Task Vectors in ICL for Complex Tasks

Guy Levy
318645694

Liad Iluz
208427435

Tal Dvir
206592867

Jonathan Makovsky
208935916

Abstract

The emergence of task vectors has recently become a subject of exploration in the context of in-context learning (ICL) within large language models (LLMs). Building upon this foundation, our research investigates the behavior of task vectors in more complex tasks than those previously studied. We show that as task complexity increases, the accuracy achieved using task vectors within a single layer diminishes. Nevertheless, through statistical methods and tests, we show that the emergence of task vectors remains prominent. Subsequently, by using arithmetic manipulations, we develop more sophisticated task vectors that enhance performance on these complex tasks or achieve comparable results using simpler tasks. Our findings provide deeper insights into the mechanisms of ICL and offer practical approaches for improving model accuracy and efficiency in the ICL paradigm.[1]

1 Introduction

In-context learning (ICL) is an NLP method where a LLM is provided with a prompt containing a small set of demonstrations (the context) and need to understand and perform a specific task based on this context. A relevant question in ICL is whether and how the mechanism understands the specific task within the ICL prompt. To address this, researchers look for a vector created within the layers of the transformer model during the execution of the ICL prompt, known as a task vector (TV) or function vector (Hendel et al., 2023; Todd et al., 2024).

This field holds significant importance as it promises to enhance the adaptability and efficiency of models in performing a wide array of tasks without extensive retraining. The ability to understand and manipulate task vectors can lead to more robust and versatile AI systems, capable of better generalization and performance across diverse applications.

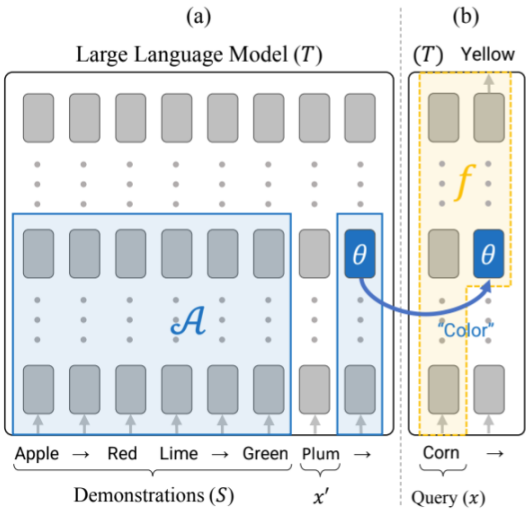


Figure 1: The vector Θ represents the task vector extracted from the ICL prompt (step (a)) and injected into the corresponding layer of the new and independent query (TV prompt, step (b)). This figure was taken from Hendel et al. (2023).

In the main work we based on (Hendel et al., 2023), the authors demonstrated whether learning of ICL prompts that related to specific tasks on LLMs creates a simple form of vector representation – one of the model's layers. Since our study directly extends this work, we provide further details, including relevant terms related to our research, in subsection 1.1. While this work has provided valuable insights and reports positive results, it primarily focuses on relatively simple tasks. The exploration of task vector mechanisms in more complex tasks remains unexplored within this framework. Our research addresses this gap by investigating the emergence of task vectors within the context of complex tasks.

[1]This work represents an extension of the research conducted by Hendel et al. (2023), building upon significant portions of their methods and implementation to address our research questions. Code of all implementation at https://github.com/Guy-Levy/icl_task_vectors.

Additionally, we explore methods to enhance the performance of complex tasks by manipulating their task vectors. Inspired by previous approaches (Todd et al., 2024; Ilharco et al., 2022), we approximate the task vectors of unsuccessful samples to the center of successful task vectors. Furthermore, we employ task vector arithmetic using a weighted sum to assemble new task vectors from building blocks, aiming to improve performance on complex tasks or get comparable results using simple methods.

To clearly present our objectives, we separate our research into two sub-sections, each revealing a different aspect: **(1) The Limits of Complexity**, which focuses on comparing the emergence of task vectors across different complexities level of tasks; and **(2) Simplifying Methods for Complex Tasks by Task Vector Arithmetic**, which explores how arithmetic manipulation of task vectors can enhance the performance of more complex tasks. For example, how complex tasks, which can be naturally decomposed into simpler sub-tasks, can be understood and represented by arithmetic operations.

Our research shows that (1) although accuracy decreases as tasks become more complex, the emergence of task vectors can still be confirmed through statistical and analytical tests; and (2) simple arithmetic manipulations on task vectors can create more effective and accurate task vectors, thereby enhancing model performance on complex tasks.

1.1 In-Context Learning Creates Task Vectors

The main study by Hendel et al. (2023) employs a comprehensive experimental approach using various LLMs (LLaMA, GPT-J, and Pythia) across 18 tasks categorized into Algorithmic, Translation, Linguistic, and Knowledge tasks. The process begins by establishing baseline accuracy through a zero-shot prompt (e.g., "Corn →"). Subsequently, they use full ICL prompts (few-shot prompts) consisting of several demonstrations and a dummy query (e.g., "Apple → Red; Lime → Green; Corn →") to derive the expected outputs and calculate task vectors.

To find the specific layer that holds the task vector, they evaluate the accuracy of new and independent zero-shot prompts (e.g., "Plum →") after injecting each layer of the ICL prompt's last "→" and running the model from that layer onward. We refer to this last prompt as the TV prompt (task vector prompt). The task vector is identified as the layer that yields the highest accuracy on the TV prompt. Figure 1 from the paper illustrates these concepts.

The study demonstrates that task vectors effectively encapsulate their tasks, enabling LLMs to perform with high accuracy even when the context is indirectly provided. By injecting task vectors into the optimal layers of LLMs, they observed that the models could understand and execute new zero-shot prompts (TV prompts) with performance comparable to the original ICL prompts. This finding validates their hypothesis that ICL acts by compressing the training set into a task vector, which then modulates the transformer for specific task execution.

2 Methods

Models. We utilize various open LLMs, including GPT-J 6B Pythia 6.9B, and Pythia 12B. The results are compared across all models. Appendix A.2 shows more details on Models.

Tasks. Similar to the approach taken by Hendel et al. (2023), we categorize all tasks into four groups: (1) Algorithms and Arithmetic, (2) Knowledge, (3) Linguistic, and (4) Multi-tasks. Within each category, we examine diverse tasks with varying levels of complexity. We define task complexity based on three criteria: (i) the number of logical steps required, (ii) the complexity of arithmetic calculations involved, and (iii) the number of unrelated sub-tasks necessary to obtain the final result. To address our research question, we investigate tasks that can be extended into more complex forms and those that can be composed into complex tasks. Table 3 shows all tasks and their complexity rank.

Unlike the main paper on which our study is based, we introduce a fourth category – "Multi-tasks". This category comprises tasks constructed by combining simple tasks from multiple categories and domains, enriching the types of complex tasks. For instance, the task "antonym to reverse" integrates linguistic tasks (antonym) with algorithmic tasks (reverse word). Table 1 presents partial view of our complex tasks. Appendix A.1 shows all tasks and describe the datasets creation.

2.1 The Limits of Complexity

To better understand the limits of task vector emergence on ICL, we took the following steps:

Category	Task	description	Example
Algorithmic & Arithmetic	Average list	Given a list of numbers return its average	[1,6,2] → 3
	First to upper	Given a word return its first letter uppercase	work → W
Knowledge	City to PM	Given a city return the PM of the country the city in	Munich → Olaf Scholz
Linguistic	Antonym to superlative	Given a word return its antonym in superlative	beautiful → ugliest
Multi-tasks	Antonym to reverse	Given a word return its antonym in reverse order	up → nwod

Table 1: Examples of complex tasks in each category.

2.1.1 Increasing Task Complexity

We selected tasks with relatively increasing complexity to determine the limits of task vector emergence from ICL, using the methodology outlined in the main paper, adapting it to handle more complex tasks:

1. **Accuracy Test:** We reports on baseline accuracy, ICL accuracy (prompt with context) and TV accuracy (prompt without context but with injected task vector) for each task and check whether injection of task vector bring TV accuracy to be closed to ICL accuracy.

2. **Task Vector Space:** We plot t-SNE graph to illustrate the closeness of tasks vector within the same task. Additionally, for each task, we present the variability of its task vectors by showing the distribution of the distances between task vectors within the same task as well as with other tasks.

