Symbol-LLM: Towards Foundational Symbol-centric Interface For Large Language Models

Anonymous ACL submission

Abstract

Although Large Language Models (LLMs) demonstrate remarkable ability in processing and generating human-like text, they do have limitations when it comes to comprehending and expressing world knowledge that extends beyond the boundaries of natural language(e.g., chemical molecular formula). Injecting a collection of symbolic data directly into the training of LLMs can be problematic, as it disregards the synergies among different symbolic families and overlooks the need for a balanced mixture of natural and symbolic data. In this work, we tackle these challenges from both a data and framework perspective and introduce Symbol-LLM series models[1]. First, we curated a data collection consisting of 34 tasks and incorporating approximately 20 distinct symbolic families, intending to capture the interrelations and foster synergies between symbols. Then, a two-stage tuning framework succeeds in injecting symbolic knowledge without loss of the generality ability. Extensive experiments on both symbol- and NL-centric tasks demonstrate the balanced and superior performances of Symbol-LLM series models.

1 Introduction

Large Language Models (LLMs), such as GPT-series (Radford et al., 2019; Brown et al., 2020; OpenAI, 2023) and LLaMA-series (Touvron et al., 2023a,b), boosted the performance in various Natural Language Processing (NLP) tasks (Zhao et al., 2023; Wei et al., 2022b; Zhou et al., 2023; Yao et al., 2023). The success of these models heavily relies on natural language (NL) as the primary interface[2] for interaction and reasoning. However, the NL-centric interface confines the inputs and outputs to an NL form, which can only address certain aspects of world knowledge, such as fact (Bordes et al., 2015), commonsense (Talmor et al., 2019).

Nevertheless, a substantial amount of abstract knowledge, notably in areas like molecular formula (e.g., $C_6H_{12}O_6$) and first-order logic (e.g., $\texttt{IsTriangle}(X) \rightarrow \texttt{SumOfAngles}(X, 180°)$), is more effectively represented in symbolic forms rather than in NL.

Compared to the NL form, the symbolic form covers a wide spectrum of scenarios and tends to be more concise and clear, enhancing its communication effectiveness (Gao et al., 2023; Qin et al., 2023). In particular, when interacting with robots, symbolic command sequences (such as `PICKUP`, `WALK`) are more accurate and efficient than NL. Similarly, when using programming languages (like SQL and Python) to call external tools (Gao et al., 2023), expressing this structured information in NL form can be difficult.

Despite the symbolic form offering a wealth of information, deploying LLMs directly via a symbolic-centric interface poses a significant challenge. This is largely attributed to the fact that LLMs are trained via large-scale unsupervised pre-training on extensive general text datasets, which inherently lack a symbolic foundation. The most straightforward approach to incorporating symbolic knowledge into LLMs is through fine-tuning (Yang et al., 2023; Xu et al., 2023b). However, the format of symbolic data significantly diverges from that used during pre-training. Consequently, merely fine-tuning with large heterogeneous data can lead to catastrophic forgetting (Kirkpatrick et al., 2017).

Meanwhile, existing injection methods primarily concentrate on specific symbols, it is important to note that symbolic forms can be quite complex and vary across tasks. Training an LLM for a particular symbolic form in a spe-

[1]We will open-source Symbol-LLM with 7B and 13B.

[2]*Interface* in this paper refers to the communication between LLM and environment (i.e., external tools).

cific task is both time-consuming and labor-intensive. Furthermore, treating each symbol independently often overlooks the interconnections between different symbols, e.g., the atom unit (e.g., $\texttt{BornIn}(\mathrm{Obama}, \mathrm{USA})$) in FOL is similar to function (e.g., $\texttt{query}(\texttt{Paris}, \texttt{nwr}(\texttt{hotel}))$) in API calls in the form.

Upon this observation, we conduct a comprehensive collection of 34 text-to-symbol generation tasks with ∼20 standard symbolic forms introduced with instruction tuning format. The symbolic data comes from three sources: (1) 88.3% of the data was collected from existing benchmarks. (2) 5.8% of the data was prompted by LLMs. Compensating for the natural absence of symbolic representations in some NL-centric tasks, prompting powerful LLMs can generate more novel text-to-symbol pairs. (3) 5.9% of data was generated by introducing the *Symbol-evol* strategy, with replaced symbolic definitions to prevent the model from memorizing specific symbols. The above sources finally are uniformly leveraged to capture the underlying connections between symbols from the data perspective.

From the framework aspect, we apply a two-stage continual tuning framework including the *Injection Stage* and the *Infusion Stage*. The *Injection Stage* prioritizes the exploitation of the inherent connections between different symbols, thereby enabling the model to thoroughly learn a wide range of symbolic knowledge. After tuning LLaMA-2-Chat models with all collected symbolic data, we obtain Symbol-LLM$_{\text{Base}}$ variants. The *Infusion Stage* focuses on balancing the model's dual capabilities by utilizing both symbolic data and general instruction tuning. After combining the general instruction-tuning data with the sampled symbolic data and tuning based on Symbol-LLM$_{\text{Base}}$, we can obtain Symbol-LLM$_{\text{Instruct}}$. Finally, Symbol-LLM series models are widely tested on both symbol-centric and NL-centric tasks, which are verified to exhibit substantial superiority.

Our contributions can be listed as the following:

- A comprehensive collection of text-to-symbol generation tasks is the first collection to treat symbolic data in a unified view and explore the underlying connections among symbols.
- The open-sourced Symbol-LLM series models build a new foundation LLM with balanced symbolic and NL abilities.
- Extensive experiments on both symbol- and NL-centric tasks are conducted to prove the superiority of Symbol-LLM.

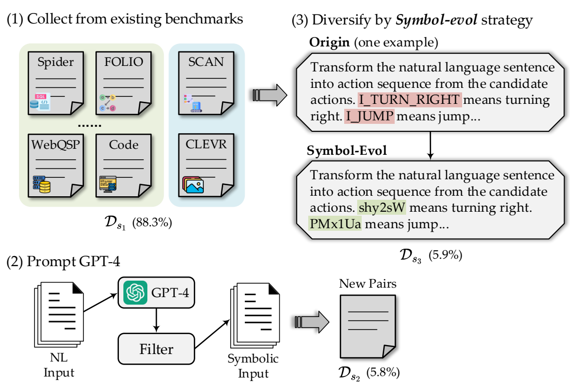


Figure 1: Overview of the data collection procedure. It involves three key sources: (1) existing benchmarks, (2) new data generated via prompting GPT-4, and (3) new data synthesized using the *Symbol-evol* strategy.

2 Approach

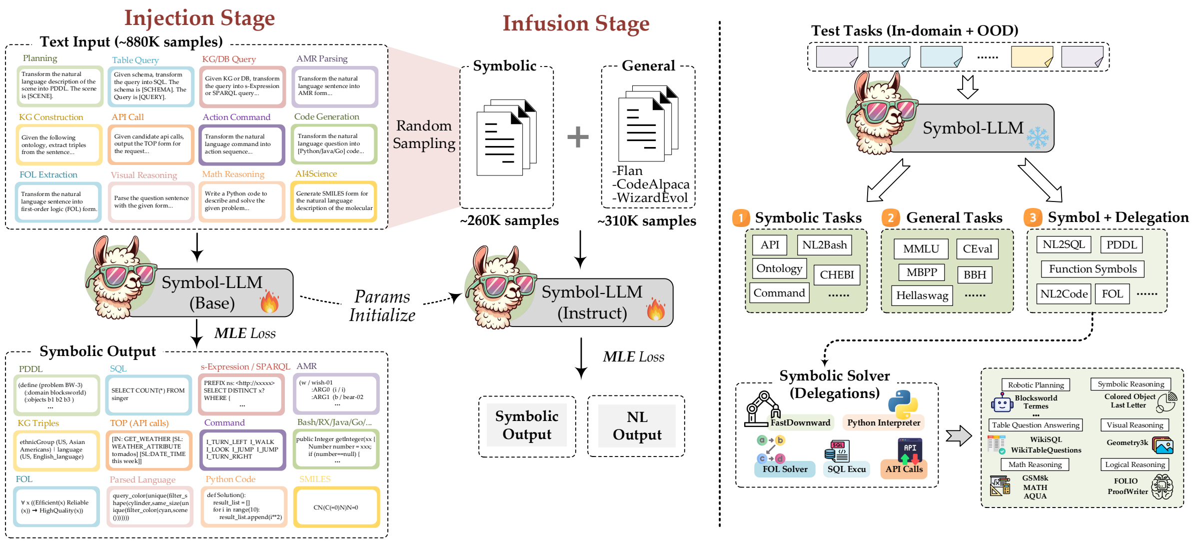
In this section, we first introduce the overall symbolic data collection procedure in Section 2.1 and then describe the two-stage tuning framework and the comprehensive test settings in Section 2.2.

2.1 Data Collection

Conducting comprehensive symbolic knowledge injection and exploiting their interrelations requires a large collection of symbolic data. However, achieving diverse knowledge coverage continues to be a significant hurdle in language modeling. Therefore, we curate an extensive collection of symbolic tasks, which is under-explored in NLP.

The overview of the symbolic data collection procedure is shown in Figure 1. The ultimate symbolic dataset is $\mathcal{D}_s = \mathcal{D}_{s_1} \cup \mathcal{D}_{s_2} \cup \mathcal{D}_{s_3}$. Here, $\mathcal{D}_{s_1}$ represents the existing benchmarks. The dataset $\mathcal{D}_{s_2}$ is a novel dataset, resulting from prompting GPT-4. $\mathcal{D}_{s_3}$ is another new dataset, generated by introducing the *Symbol-evol* strategy. Generally, we compile a set of 34 text-to-symbol generation tasks, covering ∼20 different standard symbolic forms. To maintain the general capability in NL-centric tasks, this work also includes general instruction data $\mathcal{D}_g$. Details of each dataset are attached in Appendix A.

$\mathcal{D}_{s_1}$: the existing symbolic datasets and benchmarks Previous efforts have been dedicated to specific symbolic forms, offering a natural and strong foundation for Symbol-LLM. We include



(a) Tuning Phase (b) Test Phase

Figure 2: Overall pipeline of Symbol-LLM. (a) is two-stage tuning framework, *Injection* stage and *Infusion* stage. (b) is the test phase with comprehensive settings, symbolic tasks, general tasks, and downstream tasks under the *Symbol+Delegation* paradigm.

plenty of text-to-symbol tasks from various data sources such as Spider (Yu et al., 2018), MTOP (Li et al., 2021), SCAN (Lake and Baroni, 2018), and further shape them in the defined formats. Such collection is named as $\mathcal{D}_{s_1}$.

