elien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288, 2023.

A Full Parallelogram Arithmetic Results

Table 1: Prefixes and examples for parallelogram datasets. Prefixes are used for all words in the dataset, e.g., "She travelled to Athens", "She travelled to Greece", etc.

Task	Example	Prefix
Word2Vec Tasks (Mikolov et al., [4])		
capital-common-countries	(Athens, Greece)	She travelled to
capital-world	(Valletta, Malta)	She travelled to
currency	(Algeria, dinar)	You will have to pay in
city-in-state	(Tulsa, Oklahoma)	She travelled to
family	(uncle, aunt)	Did you talk to her
gram1-adjective-to-adverb	(amazing, amazingly)	Here is a random word in English:
gram2-opposite	(likely, unlikely)	Here is a random word in English:
gram3-comparative	(big, bigger)	Here is a random word in English:
gram4-superlative	(great, greatest)	Here is a random word in English:
gram5-present-participle	(look, looking)	Here is a random word in English:
gram6-nationality-adjective	(Brazil, Brazilian)	Here is a random word in English:
gram7-past-tense	(jumping, jumped)	Here is a random word in English:
gram8-plural	(cow, cows)	Here is a random word in English:
gram9-plural-verbs	(search, searches)	Here is a random word in English:
Function Vector Tasks (Todd et al., [6])		
antonym	(wish, regret)	Here is a random word in English:
synonym	(dangerous, hazardous)	Here is a random word in English:
present-past	(separate, separated)	Here is a random word in English:
singular-plural	(spoon, spoons)	Here is a random word in English:
word-length	(7, pelican)	Here is a random word in English:
capitalize-first-letter	(R, remember)	Here is a random word/character:
capitalize-last-letter	(T, quilt)	Here is a random word/character:
capitalize-second-letter	(N, snake)	Here is a random word/character:
lowercase-first-letter	(r, RACE)	Here is a random word/character:
lowercase-last-letter	(e, OBSERVE)	Here is a random word/character:
next-capital-letter	(ostrich, P)	Here is a random word/character:
next-item	(May, June)	Here is a random word/character:
prev-item	(twenty, nineteen)	Here is a random word/character:
capitalize	(peach, Peach)	Here is a random word in English:
country-capital	(Indonesia, Jakarta)	She travelled to
country-currency	(Slovenia, Euro (EUR))	You will have to pay in
english-french	(discussed, discuté)	Voici un mot aléatoire en français:
english-german	(officials, Beamte)	Hier ist ein beliebiges Wort im Deutschen:
english-spanish	(forwards, adelante)	Aquí hay una palabra arbitraria en español:
landmark-country	(Chile, Wellington Island)	On vacation, we went to
national-parks	(California, Sequoia National Park)	On vacation, we went to
park-country	(Nepal, Bardya National Park)	On vacation, we went to
person-instrument	(piano, Tadd Dameron)	I am a big fan of
person-occupation	(architect, Gunnar Birkerts)	I am a big fan of
person-sport	(basketball, Kevin Durant)	I am a big fan of
product-company	(Apple, iPhone 5)	I am a big fan of
sentiment	(positive, It's a masterpiece.)	Here's my take on this film:

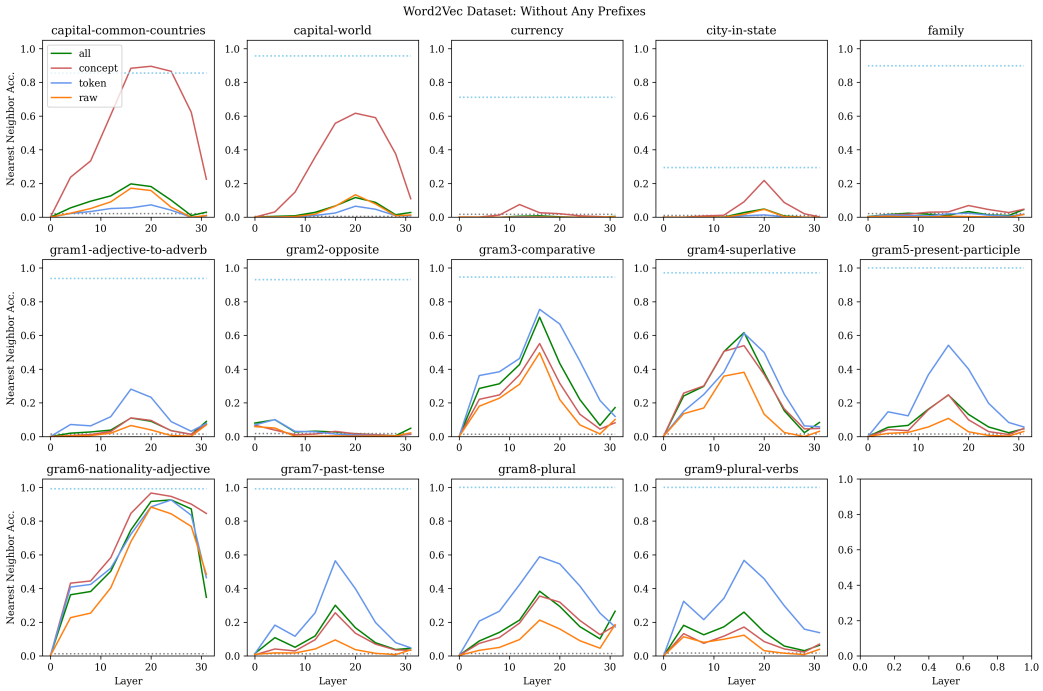

Figure 4: Nearest-neighbor accuracy for all `word2vec` tasks [4] without any prefixes (i.e., feeding each word to the model by itself with no context). Comparing with Figure 2, certain tasks like "currency" are much less accurate; this may be because currencies like "real" are not immediately recognizable out of context. However, accuracy is slightly better for "capital-common-countries" and "gram6-nationality-adjective" without any prefixes.