2.1.2 Correlation Analysis

To better understand the complexity levels and the emergence of task vectors regardless of absolute accuracy results, we examined two correlation tests over three groups: simple tasks, medium tasks, and complex tasks:

1. **Correlation between ICL prompt size (number of demonstrations) and TV accuracy:** We run the accuracy test with ICL prompts of sizes 2, 5, and 8. We hypothesized that as tasks become more complex, the ICL would require more demonstrations to "understand" the task, thereby making the task vector more meaningful for TV prompt.

2. **Correlation between distance of task vector to mean task vector and TV accuracy:** We considered the mean of task vectors from all ICL prompts and also those from just the correct TV prompts. Then, we run the accuracy test on other samples, and compute the distance of each task vector to the means computed earlier, while computing the accuracy of this task vector on TV prompts. We aimed to determine whether the accuracy of TV prompts is related to this distance, even in cases where tasks generally tend to fail.

2.2 Simplifying Methods for Complex Tasks by Task Vector Arithmetic

Baseline: Following the procedure described in the main paper, we assessed the TV accuracy for each task.

2.2.1 Using Mean Task Vector

We investigated whether the mean task vector could serve as a more effective injected task vector, potentially enhancing performance. For each task, we first run the accuracy test and compute the mean task vector for all tasks, the mean task vector for only correct TV prompts, and the best layer to inject the task vector. Then, we generate only TV prompts and calculate their accuracy where each mean task vector was injected into the best layer.

2.2.2 Complex Tasks as a Combination of Simple Ones

To examine the mechanism of task vectors for complex tasks that build on simpler ones, we aimed to create a task vector for the complex task through a linear combination of the simpler ones. We determined the coefficients via a cross-validation procedure. Specifically, Consider a complex task T that

can be naturally interpreted as consisting of two sub-tasks t_1 and t_2. Denote the average task vectors of the sub-tasks as v_1, v_2 respectively. Through cross-validation, we sought $\lambda_1, \lambda_2 \in [0, 1]$ that maximize the TV accuracy of task T based on task vector $V = \lambda_1 v_1 + \lambda_2 v_2$. This optimization problem was solved numerically using the Nelder-Mead optimization method, with the resulting task vector being injected into the best layer identified in the baseline procedure at each step.

3 Results

3.1 The Limits of Complexity

3.1.1 Increasing Task Complexity

Following the methodology of Hendel et al. (2023), we observed lower accuracy on ICL and TV prompts for more complex tasks. However, this does not necessarily imply that task vectors do not emerge in complex tasks. For instance, the limitations of the LLMs may also contribute to this lower accuracy. Table 6 reports all results.

The t-SNE graph of task vectors that we generated, shown in Figure 5, indicates that even in complex tasks, the task vector space effectively separates each task into clear groups, with closely related tasks grouped next to each other. For instance, the tasks "first letter to upper," "prev of first letter," and "first letter" are relatively close to each other. Similarly, all tasks involving "mayor" are grouped together, and the same applies to tasks involving "prime minister". These examples illustrate that tasks of varying complexity can still be effectively clustered.

Additionally, the variability measurement of Θ within the same task and with other tasks, shown in Figure 6, reveals a highly concentrated distribution of distances for task vectors within the same task, while there is a larger spread when considering distances with other tasks. This suggests that task vectors for the same task are more similar to each other compared to task vectors for different tasks, reinforcing the clustering observed in the t-SNE graph.

3.1.2 Correlation Analysis

Using accuracy of tasks across different prompt sizes, as shown in Figure 3, we conducted a Spearman Rank Correlation analysis to examine the relationship between ICL and TV accuracy. For simple tasks, the correlation was 0.434 with a p-value of 0.043, indicating a statistically significant moderate positive correlation. Medium tasks showed a slightly weaker positive correlation of 0.391 with a p-value of 0.058, which is marginally significant. Complex tasks had a weaker correlation of 0.236 with a p-value of 0.288, making it statistically insignificant but still positive.

These results highlight a consistent positive correlation across all task complexities, suggesting that ICL and TV accuracy tend to move in the same direction. Additionally, the gradual decrease in correlation strength ($0.4347 \rightarrow 0.3918 \rightarrow 0.2369$) implies that while the relationship persists, it weakens as task complexity increases.

Although the statistical significance of the distance-accuracy correlation test was not definitive, the results show a consistently negative average correlation across all levels of complexity, with relatively small error bars. For instance, the average correlation for simple tasks on Pythia 12B was -0.2 ± 0.23, whereas for complex tasks, it was -0.35 ± 0.11. Figure 4 shows the results over all tasks.

3.2 Simplifying Methods for Complex Tasks by Task Vector Arithmetic

In this section, we aim to enhance the performance of task vectors on complex tasks by applying arithmetic manipulations to these vectors.

3.2.1 Using Mean Task Vector

As shown in Figure 2, injecting the mean task vector of correct examples improves the accuracy of TV prompts. Notably, using the mean task vector of all tasks as the injected task vector leads to similar results for most tasks. This suggests that the mean task vector is a robust representation, effectively enhancing model performance across various tasks.

3.2.2 Complex Tasks as a Combination of Simple Ones

Using the mean task vectors of examples from simple tasks and applying optimization to find the best coefficients λ_1 and λ_2, we achieved similar results to the TV accuracy of the complex tasks. This approach was effective not only when using the task vectors of correct examples, but also when using the mean task vectors of all examples from the simple tasks. Moreover, as table 2 shows, the coefficients of "less accurate" task vector tend to be smaller than of the other task.

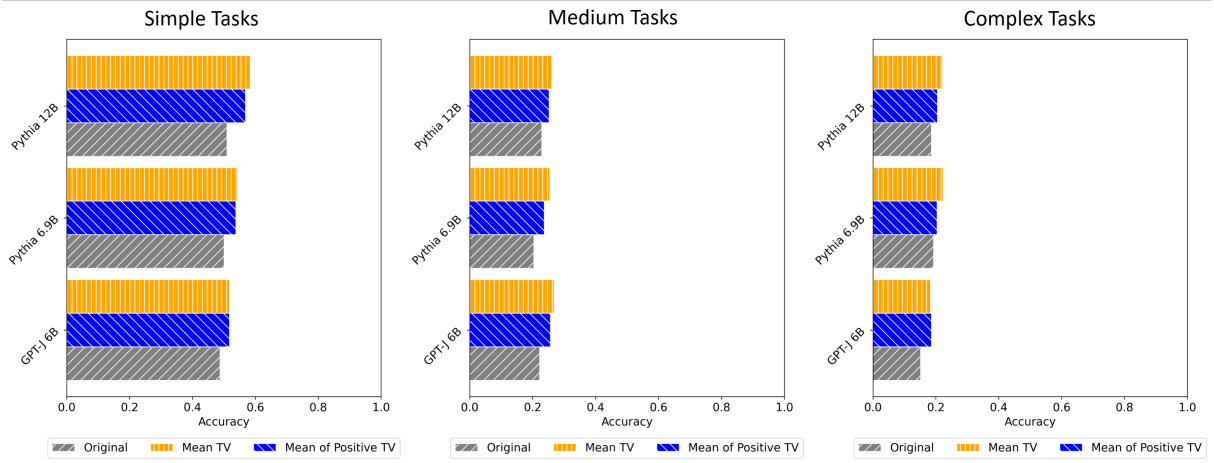


Figure 2: The baseline accuracy represents the average TV accuracy of all tasks using the general procedure. The mean TV accuracy and the mean of positive TV accuracy represent the average accuracy of all tasks using the mean task vectors of the all examples and from correct examples, respectively, as task vector to inject.

4 Related Work

Task Vector Representation. As presented in Section 1.1, the study by Hendel et al. (2023) defines task vectors as one of the layers of the ICL prompt's last arrow ("→"). These vectors encapsulate task information, enabling LLMs to perform specific tasks effectively, even with zero-shot prompts. Although the paper by Todd et al. (2024) is closely related to the work by Hendel et al. (2023), it employs a different approach to compute task vector representation (referred to as function vector in this paper). For each task t, they compute the task vector $v_t = \sum_{a_{\ell j} \in A} \bar{a}^t_{\ell j}$. Here, $\bar{a}^t_{\ell j}$ is the average of $a_{\ell j}(p^t_i)$ over all input prompts p^t_i, and $a_{\ell j}(p^t_i)$ describes the projection of the output of the j'th attention head with respect to input prompt p^t_i. The set A of projections is determined by the highest scored among indirect effect measurements of the vectors, conceptually similar to the strategy of finding the best layer to inject, as done in Hendel et al. (2023).