$\mathcal{D}_{s_2}$: novel text-to-symbol pairs by prompting GPT-4 While $\mathcal{D}_{s_1}$ has broad coverage, it lacks certain text-to-symbol pairs in some crucial scenarios. For example, some mathematical problems can be better handled when converted to programming language, but labeled samples are limited. To address this, we prompt GPT-4 to generate the corresponding symbolic outputs given the NL instructions, following Gao et al. (2023). Correct outputs judged by executing solvers (e.g., code interpreter) are retained to form new text-to-symbol pairs, constructing the collection $\mathcal{D}_{s_2}$.

$\mathcal{D}_{s_3}$: new samples generated by applying *Symbol-evol* strategy The above collection can cover a vast range of standard definitions of symbolic forms. However one concern is that large tuning data with the same symbolic definitions magnify LLM's propensity to memorize the patterns instead of truly learning to follow instructions. Thus, we introduce the *Symbol-evol* strategy, expecting to enhance the diversity of symbolic systems.

The strategy of *Symbol-evol*, as depicted in Figure 1(3), is exemplified using *SCAN* dataset (Lake and Baroni, 2018). In the original data collection, some action commands (in red background) are defined to control robots. Randomly generated strings (in green background) are leveraged to replace the original symbolic definitions. For example, the originally defined command *I_TURN_RIGHT* is replaced by *shY2sW*. In this way, diverse symbol instruction samples can be derived based on some original tasks in $\mathcal{D}_{s_1}$, forming the collection $\mathcal{D}_{s_3}$.

$\mathcal{D}_g$: general data These collected data are from three sources: (i) sampled flan collection data (Wei et al., 2022a; Longpre et al., 2023); (ii) Code Alpaca instruction tuning data (Chaudhary, 2023); (iii) sampled Evol-data from WizardLM (Xu et al., 2023a). Full details are given in Appendix A.2.

2.2 Symbol-LLM

The overview of Symbol-LLM is shown in Figure 2, comprised of both the tuning and testing phases.

The tuning framework, as illustrated in Fig.2a, encompasses two stages: the *Injection* stage and *Infusion* stage. After the *Injection* stage, we can obtain the Symbol-LLM$_{\text{Base}}$ model, which is expected to address various symbol-related scenarios. However, *Injection* stage focuses on injecting symbolic knowledge into LLMs regardless of the general capability. But we also expect Symbol-LLM to maintain the necessary proficiency in general tasks, to achieve balanced symbol and NL interfaces for interaction and reasoning. Thus, we introduce the *Infusion* stage to obtain the Symbol-LLM$_{\text{Instruct}}$.

The test phase, represented in Fig.2b, covers comprehensive settings on the symbolic and NL scenarios.

Tuning Phase 1: Injection Stage In this stage, we purely focus on injecting various symbolic knowledge into LLMs by conducting supervised fine-tuning (SFT) on the $\mathcal{D}_s$ collection. The training loss of *Injection* stage is the maximum likelihood estimation (MLE):

$$\mathcal{L}_{\text{MLE}}(\mathcal{D}_s) = -\sum_i \log p_\theta(y_i|s_i \oplus x_i), \quad (1)$$

where p_θ is the tunable LLM with parameters θ, which is initialized from LLaMA-2-Chat models. $s_i \oplus x_i$ refers to the input format: the instruction (s_i) covering the task definition concatenates ($\oplus$) with the natural language query (x_i). And y_i is the symbolic output.

Tuning Phase 2: Infusion Stage In this stage, we randomly sample $\mathcal{D}_s$ to obtain a subset $\mathcal{D}_{s'} \subset \mathcal{D}_s$, the data are proportioned to ensure a fair distribution. They are combined with general instruction tuning data $\mathcal{D}_g$ to form the training set in this stage. The loss function to be minimized is based on MLE:

$$\mathcal{L}_{\text{MLE}}(\mathcal{D}_{s'} \cup \mathcal{D}_g) = -\sum_j \log p_{\theta_1}(y_j|s_j \oplus x_j), \quad (2)$$

where the tunable parameters θ_1 are initialized from Symbol-LLM$_{\text{Base}}$. s_j, x_j, and y_j are the instruction, input, and output for a single sample, respectively.

Testing Phase This work presents comprehensive testing settings for border applications. For detailed task descriptions refer to Appendix C.

- Symbolic Tasks: Extensive symbolic generation tasks stress the unique advantages of addressing symbolic language beyond NL.
- General Tasks: Classical benchmarks of general tasks are leveraged to verify the balanced capabilities in symbol- and NL-centric scenarios.
- Symbol+Delegation Tasks: Verifying the effectiveness of LLM with symbolic-centric interface. We refer to this promising setting as *Symbol+Delegation*, where the model first generates the symbolic representation of the question and then relies on the external solvers for solution (e.g., Python interpreter, SQL execution).

3 Experiments

In this section, we fully evaluate Symbol-LLM[3] on three parts of experiments: the symbolic tasks in Sec. 3.1, the general tasks in Sec. 3.2, and the Symbol+Delegation tasks in Sec. 3.3. The implementation details refer to Appendix C and Appendix D. The overall performances of Symbol-LLM are concluded in Appendix E.

3.1 Symbolic Tasks

Table 1 presents the results of 34 symbolic generation tasks. For model comparison, we include GPT-3.5, Claude-1, LLaMA-2-Chat, and the optimized model after single-domain SFT on LLaMA-2-Chat. Due to the limited space, we leave the results of other baseline models (e.g., CodeLLaMA-Instruct) in Appendix E. The main results are as follows:

Symbol-LLM largely enhances the symbol-related capabilities of LLM. In comparison with the LLaMA-2-Chat model, Symbol-LLM presents overwhelming advantages in symbolic tasks. It improves the baseline performances of 7B and 13B by 49.29% and 55.88%, respectively. Also, cutting-edge close-source LLMs like GPT-3.5 and Claude-1, are far behind Symbol-LLM, with the minimum gaps of 39.61% (GPT-3.5 v.s. Symbol-LLM-7B). In short, Symbol-LLM brings huge advantages in symbolic scenarios.

The unified modeling helps Symbol-LLM successfully capture the intrinsic relationships between different symbols. Fine-tuning LLaMA-2-Chat on single-domain tasks fully overfits domain-specific symbolic forms, as shown in *Single SFT* of Table 1. Compared with it, Symbol-LLM shows better performances, with averaged 0.42% and 2.02% gains in 7B and 13B. It verifies that the unified modeling of various symbolic forms is beneficial to capturing symbolic interrelations.

3.2 General Tasks

To verify Symbol-LLM's power in tackling NL-centric tasks, we conduct the experiments on two widely-used benchmarks, MMLU and BIG-Bench-Hard (BBH). Results are shown in Table 2.

Competitive performances in general tasks are maintained in Symbol-LLM. Overall, Symbol-LLM is well optimized with the two-stage framework in keeping general abilities. For 7B models,

[3] Unless otherwise specified, Symbol-LLM represents the final model after two stages (i.e., *Instruct* version).

Domains / Tasks		Metrics	Close-Source		Open-source (7B)			Open-source (13B)		
			GPT-3.5	Claude-1	LLaMA-2-Chat	Single SFT	Symbol-LLM	LLaMA-2-Chat	Single SFT	Symbol-LLM
Planning	Blocksworld	BLEU	96.54	91.35	85.16	97.40	**99.02**	31.27	97.06	**99.02**
	Termes	BLEU	74.73	26.94	53.08	**67.46**	48.69	59.30	68.63	**90.09**
	Floortile	BLEU	54.23	13.94	59.41	78.07	**95.84**	0.00	74.22	**95.24**
	Grippers	BLEU	99.90	90.91	86.15	94.84	**98.53**	95.36	97.46	**98.89**
SQL	Spider	EM	42.60	32.70	16.50	**65.30**	63.80	10.30	68.20	**69.20**
	Sparc	EM	29.90	28.60	12.50	**55.40**	55.00	10.20	57.50	**58.90**
	Cosql	EM	18.80	22.70	9.30	**51.30**	48.20	1.20	**54.60**	52.70
KG / DB	WebQSP	F1	36.49	41.37	0.09	**84.93**	84.43	0.00	84.80	**85.29**
	GrailQA	EM	28.52	25.56	0.00	**80.58**	79.24	0.06	**81.82**	81.17
	CompWebQ	EM	0.00	0.00	0.00	**56.30**	50.98	0.00	**59.02**	54.94
AMR	AMR3.0	Smatch	18.00	10.00	6.00	**55.00**	54.00	2.00	55.00	55.00
	AMR2.0	Smatch	14.00	12.00	7.00	**46.00**	45.00	1.00	**47.00**	46.00
	BioAMR	Smatch	23.00	3.00	24.00	**80.00**	78.00	0.00	80.00	80.00
Ontology	Tekgen	F1	8.92	1.86	4.50	56.69	**57.34**	6.24	58.49	**58.55**
	Webnlg	F1	28.34	8.89	7.38	**63.75**	60.42	17.23	62.13	**63.08**
API	MTOP	EM	3.80	8.40	0.00	**84.80**	84.40	0.00	86.20	**86.60**
	TOPv2	EM	6.60	7.60	0.00	**86.60**	85.80	0.00	**87.20**	85.20
	NLmaps	EM	30.88	16.77	2.00	91.95	**92.18**	3.60	**92.38**	92.21
Command	SCAN	EM	15.09	15.97	0.00	98.23	**98.35**	0.00	98.99	**99.28**
Code	NL2BASH	BLEU	54.19	42.24	23.29	59.22	**60.25**	19.06	60.68	**60.76**
	NL2RX	BLEU	38.60	18.30	5.91	**85.25**	85.08	0.00	**85.55**	84.97
	NL2Python	BLEU	37.01	36.73	26.68	38.19	**39.79**	34.94	40.35	**40.76**
	NL2Java	BLEU	24.88	22.79	25.77	27.33	**28.08**	23.49	**28.47**	28.25
	NL2Go	BLEU	19.08	26.65	24.00	**30.77**	29.19	1.26	24.75	**30.31**
FOL	FOLIO	LE	60.65	53.47	33.98	**90.81**	90.58	28.79	**91.59**	90.65
	MALLS	LE	69.15	30.46	55.13	**89.24**	88.88	11.71	89.41	**89.50**
	LogicNLI	LE	73.11	69.16	39.95	**100.00**	99.97	32.26	99.99	**100.00**
Visual	GQA	EM	7.55	7.70	0.30	**85.65**	85.50	8.85	**86.10**	85.95
	CLEVR	EM	6.35	5.90	0.25	86.35	**94.80**	1.15	92.20	**95.60**
	Geometry3k	EM	65.25	40.84	36.88	93.92	**95.13**	52.17	94.52	**95.67**
Math	GSM8K-Code	BLEU	82.20	63.42	53.66	**85.31**	84.14	72.29	84.01	**84.42**
	AQUA-Code	BLEU	67.48	48.88	39.25	66.27	**67.05**	55.13	65.66	**67.20**
	MATH-Code	BLEU	56.48	48.87	29.88	56.43	**57.36**	48.85	**58.24**	56.97
AI4Science	CheBi-20	EM	1.15	0.30	0.00	40.36	**58.97**	0.00	46.82	**65.27**
Average Performance			32.27	25.04	22.59	71.46	**71.88**	18.46	72.32	**74.34**

Table 1: Main results on 34 text-to-symbol generation tasks. The better results with the same model size are marked in bold. *GPT-3.5*, *Claude-1*, and *LLaMA-2-Chat* column presents the baseline performances of prompting these models under the few-shot setting. *Single-SFT* represents the models fine-tuned with single-domain samples based on LLaMA-2-Chat. *Symbol-LLM* column represents the final obtained model after two-stage tuning.