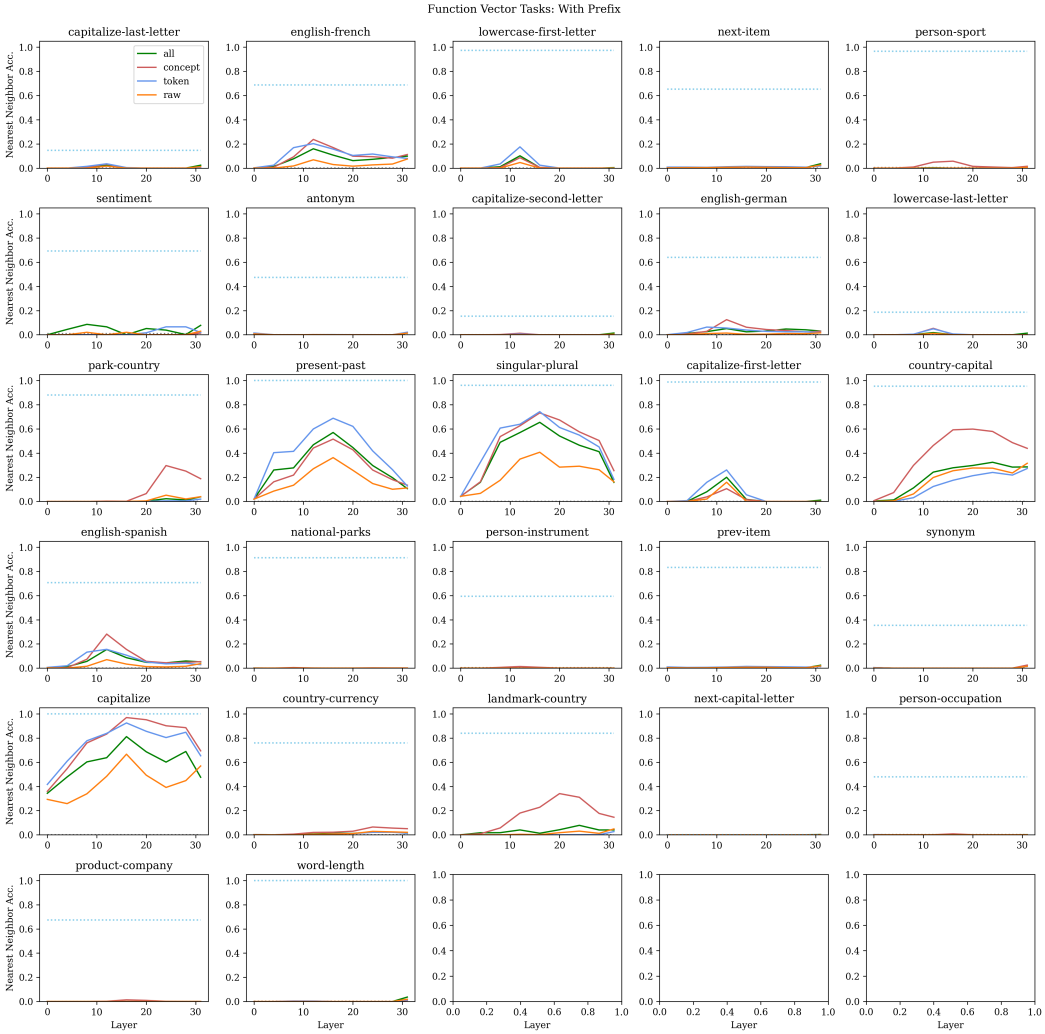

Figure 5: Nearest-neighbor accuracy for all function vector tasks [6] with prefixes for each task listed in Table 1. Dotted gray lines indicate guessing accuracy (out of all possible neighbors/words in the dataset). Dotted light blue lines indicate 5-shot ICL accuracy, i.e., the best possible performance this model can have for this task. The failure of many of these tasks is unsurprising: some tasks are many-to-one relations that may not be represented as parallelograms ("capitalize-first-letter"), whereas others may be too complex to be directly encoded in the model's embedding space ("national-parks"). Note: "country-currency" includes more countries (197) than the `word2vec` "currency" task (30).

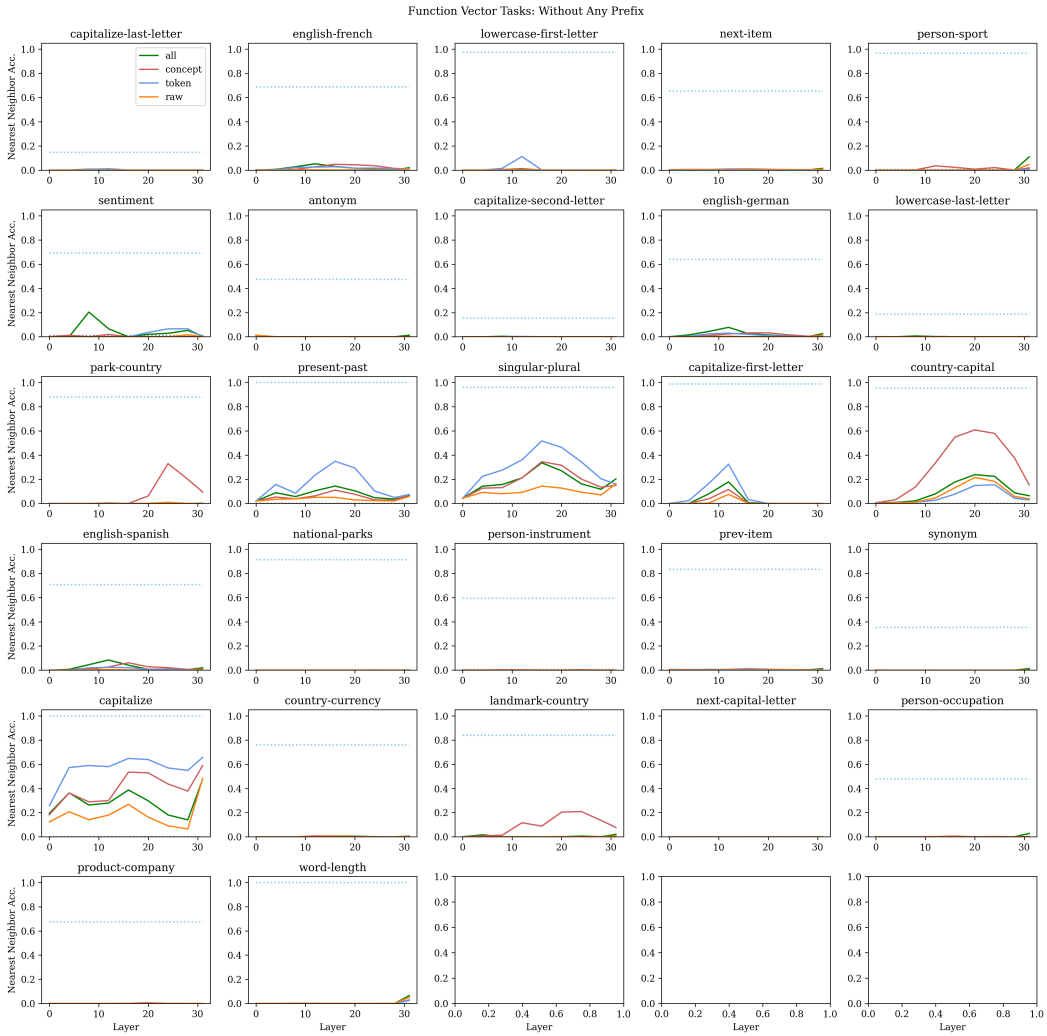

Figure 6: Nearest-neighbor accuracy for all function vector tasks [6] without any prefixes (i.e., feeding each word to the model by itself with no context). Dotted gray lines indicate guessing accuracy (out of all possible neighbors/words in the dataset). Dotted light blue lines indicate 5-shot ICL accuracy for this task, i.e., the best possible performance this model can have for this task. Without prefixes, accuracy for many tasks is lower overall.