In contrast, Ilharco et al. (2022) defined a task vector as the difference between the weights of a pre-trained model and the weights after fine-tuning. Their work focused on image models rather than language models. We used the definition of Hendel et al. (2023).

Prompt Types. Since ICL is based on the context (prompt), the prompt type can significantly affect performance. Common types of prompts include zero-shot (query without output), one-shot, and few-shot, which differ by the number of demonstrations given in each prompt. Hendel et al. (2023) employed zero-shot and few-shot ICL prompts to establish baseline accuracy and ICL accuracy and to derive task vectors. Their prompt structure used the word "example:" at the start of each demonstration, and the connection between query and output was denoted by "→".

Todd et al. (2024) also used these types of prompts but phrased their zero-shot prompts more naturally using sentence completions, such as "The word x means" or "When I think of x, I usually". Ilharco et al. (2022) focused on image classification and generation tasks, and therefore their prompts included images. For example, they showed a list of real images with related drawing images to help the model understand the task of generating a real drawing image from a real one. In our study, we focused on the same prompt types introduced in Hendel et al. (2023), while we also examined the correlation of the size of the few-shot ICL prompt to ICL and TV accuracy.

Arithmetic Manipulations in Task Vectors. Both Hendel et al. (2023) and Todd et al. (2024) examined the effects of injecting the mean task vector (as defined in their respective studies) into a conflicting task. In both cases, this was not the main focus of the papers. For instance, Hendel et al. (2023) found that injecting the mean task vector of the "next letter" task into the ICL prompt for "previous letter" tasks resulted in significantly lower accuracy. Similarly, Todd et al. (2024) showed that injecting the task vector of an opposite word led to the zero-shot prompt "The word simple means" outputting

Model	Complex task	Base	New	Sub-tasks	Base	λ
GPT-J 6B	Prev of first letter	0.06	0.01	first letter	0.8	0.5
				prev letter	0.34	0.5
	Next of first letter	0.07	0.06	first letter	0.8	0.5
				next letter	0.92	0.5
	First letter to upper	0.89	0.63	first letter	0.8	0.098
				to uppercase	0.99	0.901
	Abs and round	0.39	0.54	round	0.52	0.5
				absolute	0.44	0.5
Pythia 6.9B	Prev of first letter	0.04	0.03	first letter	0.88	0.5
				prev letter	0.27	0.5
	Next of first letter	0.05	0.05	first letter	0.88	0.29
				next letter	0.64	0.5
	First letter to upper	0.91	0.97	first letter	0.97	0.368
				to uppercase	0.99	0.709
	Abs and round	0.44	0.53	round	0.48	0.493
				absolute	0.57	0.506
Pythia 12B	Prev of first letter	0.04	0.05	first letter	0.97	0.5
				prev letter	0.22	0.5
	Next of first letter	0.03	0.03	first letter	0.97	0.5
				next letter	0.92	0.5
	First letter to upper	0.95	0.99	first letter	0.97	0.368
				to uppercase	1.00	0.631
	Abs and round	0.53	0.45	round	0.44	0.5
				absolute	0.46	0.5

Table 2: Base accuracy refers to the standard TV accuracy following the general procedure. New accuracy is the accuracy obtained by applying the task vector as a linear combination of the task vectors of sub-tasks, using the specified lambda coefficients.

"complex". Notably, the task vector definition in Todd et al. (2024) is based on averaging.

Ilharco et al. (2022) applied more arithmetic manipulations on task vectors: (1) negation—to make language models produce less toxic content, (2) addition—to build multi-task image models, and (3) addition with subtraction—to improve domain generalization of task analogies (A is to B as C is to D). This last manipulation was also done in Todd et al. (2024) by combining vectors of simple tasks (e.g., Country-Capital) with item positions in a list (e.g., First-Last). All these studies demonstrated that task vectors could be manipulated through arithmetic operations to steer model behavior and achieve positive results. We focused on mean operation and weighted sum as arithmetic manipulations, defining a more general way to encapsulate complex tasks with simpler ones.

5 Discussion and Conclusions

Our study provides valuable insights into the nature and effectiveness of task vectors (TVs) in complex language tasks. We present several key findings that support the continued relevance of TVs even as task complexity increases. Notably, we observed a consistent relationship between TV accuracy and ICL accuracy across various task complexities. This correlation, while weakening with increased complexity, remains positive, indicating that TVs continue to be reliable task representations. Furthermore, the tests we did to examine the task vector space for complex tasks maintain clear and distinct patterns, similar to those observed for simpler tasks. This suggests that TVs preserve coherent representations regardless of task complexity.

A significant discovery in our research is the effectiveness of mean TV injection. We found that injecting mean task vectors, including all vectors and not just the correct ones, significantly enhances TV accuracy. This "black-box" method, which averages all examples regardless of individual accuracy, simplifies the process and improves overall performance.

Our research also explored the concept of complex tasks as combinations of simpler ones, yielding important insights. We found a discernible connection between the TVs of simpler tasks and those of more complex tasks, suggesting that leveraging simpler TVs can be beneficial in addressing more complex challenges. While a weighted sum of TVs provides good results, there is still space for more sophisticated combination methods to further optimize performance. Notably, using datasets

and computations for simpler tasks can produce acceptable results for complex tasks, even without dedicated them for the complex task. We think that this approach not only may simplify the process, but also hold promise for achieving higher accuracy in relatively complex tasks, especially in models with architectural limitations.

Limitations and Future Research

This study extends the research conducted by Hendel et al. (2023) and relies heavily on their methods, code, and prompt format (x $\rightarrow$ y). This constraint may limit the depth of our conclusions regarding task vector emergence in complex tasks. Exploring a wider variety of ICL prompts, as done by Todd et al. (2024), could potentially yield richer insights. Additionally, we focused on task vectors derived from a single layer at the final '$\rightarrow$' in the ICL prompt. As tasks become more complex, the structure of task vectors may also become more intricate, possibly requiring expressions from multiple layers.

Furthermore, similar to the limitations mentioned by Hendel et al. (2023), our study primarily targeted tasks that involve single tokens or those with a limited number of expected tokens, such as names of people. The behavior of task vector emergence could vary significantly with more comprehensive tasks. Due to time constraints, we did not investigate complex tasks that involve multi-token outputs. The latter requires developing techniques to evaluate the accuracy of prompts that can be phrased in various ways.

From the perspective of LLMs and tasks, we did not run our procedures on the LLaMA models, which produced stronger results in Hendel et al. (2023). Based on the chosen LLMs, some of the knowledge-based tasks may have yielded lower accuracy due to the use of models trained with less updated data, impacting tasks that rely on current information, such as identifying the current prime minister of a country. Therefore, when choosing tasks of different complexities, the limitations of the LLMs should also be considered.

Additionally, task difficulty and sub-tasks were classified by human benchmarks without any sophisticated methodology to determine the complexity of tasks and their natural sub-tasks. Further consideration of this aspect, taking into account the constraints of the LLMs mentioned earlier, can reveal more interesting insights and potentially improve the accuracy and performance of tasks in specific models. However, without doing so, we preferred to classify all tasks before running the models and to address equally the levels of complexity that we determined.

Acknowledgements

We would like to thank Mor Geva for the valuable advice and insights that helped refine our research question and methodology.

References

Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-context learning creates task vectors. *Findings of the Association for Computational Linguistics: EMNLP 2023*.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2022. Editing models with task arithmetic. *The Eleventh International Conference on Learning Representations*.

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

A Appendix

This is a section of additional details on some parts of the paper.

A.1 Tasks and Datasets

Tasks. Based on a few simple tasks from Hendel et al. (2023), we developed more complex variations. We examined a total of 40 tasks: 16 were categorized as complex tasks, 9 as intermediate tasks, and the remainder as simple tasks, which had already been studied in related works. Table 3 lists all the tasks we examined along with relevant examples.

Datasets. The GitHub repository link provided includes all datasets and associated code used for dataset creation. Most algorithmic and arithmetic tasks were generated programmatically. Knowledge-based tasks were created using GPT [2] and Claude [3]. Some simple tasks were adapted from Hendel et al. (2023). Table 4 shows all datasets we used in this project with more detailed.

[2] https://chatgpt.com/

[3] https://claude.ai/

To ensure sample independence in our results, we (1) shuffled datasets, (2) utilized diverse sources and samples, and (3) generated datasets randomly. For instance, algorithmic sequences were generated using random numbers within specific ranges. Additionally, for the city-country dataset, we restricted capital cities and included cities from diverse countries worldwide.