Symbol-LLM$_{\text{Instruct}}$ shows consistent superiority on MMLU and BBH benchmarks, with ∼4% gains compared with LLaMA-2-Chat. For 13B models, although Symbol-LLM$_{\text{Instruct}}$ slightly falls behind its LLaMA counterpart, it achieves 7.20% performance advantages in BBH. The superiority on average is obvious. While Symbol-LLM may not yet match the performance of closed-source LLMs, its well-rounded general capability is notable.

To verify the generalization in a broader scope, the evaluation of extensive general tasks is attached in Appendix F.

3.3 Symbol+Delegation Tasks

A wide range of experiments are done under the *Symbol+Delegation* paradigm, covering the fields of math reasoning, symbolic reasoning, logical reasoning, robotic planning, visual reasoning as well as table question answering. For detailed settings, please refer to Appendix C.3. Limited by space, we only present the results of the math reasoning in the main paper. The remaining parts are attached in Appendix G.

We select 9 commonly used math datasets for testing, including both in-domain and OOD tasks. To demonstrate the surprising performances of Symbol-LLM, we also include several math-domain LLMs (e.g., WizardMath (Luo et al., 2023), MAmmoTH (Yue et al., 2023)) as strong baselines. Comparison results are presented in Table 3.

Advanced abilities in math reasoning are possessed by Symbol-LLM. GSM8K and MATH are widely used to evaluate the math reasoning capabilities of LLMs. Compared with recent math-domain LLMs, Symbol-LLM presents great com-

Models	MMLU (5-shot) Humanities	SocialSciences	STEM	Others	Average	BBH (0-shot) Average
Close-source LLMs						
GPT-3.5	54.90	69.58	49.73	66.75	59.74	56.84
Claude-1	56.60	74.15	53.66	60.35	62.09	47.01
Open-source LLMs (7B)						
LLaMA-2-Chat	42.47	52.49	36.94	52.47	45.78	35.01
CodeLLaMA-Instruct	39.47	46.31	35.95	45.34	41.57	35.69
Symbol-LLM$_{\text{Base}}$	40.04	46.28	33.73	47.16	41.70	33.82
Symbol-LLM$_{\text{Instruct}}$	**46.33**	**57.20**	**40.39**	**54.53**	**49.30**	**39.30**
Open-source LLMs (13B)						
LLaMA-2-Chat	**49.52**	**62.43**	**43.84**	**60.02**	**53.55**	36.99
CodeLLaMA-Instruct	33.88	41.92	34.69	42.17	37.73	36.71
Symbol-LLM$_{\text{Base}}$	45.67	55.67	40.09	53.89	48.56	35.26
Symbol-LLM$_{\text{Instruct}}$	48.88	62.14	43.44	57.93	52.71	**44.09**

Table 2: Results on general tasks. We include 57 tasks in the MMLU benchmark for testing under the 5-shot setting (Hendrycks et al., 2021a), while we select 21 tasks in BBH under the 0-shot setting following Gao et al. (2021a). The best results are marked in bold while sub-optimal results are underlined (same for the following tables).

Models	Del.	GSM8k	MATH	GSM-Hard	SVAMP	ASDiv	ADDSUB	SingleEQ	SingleOP	MultiArith
Is OOD Setting		✗	✗	✓	✓	✓	✓	✓	✓	✓
Close-source LLMs										
GPT-3.5	✓	4.60	1.05	4.62	5.10	6.30	1.01	3.94	8.54	17.33
GPT-3.5 (3-shot)	✓	76.04	36.80	62.09	83.40	85.73	87.59	96.46	90.74	96.67
Claude-1	✓	11.14	1.07	9.02	10.30	6.30	5.06	4.53	0.36	12.67
Claude-1 (3-shot)	✓	58.07	13.17	43.75	78.90	74.19	79.49	88.19	87.72	91.83
Open-source LLMs (7B)										
LLaMA-2-Chat (3-shot)	✓	12.21	1.32	10.69	22.00	25.86	29.11	27.36	39.15	23.17
CodeLLaMA-Instruct (3-shot)†	✓	34.00	16.60	33.60	59.00	61.40		Average performance 79.60		
WizardMath†	✗	54.90	10.70	-	57.30	-	-	-	-	-
MAmmoTH†	✓	51.70	**31.20**	-	66.70	-	-	-	-	-
Symbol-LLM$_{\text{Base}}$	✓	**61.14**	28.24	**52.62**	72.50	**78.34**	**89.62**	**97.83**	**96.26**	**99.67**
Symbol-LLM$_{\text{Instruct}}$	✓	59.36	26.54	48.98	**72.80**	75.76	87.85	96.26	93.24	99.00
Open-source LLMs (13B)										
LLaMA-2-Chat (3-shot)	✓	34.87	6.07	28.96	45.00	46.61	45.57	47.05	56.76	56.67
CodeLLaMA-Instruct (3-shot)†	✓	39.90	19.90	39.00	62.40	65.30		Average performance 86.00		
WizardMath†	✗	63.90	14.00	-	64.30	-	-	-	-	-
MAmmoTH†	✓	61.70	**36.00**	-	72.40	-	-	-	-	-
Symbol-LLM$_{\text{Base}}$	✓	**68.69**	33.39	**58.53**	**78.80**	**80.15**	91.14	**96.85**	**95.55**	98.83
Symbol-LLM$_{\text{Instruct}}$	✓	65.58	31.32	55.57	76.80	79.01	**91.90**	**96.85**	94.84	**99.33**

Table 3: Results on Math Reasoning. Del. represents whether uses delegation (i.e., Python Interpreter for math reasoning tasks). Results are under the zero-shot setting unless otherwise stated (the following tables share the same setting). † indicates that the results are reported from Luo et al. (2023), Yue et al. (2023) and Gou et al. (2023).

petitive results on them. Especially on GSM8K, Symbol-LLM consistently wins all strong baselines with great margins with all the model variants. On the MATH dataset, Symbol-LLM merely falls behind MAmmoTH, which is a strong LLM specially designed for math reasoning tasks. Notably, MAmmoTH includes GSM8K and MATH in the tuning stage and it also uses delegation (i.e., Python Interpreter) for inference, thus our comparisons are fair. Similar superiority is also observed under the OOD tasks (e.g., SVAMP).

Symbol-LLM exhibits competitive performances in extrapolating to OOD tasks. More surprisingly, Symbol-LLM consistently presents its significant superiority among all 7 OOD math tasks. Even compared with GPT-3.5 under the three-shot setting, our Symbol-LLM-7B series won 4 (out of 7) OOD tasks under the zero-shot setting. As we scale the model size to 13B, obvious performance improvements are observed in most of the tasks. These findings verify the prospects of Symbol-LLM under the Symbol+Delegation paradigm.

4 Analysis

In this section, we include the ablation studies (Sec.4.1) and the analysis on *Alignment* and *Uniformity* (Sec.4.2). Notably, additional supplementary experiments are attached in Appendix H.

4.1 Ablation Studies

Here we present two ablation experiments from both the framework and data views: (1) tuning only in one stage, and (2) tuning only on general data collection. For a fair comparison, we introduce two settings for one-stage tuning. The first setting (named *One-stage 1.46M*) simply mixes $\mathcal{D}_s$, $\mathcal{D}_{s'}$ and $\mathcal{D}_g$, regardless of sample overlap. The second setting (named *One-stage 1.20M*) mixes $\mathcal{D}_s$ and $\mathcal{D}_g$, which ensures consistency in diversity and avoids duplication. The model exclusively fine-tuned on general task $\mathcal{D}_g$ is referred to as *General-only*. Comparison results are shown in Table 4.

Two-stage tuning framework shows superiority over one-stage, especially for 13B. Simply mixing the training data in one stage is prone to affecting the symbol-related tasks. Especially under the *Symbol+Delegation* setting, the two-stage framework witnesses 3∼6% advantages over the one-stage models. In the 13B model comparison, our two-stage framework consistently demonstrates superiority across symbolic tasks, general tasks, and *Symbol+Delegation* tasks.

The incorporation of symbolic data yields a modest impact on the performances of general tasks. Compared with *General-only*, Symbol-LLM$_{\text{Instruct}}$ is optimized to largely enhance the symbol-centric capabilities. Meanwhile, it maintains the capability to address general NL-centric tasks without significant sacrifices (< 2%).

4.2 Alignment and Uniformity

Motivated by (Wang and Isola, 2020; Gao et al., 2021b), we include *Alignment* and *Uniformity* metrics to delve into the factors contributing to the superiority of Symbol-LLM.

Alignment measures the representation similarity within each symbolic form, based on Eq. 3 in Appendix I. *Uniformity* quantifies the uniformity of all the symbolic representations with Eq. 4 in Appendix I. The calculation results are visualized in Figure 3. Further, we extend the definition to measure the interrelations between any two symbolic forms, based on Eq. 5. Limited by space, we only include a part of the symbolic forms for illustration and present the results of 13B models in Figure 4. Detailed definitions and settings are attached in Appendix I.

The item-wise conclusions are listed as follows:

Symbol-LLM optimizes symbol distinctiveness and overall expressiveness in the embedding space. From Fig. 3, compared with the LLaMA-2-Chat models, Symbol-LLM series is optimized towards superior *Alignment* and *Uniformity*. It ensures the discernment of shared features within each symbolic form, simultaneously enhancing the overall information entropy. Specifically for the 7B model, the two-stage framework effectively maintains a balance of uniformity, preventing the collapse of the embedding space.

Symbol-LLM excels at capturing symbolic interrelations. From Fig. 4, the LLaMA-2-Chat model exhibits significant representation sparsity between symbolic forms. Even under the same form (e.g., *Bash*, *FOL*), the features are scattered. On the contrary, Symbol-LLM largely enhances the perception of symbolic interrelations by (1) achieving better alignments between symbols (e.g., *Python-AMR* and *CheBi-RX*) and (2) pulling closer sample features within each symbolic form (e.g., FOL).