A.2 Models and Run

Models. We based on models that used in Hendel et al. (2023) for getting comparable results. Table 5 shows the models we used with their details.

Run Environment. Our runs were done on a machine with 4 x RTX4090 GPUs and 170GB of ram which we rented from `http://Vast.ai`.

Model	Params	Dims	Layers	Heads
GPT-J	6B	4096	28	16
Pythia	6.9B	4096	32	32
	12B	5120	36	40

Table 5: The models we used in our work.

Category	Complex Task	Sub-Task	Example
Algorithmic & Arithmetic	Average list		[1,6,2] → 3
		Sum list*	[1,6,2] → 9
		Length list*	[1,6,2] → 3
		Division	[9,3] → 3
	Max diff list		[17,8,3] → 14
		Max list*	[17,8,3] → 17
		Min list*	[17,8,3] → 3
		Subtraction	[17,3] → 14
	Round and abs		-9.8 → 10
		Round	-9.8 → -10
		Abs	-10 → 10
	First to upper		work → W
		First letter	work → w
		to upper	w → w
	Next first letter		home → i
		First letter	home → h
		Next letter	h → i
	Prev first letter		home → g
		First letter	home → h
		Prev letter	h → g
	Count char		["amaa", "a"] → 3
		0-1 list*	["amaa", "a"] → [1,0,1,1]
		Sum list*	[1,0,1,1] → 3
Knowledge	City to PM birth year		Munich → 1958
	City to PM		Munich → Olaf Scholz
		City to country	Munich → Germany
		Country to PM	Germany → Olaf Scholz
	Person to birth year		Olaf Scholz → 1958
	Location to capital mayor		See of Galilee → Moshe Leon
	Country to capital mayor		Israel → Moshe Leon
		Country to capital	Israel → Jerusalem
		City to mayor	Jerusalem → Moshe Leon
		Location to country	See of Galilee → Israel
Linguistic	Antonym to superlative		beautiful → ugliest
		Antonym	beautiful → ugly
		Adjective to superlative	ugly → ugliest
	Past to opposite gerund		came → going
		Past to present	came → come
		Present antonym	come → go
		Present gerund	go → going
Multi-tasks	Antonym to reverse		up → nwod
		Antonym	up → down
		Reverse*	down → nwod
	Antonym to Spanish		old → joven
		Antonym	old → new
		ENG to SPN	new → joven

Table 3: Tasks list separated by criteria and complexity. Each complex task is given before its sub-tasks. Some of the sub-tasks are simple tasks and some of them are more complex (denoted by *).

File name (.json)	Size	Source	Example
array_average	487	Code	"[4,7]":"5.5"
array_sum	498	Code	"[3,1,6]":"10"
array_length	485	Code	"[10,0,2]":"3"
division	518	Code	"[900,50]":"18"
array_max_diff	543	Code	"[103,46,35]":"68"
array_max	543	Code	"[103,46,35]":"68"
array_min	542	Code	"[60,183]":"60"
subtraction	265	Code	"[200,8]":"192"
abs_round	548	Code	"-6.151":"6"
round	546	Code	"3.855":"4"
absolute	509	Code	"-1":"1"
first_letter_to_upper	549	Code	"propone":"P"
first_letter	550	Code	"delay":"d"
to_uppercase	550	Code	"k":"K"
next_of_first_letter	549	Code	"wha":"x"
prev_letter	550	Code	"i":"h"
city_PMbirthyear	386	Code**	"Munich":"1958"
city_PM	390	Claude	"Naples":"Giorgia Meloni"
city_country	400	Claude	"Tel Aviv":"Israel"
country_PM	180	GPT-4o	"Vatican City":"Pope Francis"
person_birthYear	312	GPT-4o + Claude	"Paul McCartney":"1942"
location_capitalMayor	230	Code**	"Saharsa district": "Shelly Oberoi"
country_capitalMayor	190	Code**	"Mauritius": "Mahfooz Saib"
country_capital	198	Claude	"Mauritius": "Port Louis"
city_mayor	396	Claude	"San Mateo": "Amourence Lee"
location_country	300	Hendel et al. (2023)	"Saharsa district": "Morocco"
antonym_superlative	336	Claude	"youthful": "oldest"
antonyms	298	GPT + Hendel et al. (2023)	"serious": "funny"
adjective_superlative	268	GPT	"clean": "cleanest"
past_oppositeGerund	334	Code	"approached": "retreating"
past_present	1005	Hendel et al. (2023)*	"apologized": "apologize"
present_gerund	1008	Hendel et al. (2023)	"interest": "interesting"
antonym_to_reversed	162	Code**	"high": "wol"
reversed_word	550	Code	"polyzoon": "noozylop"
en_es	719	Hendel et al. (2023)	"charge": "cargo"

Table 4: Tasks list separated by criteria and complexity. Each complex task is given before its sub-tasks. Some of the sub-tasks are simple tasks and some of them are more complex. The mark * implies that we did few code adjustments and corrections to the base source. The mark Code** indicates that we combined two exsiting datasets by code procedure to create a new one. All Code sources were created using GPT [4] and Claude [5].

Table 6: Complete results of the main experiment for all tasks and models.

Model	Category	Task	General Baseline	ICL	TV	Average Baseline	All TV	Pos. TV
GPT-J 6B	Algorithmic	**Average list**	0.08	0.04	0.04	0.02	0.1	0.06
		Sum list	-	-	-	-	-	-
		Length list	0.32	0.56	0.32	0.25	0.43	0.47
		Division	0.02	-	-	-	-	-
		Max diff list	-	0.01	-	-	0.01	0.02
		Max list	0.74	0.5	0.6	-	0.01	0.02
		Min list	0.46	0.59	0.48	0.51	0.48	0.53
		Subtraction	0.04	0.04	0.03	-	-	-
		Round and abs	0.34	0.52	0.39	0.3	0.32	0.34
		Round	0.52	0.66	0.57	0.52	0.52	0.52
		Abs	0.53	0.98	0.52	0.44	0.52	0.52
		First to upper	0.98	0.92	0.86	0.86	0.92	0.92
		First letter	0.88	0.99	0.82	0.8	0.87	0.86
		to upper	0.94	1.00	1.00	0.99	0.96	0.96
		Next first letter	0.08	0.07	0.07	0.03	0.12	0.13
		Next letter	0.78	0.81	0.94	0.92	0.96	0.96
		Prev first letter	0.06	0.08	0.06	0.06	0.04	0.04
		Prev letter	0.34	0.38	0.36	0.34	0.39	0.39
	Knowledge	**City to PM birth year**	0.36	0.32	0.38	0.32	0.44	0.44
		City to PM	0.36	0.32	0.38	0.32	0.44	0.44
		City to country	0.36	0.32	0.38	0.32	0.44	0.44
		Country to PM	0.36	0.32	0.38	0.32	0.44	0.44
		Person to birth year	0.36	0.32	0.38	0.32	0.44	0.44
		Location to capital mayor	0.36	0.32	0.38	0.32	0.44	0.44
		Country to capital mayor	0.36	0.32	0.38	0.32	0.44	0.44
		Country to capital	0.36	0.32	0.38	0.32	0.44	0.44
		City to mayor	0.36	0.32	0.38	0.32	0.44	0.44
		Location to country	0.36	0.32	0.38	0.32	0.44	0.44
	Linguistic	**Antonym to superlative**	-	0.17	-	-	-	-
		Antonym	0.48	0.75	0.38	-	-	-
		Adjective to superlative	-	0.17	-	-	-	-
		Past to opposite gerund	-	0.14	-	-	-	-
		Past to present	0.28	0.9	0.12	0.09	0.11	0.11
		Present gerund	0.08	0.32	0.38	0.32	0.44	0.44
	Multi-tasks	**Antonym to reverse**	-	-	-	-	-	-
		Reverse	0.08	0.01	0.06	0.05	0.04	0.06
		en to es	0.06	0.73	0.02	0.04	0.05	0.05
Pythia 6.9B	Algorithmic	**Average list**	0.08	0.06	0.03	0.02	0.03	0.02
		Sum list	-	0.55	0.45	-	-	-
		Length list	0.44	0.55	0.52	0.44	0.52	0.52
		Division	0.02	0.05	0.02	0.00	0.02	0.01
		Max diff list	0.02	0.03	0.04	0.00	0.02	0.01
		Max list	0.42	0.46	0.41	0.4	0.46	0.48
		Min list	0.26	0.55	0.45	0.37	0.35	0.38
		Subtraction	0.02	0.02	0.02	0.00	0.01	0.01
		Round and abs	0.34	0.64	0.44	0.44	0.45	0.44
		Round	0.54	0.76	0.5	0.48	0.48	0.48
		Abs	0.44	0.84	0.55	0.59	0.59	0.59
		First to upper	0.44	0.94	0.91	0.88	0.98	0.98
		First letter	0.68	0.96	0.95	0.88	0.99	0.98
		to upper	0.94	1.00	1.00	0.99	1.00	1.00
		Next first letter	0.08	0.04	0.05	0.04	0.02	0.05
		Next letter	0.68	0.85	0.67	0.64	077	0.86
		Prev first letter	0.08	0.08	0.04	0.01	0.02	0.06
		Prev letter	0.36	0.45	0.36	0.27	0.37	0.44
	Knowledge	**City to PM birth year**	0.08	0.03	0.04	0.01	0.00	0.01
		City to PM	0.12	0.13	0.04	0.01	0.08	0.11
		City to country	084	0.93	0.85	0.085	0.89	0.89
		Country to PM	0.	0.16	0.18	0.16	0.27	0.25
		Person to birth year	0.04	0.21	0.06	0.06	0.05	0.05
		Location to capital mayor	0.16	0.07	0.24	0.1	0.27	0.27
		Country to capital mayor	0.18	0.08	0.24	0.15	0.13	0.11
		Country to capital	0.22	0.57	0.52	0.87	0.9	0.9
		City to mayor	0.06	0.01	0.08	0.07	0.06	0.11
		Location to country	0.66	0.57	052	0.55	0.69	0.7

Table 6 – continued from previous page

Model	Category	Task	General Baseline	ICL	TV	Average Baseline	All TV	Pos. TV
	Linguistic	**Antonym to superlative**	-	0.18	-	-	-	-
		Antonym	0.44	0.72	0.46	0.5	0.5	0.5
		Adjective to superlative	-	0.18	0.01	-	-	-
		Past to opposite gerund	0.00	0.24	0.06	0.06	0.07	0.07
		Past to present	0.54	0.94	0.37	0.31	0.34	0.34
		Present gerund	0.08	0.92	0.08	0.13	0.15	0.14
	Multi-tasks	**Antonym to reverse**	-	-	-	-	-	-
		Reverse	0.08	0.02	0.07	0.09	0.06	0.01
		en to es	0.08	0.75	0.04	0.09	0.09	0.09
Pythia 12B	Algorithmic	**Average list**	0.00	0.07	0.03	0.04	0.05	0.04
		Sum list	-	-	-	-	-	-
		Length list	0.00	0.59	0.41	0.47	0.52	0.57
		Division	0.00	0.05	0.02	0.00	0.01	0.01
		Max diff list	0.00	0.02	0.00	0.01	0.02	0.00
		Max list	0.00	0.41	0.47	0.43	0.46	0.47
		Min list	0.00	0.47	0.26	0.18	0.25	0.27
		Subtraction	0.00	0.04	0.01	0.01	0.01	0.00
		Round and abs	0.00	0.69	0.53	0.52	0.51	0.51
		Round	0.00	0.98	0.50	0.44	0.44	0.43
		Abs	0.16	0.94	1.00	0.46	0.47	0.47
		First to upper	0.00	0.97	0.95	0.97	0.99	0.99
		First letter	0.00	0.97	0.93	0.97	1.00	1.00
		to upper	0.00	1.00	1.00	1.00	1.00	1.00
		Next first letter	0.00	0.04	0.02	0.01	0.01	0.01
		Next letter	0.00	0.76	0.98	0.92	1.00	1.00
		Prev first letter	0.00	0.05	0.01	0.08	0.05	0.05
		Prev letter	0.00	0.54	0.23	0.22	0.30	0.32
	Knowledge	**City to PM birth year**	0.00	0.04	0.01	0.00	0.00	0.05
		City to PM	0.00	0.05	0.10	0.07	0.09	0.12
		City to country	0.01	0.93	0.76	0.78	0.82	0.82
		Country to PM	0.00	0.22	0.23	0.17	0.22	0.23
		Person to birth year	0.00	0.39	0.09	0.09	0.12	0.12
		Location to capital mayor	0.00	0.13	0.15	0.13	0.18	0.30
		Country to capital mayor	0.00	0.01	0.09	0.11	0.16	0.16
		Country to capital	0.03	0.84	0.81	0.80	0.87	0.87
		City to mayor	0.00	0.10	0.10	0.04	0.04	0.20
		Location to country	0.01	0.60	0.46	0.49	0.47	0.48
	Linguistic	**Antonym to superlative**	0.00	0.16	0.00	-	-	-
		Antonym	0.04	0.76	0.39	0.39	0.41	0.42
		Adjective to superlative	0.00	0.72	0.00	-	-	-
		Past to opposite gerund	0.00	0.17	0.01	-	-	-
		Past to present	0.00	0.98	0.27	0.29	0.36	0.36
		Present gerund	0.01	0.89	0.09	0.08	0.08	0.08
	Multi-tasks	**Antonym to reverse**	-	-	-	-	-	-
		Reverse	0.00	0.02	0.06	0.03	0.10	0.10
		Antonym to Spanish	0.06	0.70	0.74	-	-	-
		en to es	0.01	0.77	0.05	0.07	0.07	0.09