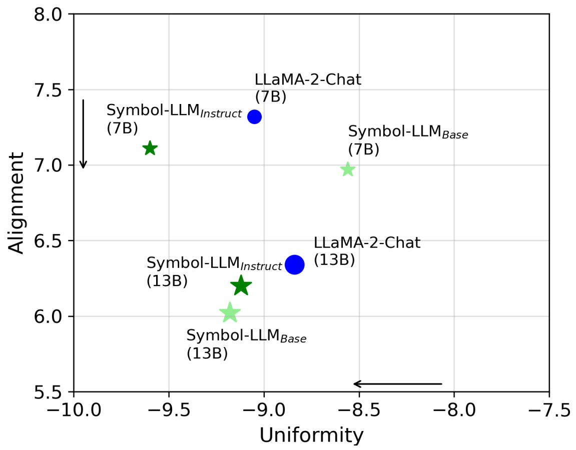
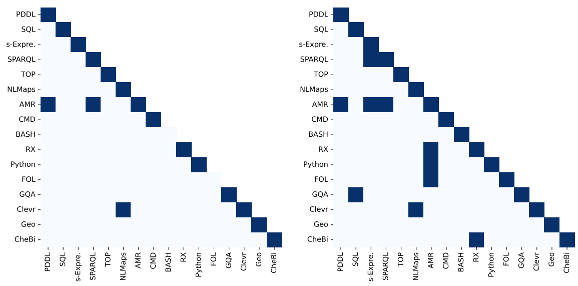


Figure 3: Visualization of *Alignment-Uniformity*. Both metrics are inversely related, which means a lower value indicates better performance.



(a) LLaMA-2-Chat (13B) (b) Symbol-LLM (13B)

Figure 4: Visualization of the alignment relations between symbols after binarization. Dark blue denotes a close relation between two symbols in the representation. Limited by space, we only showcase 13B models. More illustrations refer to Appendix I.

Models	7B Models Symbolic	General	Symbol+Del.	Avg.	13B Models Symbolic	General	Symbol+Del.	Avg.
Symbol-LLM	71.88	44.30	52.54	56.24	74.34	48.40	60.45	61.06
One-stage 1.20M	70.38	45.24	47.27	54.30	70.59	48.29	53.99	57.62
Δ	(+1.50)	(-0.94)	(+5.27)	(+1.94)	(+3.75)	(+0.11)	(+6.46)	(+3.44)
One-stage 1.46M	72.75	44.44	49.31	55.50	73.71	46.59	52.67	57.66
Δ	(-0.87)	(-0.14)	(+3.13)	(+0.74)	(+0.63)	(+1.81)	(+7.78)	(+3.40)
General-only	28.66	46.21	28.17	34.35	31.35	49.72	31.49	37.52
Δ	(+43.22)	(-1.91)	(+24.37)	(+11.89)	(+42.99)	(-1.32)	(+28.96)	(+23.54)

Table 4: Comparison experiments. *Avg.* denotes the simple averaged performances on the symbolic tasks, general tasks, and Symbol+Delegation tasks.

5 Related Works

Large Language Models Plenty of recent efforts have been made to develop foundation language models (Zhao et al., 2023), which are expected to promote the subsequent applications, such as AI agents (Wang et al., 2023a). These works on LLMs are universally categorized into closed-source and open-source models. Close-source LLMs, represented by GPT-4 (OpenAI, 2023), Claude, PaLM (Chowdhery et al., 2023), have greatly shaped our daily life through NL-centric interactions. However, their closed-source and black-box property limits further optimization. Under such circumstances, open-source LLMs (Zeng et al., 2023; Jiang et al., 2023; Touvron et al., 2023b) receive significant attention because of their tunable and small-scale properties. However, current attempts on these LLMs mainly explore NL-centric abilities, which treats NL as the interface to express knowledge and achieve interactive reasoning. In contrast, our work focuses on improving the symbol-centric capabilities of open-source LLM, which leads to a balanced symbol-centric and NL-centric foundational LLM.

Instruction Tuning To make LLMs capable of following human instructions, instruction fine-tuning (Zhang et al., 2023) is widely adopted. Meanwhile, self-instruct methods (Wang et al., 2023c; Xu et al., 2023a; Ouyang et al., 2022) have been proposed to generate diverse and abundant instruction data, based on a small collection of seed instructions. In our work, we follow the previous instruction tuning strategies in both tuning stages. For symbolic tasks, we construct instructions, covering the task and symbolic descriptions. For general tasks, we sample the off-the-shelf instruction-tuning datasets (e.g., Flan collection (Longpre et al., 2023)).

Symbol-centric Scenarios LLMs have dominated plenty of NL-centric tasks (Rajpurkar et al., 2016; Talmor et al., 2019; Nallapati et al., 2016), where NL is leveraged as the core interface for interaction, planning, and reasoning. But world knowledge is not purely represented by NL. In fact, symbolic language is also of great significance in expressing abstract world knowledge (Edwards et al., 2022; Bevilacqua et al., 2021; Li and Srikumar, 2019) and leveraging external tools (Gao et al., 2023; Liu et al., 2023; Pan et al., 2023). Some concurrent works (Xu et al., 2023b; Yang et al., 2023) shift focus to the specific forms of symbols (e.g., code), either through prompting off-the-shelf LLMs or tuning on open-source LLMs. These efforts fail to lay a solid symbolic foundation, which is expected to grasp the interrelations among various symbolic forms. In our work, we explore the possibility of treating symbols in a unified manner and lay foundations to build balanced symbol and NL interfaces.

6 Conclusion

This work proposes to enhance the LLM capability in symbol-centric tasks while preserving the performances on general tasks, leading to balanced symbol and NL interfaces. To address the challenges of capturing symbol interrelations and maintaining a balance in general abilities, we tackle the problem from both data and framework perspectives. Data-wise, we include a collection of 34 text-to-symbol tasks to systematically explore underlying symbol relations. Framework-wise, we implement SFT in a two-stage manner to reduce catastrophic forgetting. Extensive experiments across three task settings (i.e., symbolic tasks, general tasks, and symbol+delegation tasks) demonstrate Symbol-LLM's superiority in harmonizing symbol- and NL-centric capabilities. Moreover, all models and resources will be made public to facilitate a broader range of research.

Limitations

The insight of Symbol-LLM is to build a balanced symbol- and NL-centric interface for interaction and reasoning. We achieve it from both data (comprehensive symbolic collection to open-source) and framework (two-stage tuning to reduce forgetting) perspectives. It is expected to expand the scope of cutting-edge open-source LLMs largely and lay a new foundation for future work. Though plenty of experiments covering three settings are conducted, there still exist the following two directions for exploration: (1) The model's ability to self-correct or interact with environmental feedback in symbolic scenarios. It is also key to building language agents from language models. (2) Model size scaling to 70B or larger. As widely recognized, 7B or 13B LLMs are still not sufficient to build excellent language agents, especially when complex interaction is involved. Thus, it needs further exploration for the size scaling to the larger ones.

References

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2021. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 3554–3565.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*, pages 178–186.

Michele Bevilacqua, Rexhina Blloshmi, and Roberto Navigli. 2021. One SPRING to rule them both: Symmetric AMR semantic parsing and generation without a complex pipeline. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*, pages 12564–12573.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Sahil Chaudhary. 2023. Code alpaca: An instruction-following llama model for code generation. `https://github.com/sahil280114/codealpaca`.

Xilun Chen, Asish Ghoshal, Yashar Mehdad, Luke Zettlemoyer, and Sonal Gupta. 2020. Low-resource domain adaptation for compositional task-oriented semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5090–5100.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24:240:1–240:113.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. `https://github.com/open-compass/opencompass`.

Carl Edwards, Tuan Lai, Kevin Ros, Garrett Honke, Kyunghyun Cho, and Heng Ji. 2022. Translation between molecules and natural language. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 375–413.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2021a. A framework for few-shot language model evaluation.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: program-aided language models. In *International Conference on Machine Learning (ICML)*, volume 202, pages 10764–10799. PMLR.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021b. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 179–188.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Minlie Huang, Nan Duan, Weizhu Chen, et al. 2023. Tora: A tool-integrated reasoning agent for mathematical problem solving. *arXiv preprint arXiv:2309.17452*.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Luke Benson, Lucy Sun, Ekaterina Zubova, Yujie Qiao, Matthew Burtell, David Peng, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Shafiq Joty, Alexander R. Fabbri, Wojciech Kryscinski, Xi Victoria Lin, Caiming Xiong, and Dragomir Radev. 2022. Folio: Natural language reasoning with first-order logic. *CoRR*, abs/2209.00840.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems (NeurIPS)*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533.

Drew A. Hudson and Christopher D. Manning. 2019. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6700–6709.

Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. 2019. Codesearchnet challenge: Evaluating the state of semantic code search. *CoRR*, abs/1909.09436.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. 2017. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1988–1997.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Kevin Knight and et al. 2017. Abstract meaning representation (amr) annotation release 2.0. Web Download. LDC2017T10.

Kevin Knight and et al. 2020. Abstract meaning representation (amr) annotation release 3.0. Web Download. LDC2020T02.

Brenden M. Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, pages 2879–2888.

Carolin Lawrence and Stefan Riezler. 2018. Improving a neural semantic parser by counterfactual learning from human bandit feedback. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1820–1830.

Haoran Li, Abhinav Arora, Shuohui Chen, Anchit Gupta, Sonal Gupta, and Yashar Mehdad. 2021. MTOP: A comprehensive multilingual task-oriented semantic parsing benchmark. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 2950–2962.

Tao Li and Vivek Srikumar. 2019. Augmenting neural networks with first-order logic. In *Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL)*, pages 292–302.

Xi Victoria Lin, Chenglong Wang, Luke Zettlemoyer, and Michael D. Ernst. 2018. Nl2bash: A corpus and semantic parser for natural language interface to the linux operating system. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 158–167.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. 2023. LLM+P: empowering large language models with optimal planning proficiency. *CoRR*, abs/2304.11477.

Nicholas Locascio, Karthik Narasimhan, Eduardo DeLeon, Nate Kushman, and Regina Barzilay. 2016. Neural generation of regular expressions from natural language with minimal domain knowledge. In *Proceedings of the 2016 Conference on Empirical*

Methods in Natural Language Processing (EMNLP), pages 1918–1923.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning (ICML)*, volume 202, pages 22631–22648. PMLR.

Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu. 2021. Inter-gps: Interpretable geometry problem solving with formal language and symbolic reasoning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL/IJCNLP)*, pages 6774–6786.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 975–984.