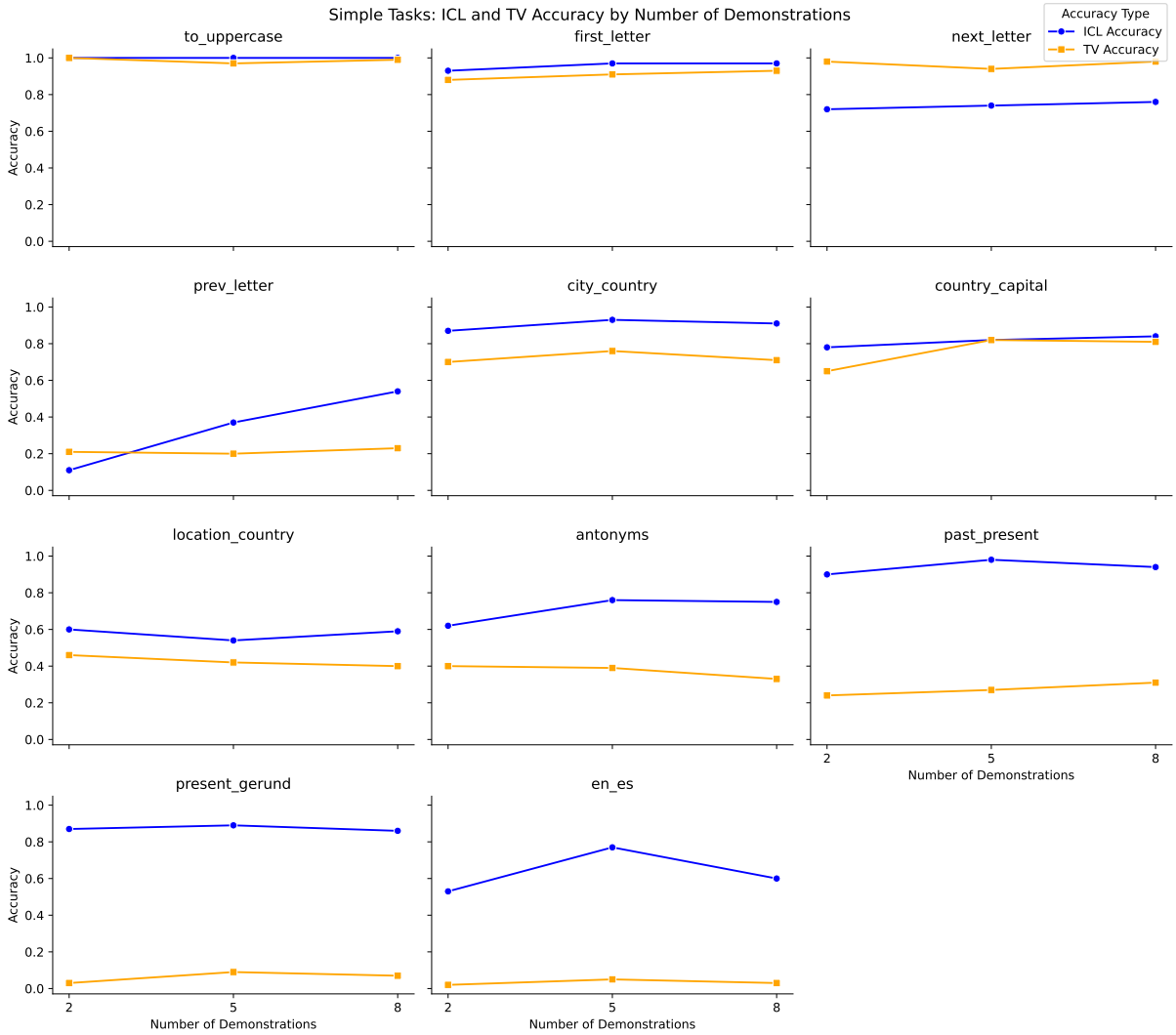
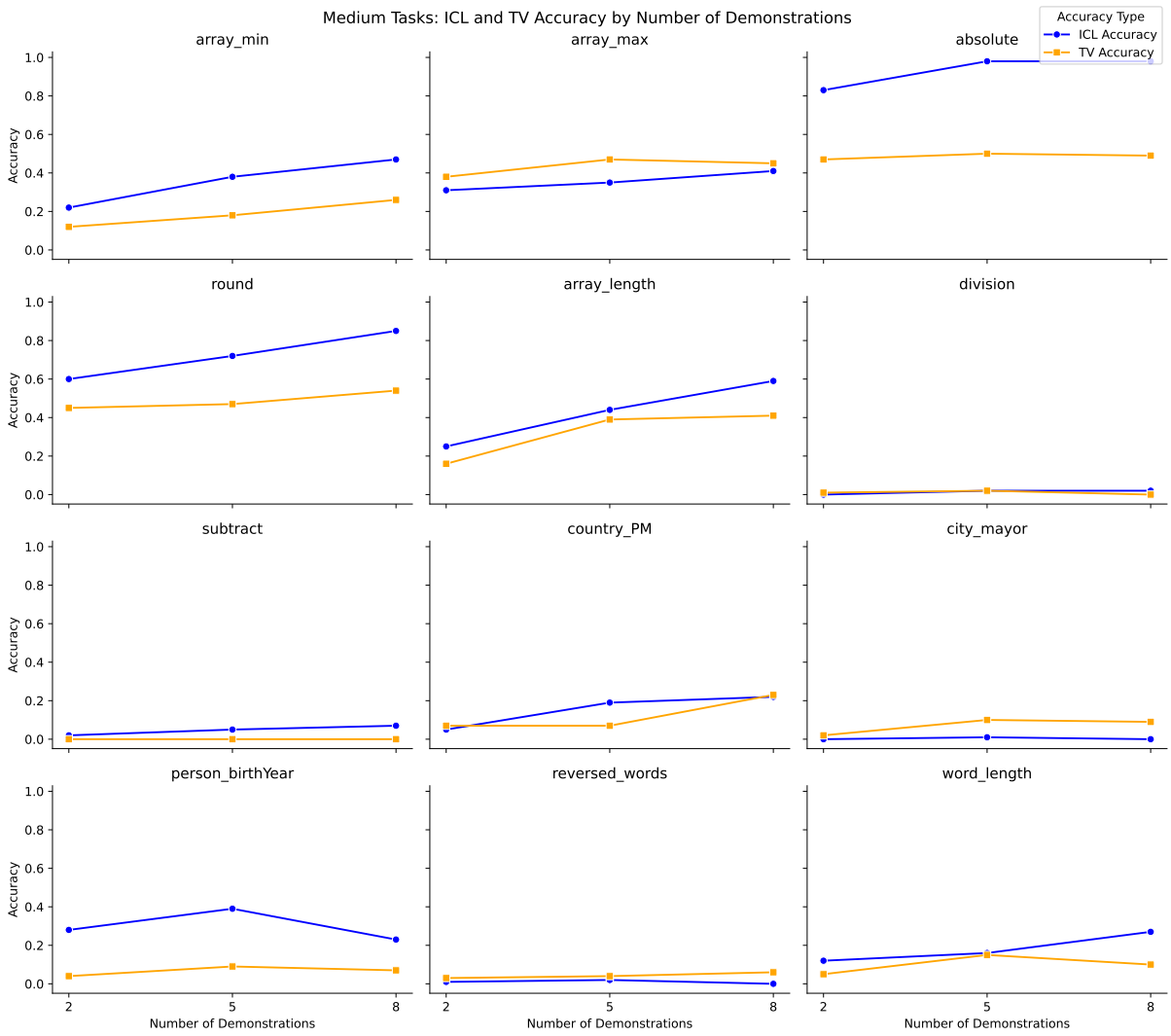
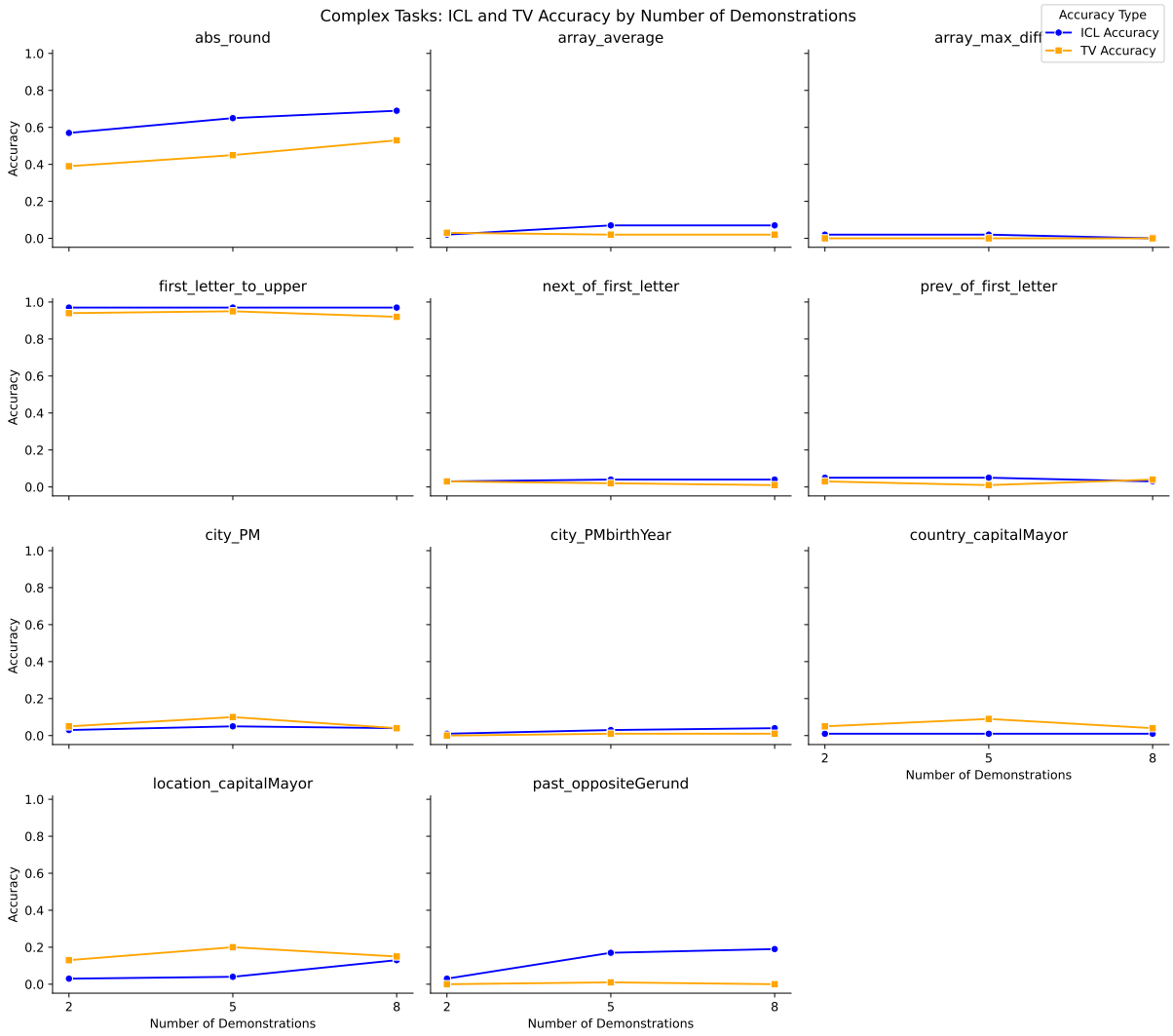


Figure 3: Correlation between accuracy of tasks and the number of demonstrations (prompt size) on Pythia 12B. The next two pages describe those results.