Nandana Mihindukulasooriya, Sanju Tiwari, Carlos F Enguix, and Kusum Lata. 2023. Text2kgbench: A benchmark for ontology-driven knowledge graph generation from text. In *International Semantic Web Conference (ISWC)*, volume 14266, pages 247–265.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*, pages 280–290.

OpenAI. 2023. Gpt-4 technical report. *CoRR*, abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. 2023. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *CoRR*, abs/2305.12295.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2080–2094.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *CoRR*, abs/2307.16789.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2383–2392.

Anna Rogers, Matt Gardner, and Isabelle Augenstein. 2023. QA dataset explosion: A taxonomy of NLP resources for question answering and reading comprehension. *ACM Computing Surveys*, 55(10):197:1–197:45.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1743–1752.

Subhro Roy, Tim Vieira, and Dan Roth. 2015. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3:1–13.

Abulhair Saparov and He He. 2022. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International Conference on Learning Representations (ICLR)*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics*, pages 13003–13051.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics*, pages 3621–3634.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 641–651.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4149–4158.

Jidong Tian, Yitian Li, Wenqing Chen, Liqiang Xiao, Hao He, and Yaohui Jin. 2021. Diagnosing the first-order logical reasoning ability through logicnli. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3738–3747.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2023a. A survey on large language model based autonomous agents. *CoRR*, abs/2308.11432.

Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning (ICML)*, pages 9929–9939. PMLR.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, et al. 2023b. How far can camels go? exploring the state of instruction tuning on open resources. *CoRR*, abs/2306.04751.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023c. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 13484–13508.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2022a. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations (ICLR)*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022b. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 24824–24837.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I Wang, et al. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 602–631.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023a. Wizardlm: Empowering large language models to follow complex instructions. *CoRR*, abs/2304.12244.

Yiheng Xu, Hongjin Su, Chen Xing, Boyu Mi, Qian Liu, Weijia Shi, Binyuan Hui, Fan Zhou, Yitao Liu, Tianbao Xie, et al. 2023b. Lemur: Harmonizing natural language and code for language agents. *CoRR*, abs/2310.06830.

Yuan Yang, Siheng Xiong, Ali Payani, Ehsan Shareghi, and Faramarz Fekri. 2023. Harnessing the power of large language models for natural language to first-order logic translation. *CoRR*, abs/2305.15541.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *CoRR*, abs/2305.10601.

Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander R. Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter S. Lasecki, and Dragomir R. Radev. 2019a. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. 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 1962–1979.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3911–3921.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Jonathan Kraft, Vincent Zhang, Caiming Xiong, Richard Socher, and Dragomir R. Radev. 2019b. Sparc: Cross-domain semantic parsing in context. In *Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL)*, pages 4511–4523.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2023. GLM-130B: an open bilingual pre-trained model. In *The Eleventh International Conference on Learning Representations (ICLR)*.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023. Instruction tuning for large language models: A survey. *CoRR*, abs/2308.10792.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *CoRR*, abs/2303.18223.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations (ICLR)*.

A Details of Data Collection

In this section, detailed information on the data collection is attached, including both text-to-symbol task collection, and general task collection.

A.1 Text-to-symbol Task Collection

We provide a detailed illustration of the symbolic task collection, which consists of 34 different text-to-symbol generation tasks. They are categorized into 12 domains in Table 5.

Note that the symbolic task collection includes but is not limited to the listed 34 tasks. To expand the diversity, we also consider some similar tasks. For example, we include some domain-specific NL-to-SQL tasks to provide diverse schema. The data is only used at the tuning stage but is not for a test. Thus, the whole collection (only count training samples) reaches ∼880K samples. All of them are leveraged in the first SFT stage.

Also, it is mentioned above that we sample parts of symbolic task collection in the second stage to reduce forgetting. For it, we uniformly sample each task domain with a ratio of 0.3, leading to a sampled collection of ∼260K.

A.2 General Task Collection

In the second tuning stage, we include a collection of general instruction-tuning data to keep the LLM capability in some NL-centric settings and further improve the instruction-following capability of Symbol-LLM.

The general data collection contains $\sim 570K$ samples, which are sourced from the following three parts:

(1) **Sampled Flan collection** (Longpre et al., 2023) of 150K samples. We obtain the collection directly following Tulu (Wang et al., 2023b).

(2) **Code Alpaca collection** (Chaudhary, 2023) of 20K samples. In fact, this collection is not in an NL-to-NL form as we expected. However, it stresses much on the instruction-following capabilities, which may help enhance the general ability of LLMs. Also, it is expected to act as the bridge between NL data and symbolic form (i.e., code in this case).

(3) **Sampled WizardLM collection** (Xu et al., 2023a) of 143K samples. To further expand the diversity of our instruction-tuning collection, we leverage the evol-data from WizardLM.

B Data Format

To support the instruction tuning, each piece of data i in the training collection contains three parts, i.e., instruction s_i, input x_i, and output y_i. During the training process, instruction s_i and input text x_i are concatenated as the whole input sequence. The model is optimized to generate output y_i. One example in the *FOLIO* dataset is as follows:

[Instruction] Transform the natural language sentence into first-order logic forms.

[Input] All people who regularly drink coffee are dependent on caffeine.

[Output] ∀x (Drinks(x) → Dependent(x))

In the implementation, we rewrite the instruction for each sample by prompting GPT-4, keeping the diversity of the instruction.

C Test Datasets and Benchmarks

Our main experiments are conducted on both text-to-symbol tasks and general NL-centric tasks. Then this work also extends the scope to *Symbol+Delegation* setting, which uses LLM to generate symbolic representation and delegate the reasoning process to the external solver. Such a setting satisfies our expectation to build a better symbol interface.

C.1 Tests in Text-to-Symbol Generation Tasks

Planning These tasks involve controlling the robot to finish some tasks in the defined environments. The input is the natural language description of the initial states and the final goals, while the symbolic output in the Planning Domain Definition Language (PDDL) form can be executed by the symbolic planner. For the four settings, *Blocksworld* involves stacking blocks in order. *Termes* involves moving blocks to a specific position in the grid. *Floortile* is to color the floors with the instructions. *Grippers* is to gripper and move balls from room to room. We use the BLEU metric to measure the correctness of generated forms.

SQL They cover three representative Text-to-SQL datasets, *Spider*, *Sparc* and *Cosql*. Given the schema and the natural language query, the output is the corresponding SQL. We use the exact match as the metric.

KG / DB It is similar to the Text-to-SQL tasks, which require generating the symbolic form of the query given the natural language question and

Domains	Tasks	# Train	# Test	Sampled?	Access	Few-shot?	Original Source
Planning	Blocksworld	37,600	20		GPT-4+Evol	✓	Liu et al. (2023)
	Termes		20		GPT-4+Evol	✓	Liu et al. (2023)
	Floortile		20		GPT-4+Evol	✓	Liu et al. (2023)
	Grippers		20		GPT-4+Evol	✓	Liu et al. (2023)
SQL	Spider	109,582	1,034		Direct		Yu et al. (2018)
	Sparc		1,625		Direct		Yu et al. (2019b)
	Cosql		1,300		Direct		Yu et al. (2019a)
KG / DB	WebQSP	3,241	1,639		Direct	✓	Yih et al. (2016)
	GrailQA	53,222	6,463		Direct	✓	Rogers et al. (2023)
	CompWebQ	37,444	3,531		Direct	✓	Talmor and Berant (2018)
AMR	AMR3.0	68,778	1,898		Direct	✓	Knight and et al. (2020)
	AMR2.0	45,436	1,371		Direct	✓	Knight and et al. (2017)
	BioAMR	7,150	500		Direct	✓	Banarescu et al. (2013)
Ontology	Tekgen	11,219	4,062		Direct	✓	Agarwal et al. (2021)
	Webnlg	3,415	2,014		Direct	✓	Gardent et al. (2017)
API	MTOP	18,784	500	✓	Direct	✓	Li et al. (2021)
	TOPv2	149,696	500	✓	Direct	✓	Chen et al. (2020)
	NLmaps	21,657	10,594		Direct	✓	Lawrence and Riezler (2018)
Command	SCAN	25,990	4,182		Direct+Evol	✓	Lake and Baroni (2018)
Code	NL2BASH	11,971	746		Direct	✓	Lin et al. (2018)
	NL2RX	10,808	1,000	✓	Direct	✓	Locascio et al. (2016)
	NL2Python	12,005	500	✓	Direct	✓	Husain et al. (2019)
	NL2Java	11,978	500	✓	Direct	✓	Husain et al. (2019)
	NL2Go	12,001	500	✓	Direct	✓	Husain et al. (2019)
FOL	FOLIO	2,006	500	✓	Direct	✓	Han et al. (2022)
	MALLS	39,626	1,000	✓	Direct	✓	Yang et al. (2023)
	LogicNLI	11,559	2,373	✓	Direct	✓	Tian et al. (2021)
Visual	GQA	36,086	2,000	✓	Direct	✓	Hudson and Manning (2019)
	CLEVR	47,081	2,000	✓	Direct+Evol	✓	Johnson et al. (2017)
	Geometry3k	2,864	601		Direct	✓	Lu et al. (2021)
Math	GSM8K-Code	8,453	100	✓	GPT-4	✓	Cobbe et al. (2021)
	AQUA-Code	31,144	100	✓	GPT-4	✓	Ling et al. (2017)
	MATH-Code	4,426	100	✓	GPT-4	✓	Hendrycks et al. (2021b)
AI4Science	CheBi-20	35,629	3,300		Direct	✓	Edwards et al. (2022)

Table 5: Detailed illustrations of 34 text-to-symbol generation tasks. *# Train* and *# Test* represent the number of training and test samples respectively. *Sampled?* means whether the test split is sampled from the original dataset. *Access* is related to how we obtain the data, including directly from off-the-shelf benchmarks (Direct), prompting GPT-4 (GPT-4), and applying the symbol-evol strategy (Evol). *Few-shot?* denotes whether few-shot samples are included. *Original Source* is the citation of the original paper.

schema. But *WebQSP* and *GrailQA* leverage the s-Expression form while *CompWebQ* uses SPARQL format. We use the F1 metric for *WebQSP* and the exact match metric for *GrailQA* and *CompWebQ*, following previous work (Xie et al., 2022).

AMR They are classical semantic parsing tasks, where the input sentence is parsed into an abstract syntax graph. We use the Smatch metric to measure the generated form on *AMR3.0*, *AMR2.0*, and *BioAMR* datasets.

Ontology It focuses on the domain of knowledge graph construction. Given the ontology (i.e, predefined relations or entities) and natural language sentence, it is required to output the triples. We employ F1 scores introduced in (Mihindukulasooriya et al., 2023) to measure the performances on *Tekgen* and *WebNLG*.

API These tasks require the output of the API calling form based on the natural language query. *MTOP* and *TOPv2* cover various domains like controlling the music player, and setting alarms. *NLMAPS* focuses on calling the maps.

Command *SCAN* involves outputting action sequences based on the commands to control robots. The exact match metric is used to measure the generation accuracy.

Code It involves five representative programming languages, including *Bash*, *Regular Expression*, *Python*, *Java* and *GO*. They are tested with the BLEU metric.

FOL It covers three datasets in NL-to-FOL domain, that is *FOLIO*, *MALLS* and *LogicNLI*. Logic Equivalence (LE) is leveraged as the metric, following (Yang et al., 2023).

Visual Three multi-modal question answering datasets *GQA*, *Clevr* and *Geometry3K* are included for test. In these scenarios, we only focus on the natural language parts and transform the natural language query into function symbol forms. The exact match metric is used to measure the performances.

Math As we discussed, transforming the natural language question into Python code is one of the faithful ways to solve math problems. Hence, we measure the accuracy of the generated Python code with the BLEU metric. The ground-truth code is derived by prompting GPT-4, where the ones that can execute the correct answer are preserved.

AI4Science In *CheBi* dataset, the model is required to generate the correct molecular formula given the natural language descriptions. Exact match metric is used for measure.

C.2 Tests in General Tasks

MMLU It covers 57 tasks including different subjects STEM, humanities, social sciences, and others. Our evaluations are based on (Hendrycks et al., 2021a).

Big Bench Hard The benchmark is designed for testing LLM capability in challenging reasoning tasks. We select 21 tasks in BBH for the test, based on Open-LLM-Leaderboard[4].

C.3 Tests in Symbol+Delegation Setting

Math Reasoning We generate Python code with Symbol-LLM and use Python interpreter as the delegation. The datasets include GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021b), GSM-Hard (Gao et al., 2023), SVAMP (Patel et al., 2021), Asdiv (Miao et al., 2020), AddSub (Hosseini et al., 2014), SingleEQ (Roy et al., 2015), SingleOP (Roy et al., 2015) and MultiArith (Roy and Roth, 2015). The former two datasets are in-domain, while the latter seven datasets are under OOD settings.

Note that MATH dataset includes various ground-truth answer formats (e.g., with diverse units), thus it is difficult to parse the correct values to evaluate the LLMs. Hence, we use manually-crafted templates to derive the ground-truth values, leading to around 4,000 samples for test.

Symbolic Reasoning Same as math reasoning, we use Python code + Python interpreter to solve the problems. Two OOD tasks are used for test, i.e., Colored Objects (Suzgun et al., 2023) and Last Letter Concatenation [5].

Logical Reasoning We take three representative datasets into consideration, i.e., FOLIO (Han et al., 2022), ProofWriter (Tafjord et al., 2021) and ProntoQA (Saparov and He, 2022). We follow the strategy proposed in (Pan et al., 2023) to conduct the reasoning. Detailedly, for FOLIO, we generate FOL representations first and delegate the solution to the FOL solver. For ProofWriter and ProntoQA tasks, we generate logic programming language and delegate the reasoning to *Pyke* expert system.

Robotic Planning For robotic planning tasks, we transform the natural language description into PDDL and use fastdownward (**?**) as the symbolic solver. Besides the four datasets mentioned in text-to-symbol generation tasks, we also employ two OOD datasets into account, i.e. Barman and Tyreworld.

Visual Question Answering We further extend the application scope of Symbol-LLM to the multi-modal domain and test on Geometry3K dataset (Lu et al., 2021) for illustration. But we only concentrate on the processing of the NL part. Detailed, we parse the natural language sentence into logic forms and rely on the baseline method (Lu et al., 2021) to conduct the multi-modal reasoning.

D Experimental Settings

In the implementation, this work leverages the AdamW optimizer with a learning rate of 2e-5 for both *Injection* and *Infusion* stages. The learning rate schedular is set to *Linear*. The epoch number is set to 1 for both stages. In the *Injection* stage, the model weights are initialized from LLaMA-2-Chat and the tuned model is named Symbol-LLM_{Base}. In the *Infusion* stage, we initialize the model from Symbol-LLM_{Base} and obtain Symbol-$\text{LLM}_{\text{Instruct}}$ at last. These settings are consistent for both 7B and 13B variants.

For a comprehensive evaluation, we include the following strong baselines. They are categorized into *Close-source* and *Open-source* ones:

[4] https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

[5] Test data is based on: https://huggingface.co/datasets/ChilleD/LastLetterConcat

Close-source Baselines

- **GPT-3.5** We access OpenAI API to call the model. Specifically, `GPT-3.5-turbo` version is employed for evaluation across a wide range of tasks.

- **Claude-1** We access Anthropic API to call the model. We select `Claude-instant-1.2` version for evaluation.

Open-source Baselines

- **LLaMA-2-Chat** Since Symbol-LLM is initialized from LLaMA-2-Chat, we include it as the baseline. In general, LLaMA-2-Chat series is regarded as an excellent NL-centric interface for interaction and reasoning, which exhibits great performance on vast NL tasks.

- **Single SFT** We conduct SFT on LLaMA-2-Chat models for tasks in one specific domain. The obtained models can fully overfit the single domain, thus serving as a strong baseline for comparison.

- **CodeLLaMA-Instruct** Based on the origin LLaMA-2 models, the CodeLLaMA series is continually pretrained and finetuned with code data. Considering code is one of the specific symbols in our work, we include it as one of the strong baselines. For balanced capabilities in general tasks, we leverage *CodeLLaMA-Instruct* for evaluations.

E Overall Performances

Figure 5 presents the overall performance comparison among baseline models. It intuitively demonstrates the obvious advantages of Symbol-LLM on the wide range of tasks. Also, it supports our claim to make Symbol-LLM a balanced foundation LLM on symbols and NL.

F Results on Extensive General Tasks

In Table 6, we select extensive general tasks for comparison. According to OpenCompass (Contributors, 2023), these tasks are divided into several categories, covering *Examinations*, *Knowledge*, *Understanding* and *Reasoning*. Considering the computation cost, we only report the performances of 7B models. The major takeaways are as follows:

Tasks	LLaMA-2-Chat†	Symbol-LLM
Examinations		
AGI-Eval	**28.50**	27.55
C-Eval	31.90	**34.96**
GaokaoBench	**16.10**	13.37
ARC-c	54.90	**61.69**
Knowledge		
BoolQ	**81.30**	77.00
CommonsenseQA	**69.90**	59.21
TrivialQA	**46.40**	40.90
NaturalQuestions	**19.60**	16.48
Understanding		
OpenbookQA	74.40	**79.20**
XSUM	20.80	**33.29**
LAMBADA	66.90	**70.10**
C3	49.80	**52.82**
Reasoning		
CMNLI	36.10	**55.43**
OCNLI	36.40	**48.10**
Ax-b	58.50	**62.68**
Ax-g	51.70	**64.61**
Hellaswag	**74.10**	53.10
SIQA	55.40	**69.60**
MBPP	17.60	**22.80**
ReCoRD	22.50	**39.49**

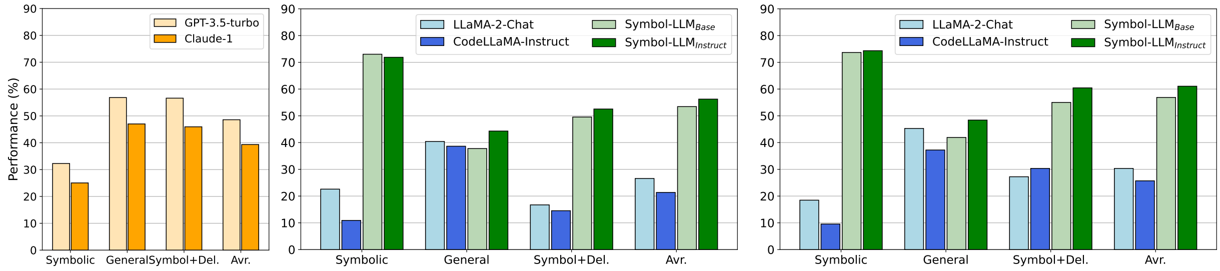
Table 6: Results on extensive general tasks. The evaluations are based on OpenCompass (Contributors, 2023). † denotes the model results directly derived from the leaderboard.

Symbol-LLM demonstrates better overall performances compared with LLaMA-2-Chat. Generally speaking, Symbol-LLM wins more of the tasks than LLaMA-2-Chat. Such superiority is consistent with the findings in MMLU and BBH benchmarks in Table 2. It illustrates that Symbol-LLM can serve as a solid foundational model, significantly enhancing its symbolic capabilities while maintaining its generality.

Optimization empowers Symbol-LLM with improved understanding and reasoning abilities. Among the four task categories, Symbol-LLM is particularly better at understanding and reasoning, beating LLaMA-2-Chat on almost all tasks. Such findings are intuitive because text-to-symbol can be regarded as an abstract form of NL, which enriches the understanding abilities of the model. Meanwhile, the generation of some symbolic forms (e.g., code) involves the implicit reasoning process, which is actually similar to the chain-of-thought strategy. To this end, the superior reasoning capability is within our expectations.

G Results on Symbol+Delegation Setting

In the main paper, we present the results of math reasoning under the *Symbol+Delegation* paradigm.



(a) Close-source (b) Open-source 7B (c) Open-source 13B

Figure 5: Overall results comparison. We report the performances on close-source, open-source 7B as well as open-source 13B LLMs. The results on symbolic tasks, general tasks, symbol+delegation tasks and the average ones are included.

Models	**Del.**	**ColoredObject**	**LastLetter**
Is OOD Setting		✓	✓
Close-source LLMs			
GPT-3.5-turbo	✓	12.45	94.00
Claude-1	✓	46.05	90.67
Open-source LLMs (7B)			
LLaMA-2-Chat	✓	**28.70**	0.00
CodeLLaMA-Instruct	✓	4.60	0.00
Symbol-LLM$_{\text{Base}}$	✓	22.65	90.67
Symbol-LLM$_{\text{Instruct}}$	✓	25.50	**96.67**
Open-source LLMs (13B)			
LLaMA-2-Chat	✓	30.35	0.00
CodeLLaMA-Instruct	✓	1.35	0.00
Symbol-LLM$_{\text{Base}}$	✓	**36.35**	94.00
Symbol-LLM$_{\text{Instruct}}$	✓	34.00	**96.67**

Table 7: Results on Symbolic Reasoning.