Medium Tasks: ICL and TV Accuracy by Number of Demonstrations
Accuracy Type
ICL Accuracy
TV Accuracy
array_min
array_max
absolute
round
array_length
division
subtract
country_PM
city_mayor
person_birthYear
reversed_words
word_length
Accuracy
Number of Demonstrations


Complex Tasks: ICL and TV Accuracy by Number of Demonstrations
Accuracy Type
ICL Accuracy
TV Accuracy
abs_round
array_average
array_max_diff
first_letter_to_upper
next_of_first_letter
prev_of_first_letter
city_PM
city_PMbirthYear
country_capitalMayor
location_capitalMayor
past_oppositeGerund
Accuracy
Number of Demonstrations
0.0
0.2
0.4
0.6
0.8
1.0
2
5
8

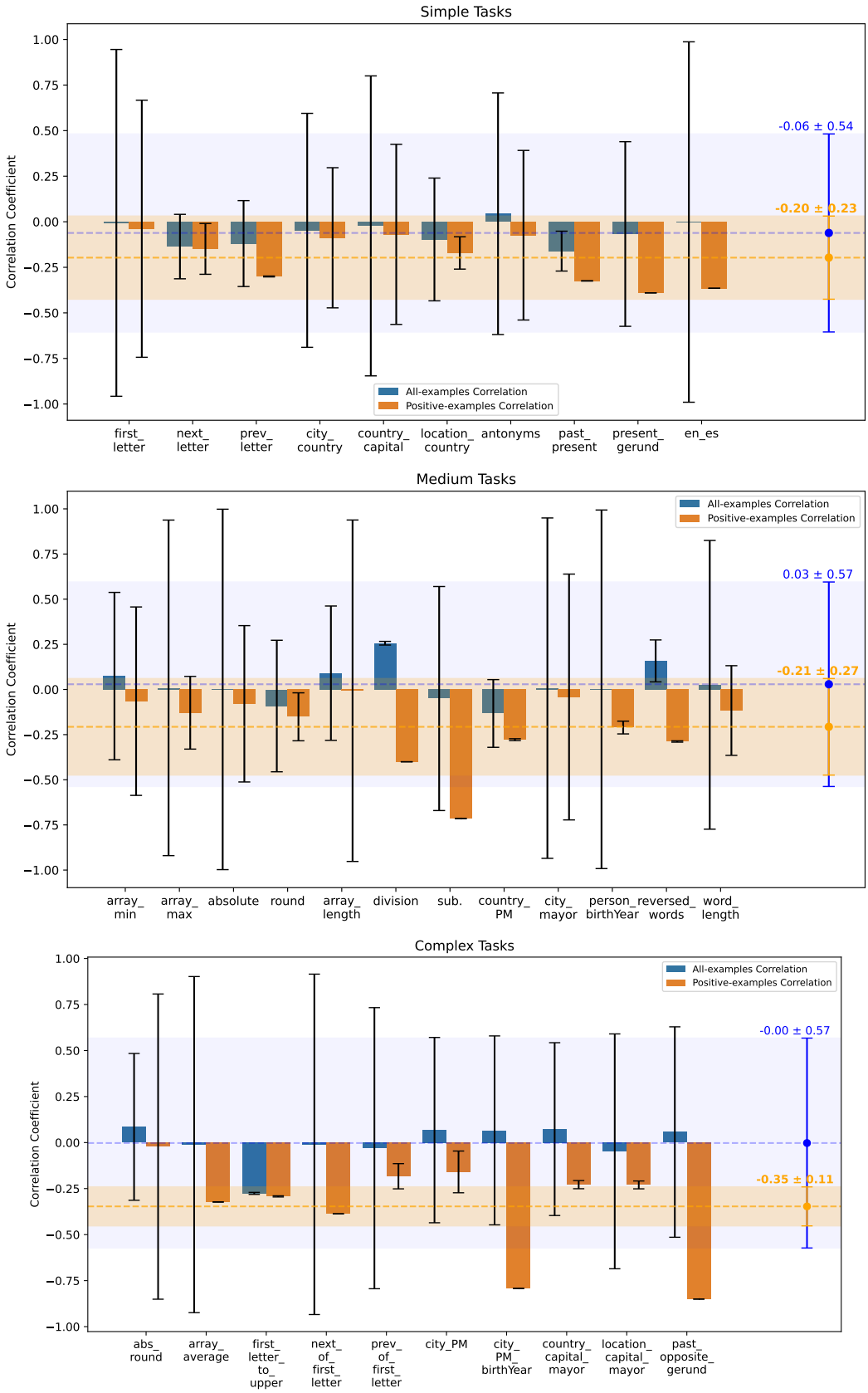


Figure 4: Correlation between accuracy of tasks and the distance of the task vector to mean task vector on Pythia 12B. The blue ones are mean over all examples and the oranges are over only correct examples.

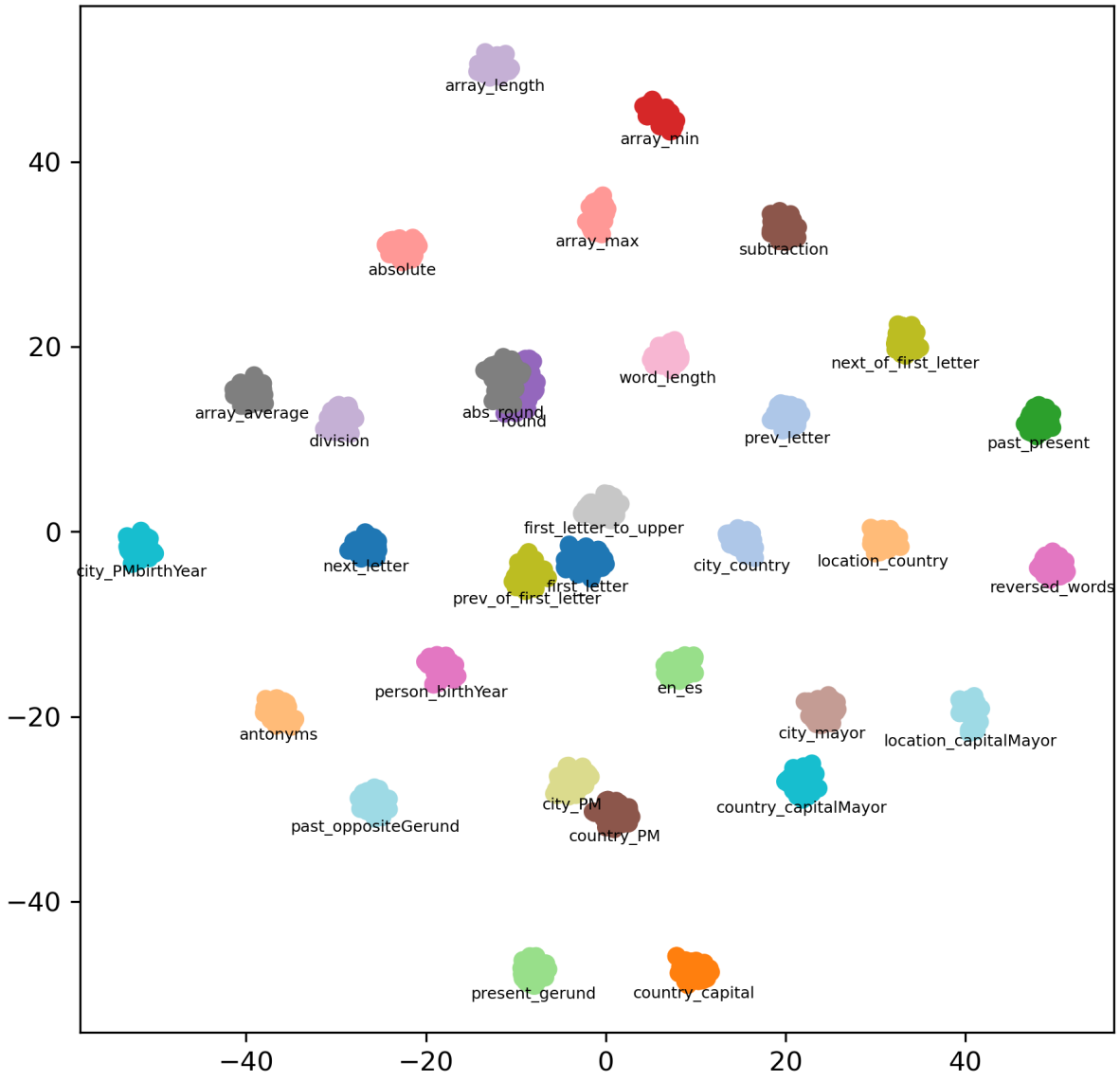


Figure 5: t-SNE graph illustrating task vector space.