Models	**Del.**	**FOLIO**	**ProofWriter**	**PrOntoQA**
Is OOD Setting		✗	✗	✓
Close-source LLMs				
GPT-3.5-turbo	✓	44.61	29.00	52.00
Claude-1	✓	37.25	35.83	55.80
Logic-LM (SOTA)	✓	61.76	70.11	93.20
Open-source LLMs (7B)				
LLaMA-2-Chat	✓	34.80	34.83	50.00
CodeLLaMA-Instruct	✓	32.84	32.50	50.20
Symbol-LLM$_{\text{Base}}$	✓	46.08	**76.50**	55.60
Symbol-LLM$_{\text{Instruct}}$	✓	**49.02**	76.33	**57.20**
Open-source LLMs (13B)				
LLaMA-2-Chat	✓	33.33	35.83	49.20
CodeLLaMA-Instruct	✓	32.84	34.00	50.00
Symbol-LLM$_{\text{Base}}$	✓	33.82	**76.33**	48.40
Symbol-LLM$_{\text{Instruct}}$	✓	**35.29**	75.50	**53.60**

Table 8: Results on Logical Reasoning. All results are obtained under the one-shot setting.

Next, we will provide the remaining 5 scenarios, i.e., symbolic reasoning (G.1), logical reasoning (G.2), robotic planning (G.3), visual question answering (G.4) and table question answering (G.5).

G.1 Symbolic Reasoning

In symbolic tasks, we also adopt the Python code as the generated symbolic representations, and leverage a Python interpreter to conduct the reasoning. Two representative tasks, *Colored Objects* and *Last Letter Concatenation* are selected for testing under the zero-shot setting.

From the results in Table 7, the Symbol-LLM series are competitive in both tasks. Notably, even Symbol-LLM-7B shows over 10% superiority over GPT-3.5-turbo in *Colored Object* task. It is worth noticing that LLaMA-2-Chat models underperform consistently in *Last Letter* task. Since samples in this dataset share similar forms, the model tends to fail if the model does not master the techniques required for solving it.

G.2 Logical Reasoning

In logical reasoning tasks, we take three tasks into consideration, i.e., FOLIO, ProofWriter and ProntoQA. For the FOLIO task, Symbol-LLM first transforms the natural language into FOL forms and delegates the solution to the FOL solver. ProofWriter and ProntoQA are represented in logic programming language and integrate *Pyke* expert system for deductive reasoning.

Results are listed in Table 8. Symbol-LLM-7B series performs relatively better than 13B counterparts. Among all three tasks, the Symbol-LLM-7B series outperforms GPT-3.5-turbo with large advantages. In comparison with the SOTA model Logic-LM, which is based on off-the-shelf LLMs, Symbol-LLM also wins the ProofWriter tasks, with 5%-6% improvements.

G.3 Robotic Planning

In the field of robotic planning, Symbol-LLM first transforms the natural language description into PDDL forms and relies on the fast downward solver to give the faithful action sequence.

In total, we select 6 different robotic settings to

Models	Del.	Blocksworld	Termes	Floortile	Grippers	Barman	Tyreworld
Is OOD Setting		✗	✗	✗	✗	✓	✓
Close-source LLMs							
GPT-3.5-turbo	✓	55.00	0.00	0.00	100.00	95.00	30.00
Claude-1	✓	55.00	0.00	0.00	85.00	50.00	5.00
Open-source LLMs (7B)							
LLaMA-2-Chat	✓	5.00	0.00	0.00	5.00	0.00	0.00
LLaMA-2-Chat SFT	✓	75.00	**100.00**	0.00	0.00	0.00	0.00
CodeLLaMA-Instruct	✓	5.00	0.00	0.00	20.00	0.00	0.00
Symbol-LLM$_{\text{Base}}$	✓	90.00	**100.00**	5.00	15.00	0.00	0.00
Symbol-LLM$_{\text{Instruct}}$	✓	**100.00**	50.00	**20.00**	**20.00**	0.00	**5.00**
Open-source LLMs (13B)							
LLaMA-2-Chat	✓	0.00	0.00	0.00	**45.00**	**50.00**	5.00
LLaMA-2-Chat SFT	✓	70.00	**100.00**	**25.00**	10.00	0.00	0.00
CodeLLaMA-Instruct	✓	5.00	0.00	0.00	0.00	0.00	0.00
Symbol-LLM$_{\text{Base}}$	✓	90.00	**100.00**	0.00	30.00	0.00	10.00
Symbol-LLM$_{\text{Instruct}}$	✓	**100.00**	90.00	**25.00**	**45.00**	20.00	**35.00**

Table 9: Results on Robotic Planning. The evaluation is under the one-shot setting.

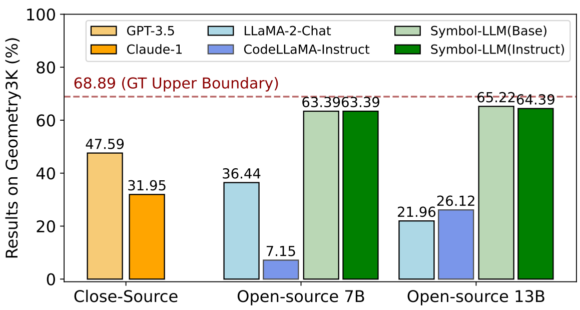


Figure 6: Performances on Geometry3k task.

Models	Del.	WikiSQL	WikiTQ
Is OOD Setting		✓	✓
Close-source LLMs			
GPT-3.5-turbo	✓	28.49	11.58
Claude-1	✓	26.79	8.79
Open-source LLMs (7B)			
LLaMA-2-Chat	✓	21.05	3.50
CodeLLaMA-Instruct	✓	20.18	2.88
Symbol-LLM$_{\text{Base}}$	✓	70.88	**17.15**
Symbol-LLM$_{\text{Instruct}}$	✓	**73.75**	16.97
Open-source LLMs (13B)			
LLaMA-2-Chat	✓	34.86	7.50
CodeLLaMA-Instruct	✓	33.15	6.70
Symbol-LLM$_{\text{Base}}$	✓	**71.69**	**17.31**
Symbol-LLM$_{\text{Instruct}}$	✓	69.83	15.31

Table 10: Results on Table Question Answering.

verify the proposed method. Results are presented in Table 9. Among four in-domain tasks, Symbol-LLM performs pretty well compared with strong baselines, achieving the best results in most cases. Even with GPT-3.5-turbo and Claude-1, both our 7B and 13B series win 3 (out of 4) tasks. However, it struggles a lot in OOD tasks. Only in *Tyreworld* scenario, Symbol-LLM$_{\text{Instruct}}$-13B achieves the best result, beating all close-source and open-source baselines. It is required to state that these selected robotic planning tasks are very challenging, given the length and rigor requirements of the generated programming language. Even close-source LLMs fail in some scenarios. Therefore, we argue it is still an open question for future studies.

G.4 Visual Question Answering

We also explore our potential in the multi-modal scenario. Geometry question answering is selected as the task for the test. Note that the understanding of image is not within our scope, we only focus on the text part and transform the natural language query into logical forms. Then the solution is delegated to the off-the-shelf baseline methods. Comparison results are shown in Figure 6.

The top red bar means the performances using the annotated logic forms from the text. Note that since the utilized delegation method is a neural-based baseline just for a simple evaluation, the upper boundary does not represent the boundary of this task. From the figure, Symbol-LLM variants are approaching it and significantly outperform all the other baselines.

G.5 Table Question Answering

Table (or database) question answering is also a hot topic in recent years. Thus, we select two OOD tasks WikiSQL and WikiTQ for evaluations. The natural language query is first transformed into an SQL query and it is executed by an SQL solver over the given tables or databases under the zero-shot setting. We report experimental results in Table 10.

Symbol-LLM series are consistently superior to all open-source and close-source baselines, with over 40% margins in WikiSQL and 3%∼14% advantages in WikiTQ.

H Supplementary Experiments

H.1 Comparison with Single Domain SFT

As discussed above, one of our hypotheses is that various symbols share underlying interrelations, though they are in quite different forms. Thus, we expect that the learning of symbolic knowledge will mutually benefit each other if they are treated in a unified manner.

We present the comparison results between single SFT and unified SFT in Figure 7. The light blue bar denotes the single domain SFT while the dark blue one is the unified SFT. We categorize the text-to-symbol generation tasks into 12 task domains, according to their similarity in symbolic forms. Within one domain, all tasks are tuned together and the performances on test splits are averaged as the single SFT results. To reduce the effect of the tuning strategy, we utilize the Symbol-LLM$_{\text{Base}}$ model to measure the results on the unified SFT setting. Each sub-figure in Figure 7 corresponds to one specific domain.

In most domains, unified SFT is superior to single-domain SFT. Larger gains are observed in some uncommon symbolic forms, such as PDDL for planning tasks and molecular formulas in AI for Science scenarios. It presents the possibility that unified SFT on various symbols may help extend the model coverage to low-resource cases. It is also worth noting that in some cases, single-domain SFT performs a little better than Symbol-LLM$_{\text{Base}}$. This is because purely overfitting on specific symbolic forms with powerful LLMs is usually easy to get promising results.

H.2 Extrapolating to New Symbols

In the above section, we introduce *symbol-evol* strategy to expand sample diversity and facilitate the training of instruction-following ability. Following this strategy, we can also automatically generate abundant novel instructions to extrapolate to new symbols. To this end, we further evaluate Symbol-LLM by following novel instructions.

The experiments are based on *Clevr* and *SCAN* tasks. Applying *symbol-evol* strategy, we obtain *Clevr-evol* and *SCAN-evol* datasets. Evaluation results are presented in Figure 8.

From the results, the more complex setting (i.e., green bar) does not induce a significant decrease in model performance. Especially, in the *Clevr* task, Symbol-LLM even does better given the novel instructions. It uncovers that Symbol-LLM follows the instructions during the reasoning process, instead of merely memorizing the specific symbolic forms.

H.3 Training Data Scaling

We also explore the scaling law of the training data. Specifically, we sample $\mathcal{D}_s$ in the *Injection* stage at a ratio of 10%, 40%, and 70%. And the performances on the 34 symbolic generation tasks are reported in Figure 9.

As the proportion of training data increases, the performance of the model continues to improve and has not been saturated. This indicates that the ability to handle symbol tasks is not well stored in the origin LLaMA-2-Chat model. It requires additional symbolic knowledge injection to facilitate the performances.

Also, the performance differences between 7B and 13B are not significant. Especially when provided with more symbolic data, the 7B model is approaching the 13B model.

I Analysis: Alignment and Uniformity

Beyond performances on symbolic tasks, it is also required to reveal what leads to superiority. Inspired by (Wang and Isola, 2020), we extend the ideas of *Alignment* and *Uniformity* to evaluate the model perception of symbolic knowledge. *Alignment* [6] is utilized to measure the interrelations between symbolic forms. *Uniformity* quantifies the degree of evenness or uniformity in the distribution of symbolic representations. The concept of a uniform feature distribution is valuable as it encourages a higher information entropy, representing more information retention.

Alignment Different from the original implementation (Wang and Isola, 2020) which considers positive pairs in the contrastive learning, this work takes the symbolic sequences under the same symbolic form as the positive pairs. For any symbolic form *X*, their data distributions are referred to as P_X. The alignment within *X* can be measured with the following formula:

[6] Here, *Alignment* refers to the concept in contrastive learning, but is not related to the alignment technique in LLMs.

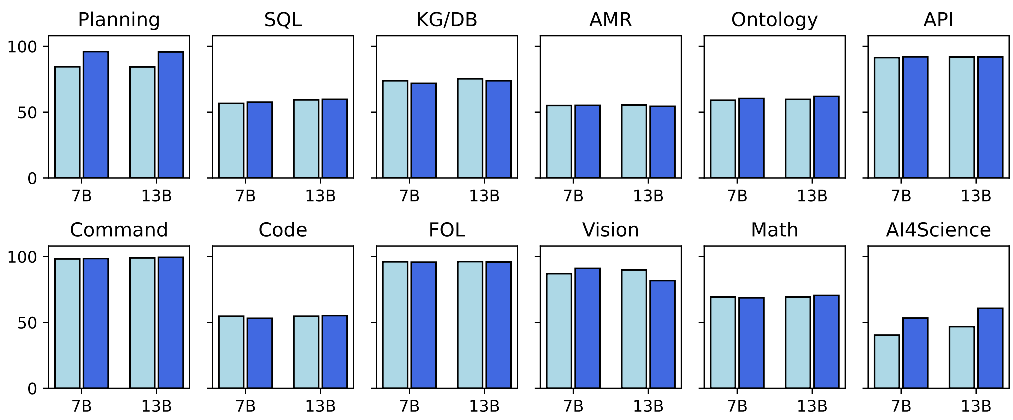


Figure 7: Comparison between single SFT and unified SFT.

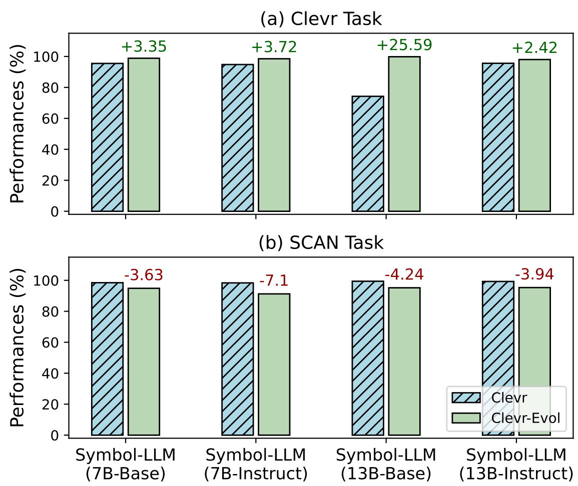


Figure 8: Comparisons between original setting and user-defined setting.

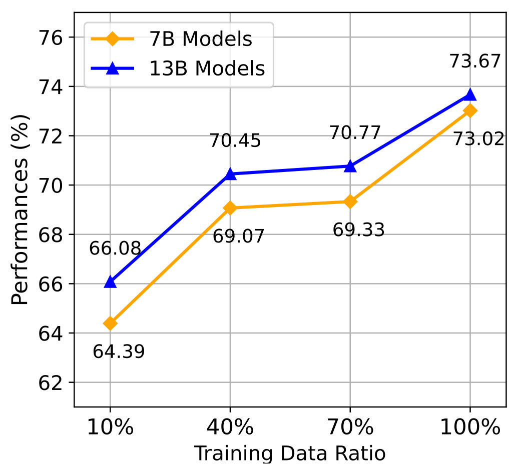


Figure 9: Training data scaling.

$$\mathcal{L}_{align}(X) = \underset{x_1,x_2 \overset{i.i.d}{\sim} P_X}{\mathbb{E}} \|f(x_1) - f(x_2)\|^2, \quad (3)$$

where x_1 and x_2 are samples from the specific symbolic form X. $f(\cdot)$ returns the LLM embeddings of the symbolic sequences. $\|\cdot\|$ returns the norm of the vector.

In the implementation, we select 16 main symbolic forms and sample 100 symbolic sequences for each form to measure the alignment. $f(\cdot)$ leverages the mean pooling representation of the last hidden states of the LLM. We average all the alignment scores from the 16 symbols to obtain the final one. Notably, we employ logarithmic operations on the *Alignment* loss to reduce scale, without impacting their relative comparison.

Uniformity Apart from alignment, we also calculate the uniformity of the LLMs on symbolic sequences. The evaluation of the uniformity $\mathcal{L}_{uniform}$ is implemented by the following formula:

$$\mathcal{L}_{uniform} = \log \underset{x,y \overset{i.i.d}{\sim} P_{data}}{\mathbb{E}} e^{-2\|f(x)-f(y)\|^2}, \quad (4)$$

where the data distribution P_{data} covers all the symbolic sequences. $f(\cdot)$ also utilizes the mean pooling representation of the last hidden states of the LLM.

Leveraging the above definitions, further analysis and comparison on Symbol-LLM are conducted. The item-wise conclusions are listed as follows:

(1) Symbol-LLM optimizes symbol distinctiveness and overall expressiveness in the embedding space (superior *Alignment* and *Uniformity*).

Based on the equation 3 and 4, we can assess the proficiency of LLMs in handling symbols. Figure 10 presents the visualization of *Alignment*-

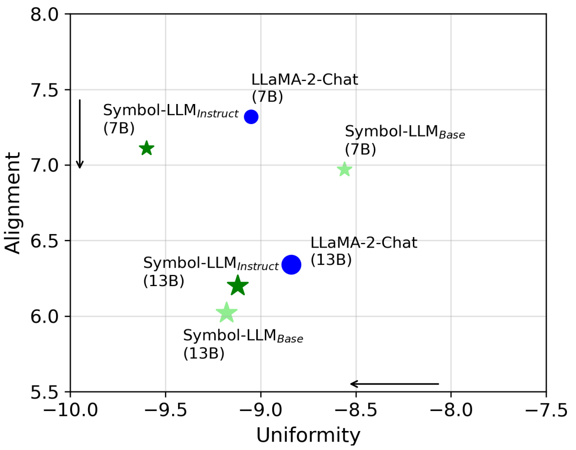
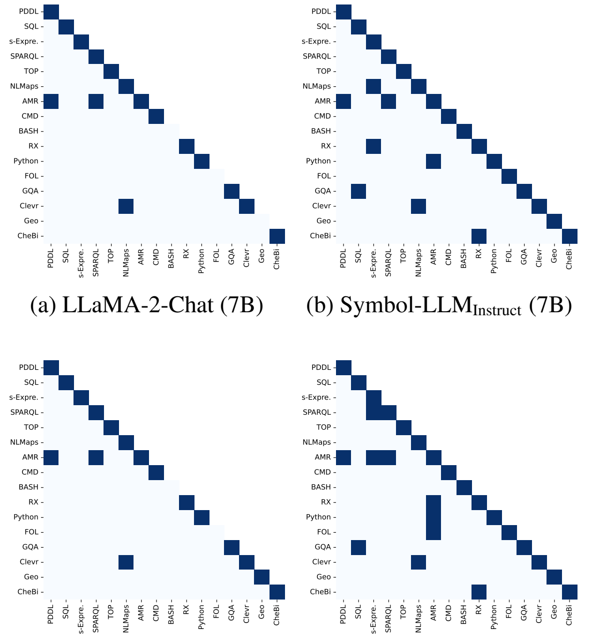


Figure 10: Visualization of *Alignment-Uniformity*. Both metrics are inversely related, which means a lower value indicates better performance.



(a) LLaMA-2-Chat (7B) (b) Symbol-LLM$_{\text{Instruct}}$ (7B)

(c) LLaMA-2-Chat (13B) (d) Symbol-LLM$_{\text{Instruct}}$ (13B)

Figure 11: Visualization of the alignment relations between symbols. Dark blue denotes a close relation between two symbols in the representation.

Uniformity. The x-axis stands for uniformity while the y-axis is the alignment. Both of these metrics are better when kept as small as possible.

From the figure, Symbol-LLM$_{\text{Instruct}}$ models perform consistently better than the original LLaMA-2-Chat models, with obvious merit in *Alignment* and *Uniformity*. It can be regarded as an in-depth explanation for the superior performances on symbolic generation tasks. Further, it witnesses that the two-stage tuning framework actually corrects the weakness of *Uniformity* under 7B settings (Symbol-LLM$_{\text{Instruct}}$ v.s. Symbol-LLM$_{\text{Base}}$).

Both metrics are well optimized with the proposed two-stage tuning framework as well as the symbolic data collection. For Symbol-LLM$_{\text{Base}}$ models, though the 7B version witnesses some loss in Uniformity, they consistently achieve superior alignment.

(2) Symbol-LLM excels at capturing symbolic interrelations.

The above calculation of *Alignment* roughly depicts the similarity among samples under the same symbolic form. To this end, we extend the idea of *Alignment* to measure the interrelation between any two symbolic forms X and Y. The score $S(X, Y)$ is calculated based on the following formula:

$$S(X, Y) = \underset{x \sim P_X, y \sim P_Y}{\mathbb{E}} \|f(x) - f(y)\|^2, \quad (5)$$

where x is one symbolic sample in the form of X, while y is one sample in the symbolic form Y. We binarize the scores with the manually defined threshold for a more intuitive illustration. We ensure the same threshold under a fair comparison of the same model size. And the set of thresholds will not affect the overall conclusion.

The visualization is presented in Figure 11, where the dark blue denotes the closer relation in the representation while the light one is the opposite. We make comparisons between the original LLaMA-2-Chat models and Symbol-LLM$_{\text{Instruct}}$ models, separately for the size of 7B and 13B.

For the original LLaMA model (Figure 11a and 11c), the representations between different symbols exhibit significant sparsity. There are only three pairs of symbolic forms that effectively demonstrate the interrelations in the embedding space, i.e., *AMR-PDDL*, *AMR-SPARQL* and *CLEVR-NLMaps*. Also, under several symbol systems (e.g., *Bash*, *FOL*), the representation space of samples is also very scattered. The above observations demonstrate that previous foundational LLMs (i.e., LLaMA-2-Chat) lack the ability to capture the interrelations among symbolic systems.

In comparison, Symbol-LLM$_{\text{Instruct}}$ series models excel at reflecting the interrelations between symbols. As presented in Figure 11b and 11d: 1) Symbols exhibiting potential connections are effectively aligned within the representation space, i.e., *Python-AMR* and *CheBi-RX*. 2) Samples within each symbol are pulled closer together.

Combining the above two observations and analysis, the superior performances of Symbol-LLM on the symbolic generation tasks are sourced from better alignment among symbols in the embedding space as well as the optimized uniformity.