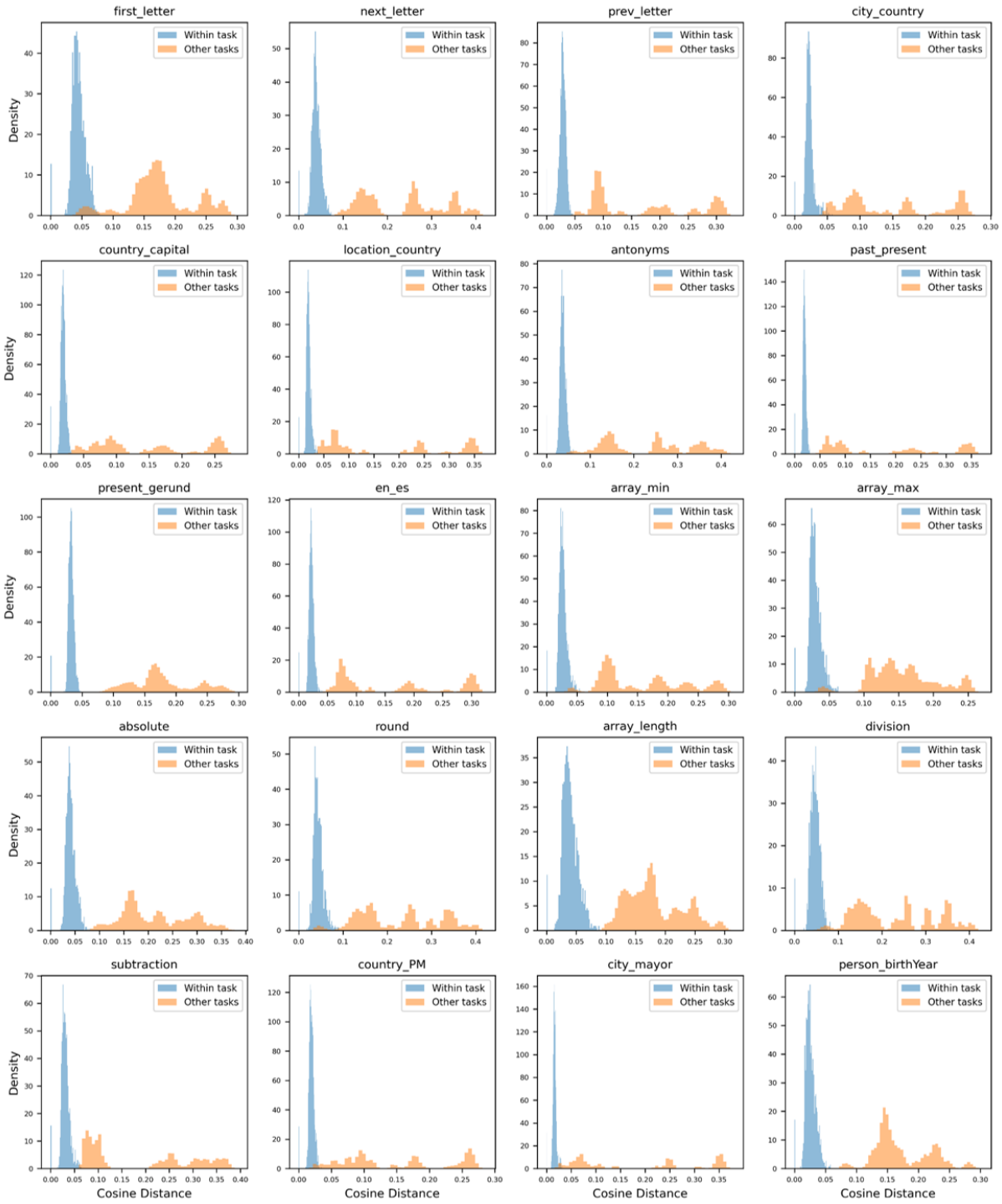
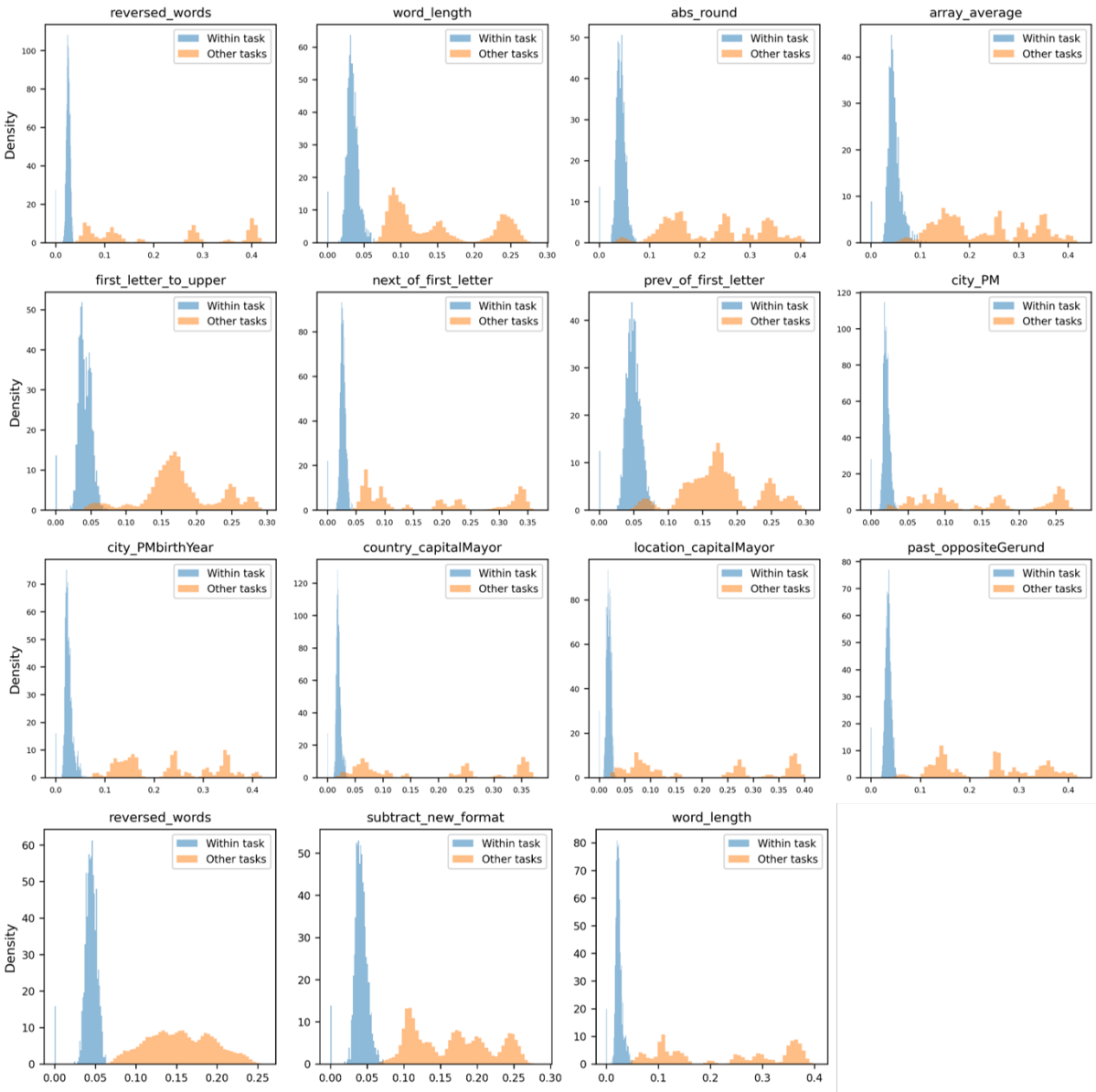


Figure 6: Task Vector Variability (on this page and the next page). In blue the distances distribution of task vectors within the same task, and in orange the distances distribution of task vectors with other tasks.


reversed_words
word_length
abs_round
array_average
first_letter_to_upper
next_of_first_letter
prev_of_first_letter
city_PM
city_PMbirthYear
country_capitalMayor
location_capitalMayor
past_oppositeGerund
reversed_words
subtract_new_format
word_length
Within task
Other tasks
Density