Learning to Solve Domain-Specific Calculation Problems with Knowledge-Intensive Programs Generator

Chengyuan Liu[1,2†], **Shihang Wang**[2], **Lizhi Qing**[2], **Jun Lin**[2], **Ji Zhang**[2], **Fei Wu**[1], **Kun Kuang**[1,3*]

{liucy1,wufei,kunkuang}@zju.edu.cn,

{wangshihang.wsh,yekai.qlz,linjun.lj,zj122146}@alibaba-inc.com

[1]*College of Computer Science and Technology, Zhejiang University,*

[2]*Tongyi Lab, Alibaba Group,* [3]*Law&AI Lab, Zhejiang University*

Abstract

Domain Large Language Models (LLMs) are developed for domain-specific tasks based on general LLMs. But it still requires professional knowledge to facilitate the expertise for some domain-specific tasks. In this paper, we investigate into knowledge-intensive calculation problems. We find that the math problems to be challenging for LLMs, when involving complex domain-specific rules and knowledge documents, rather than simple formulations of terminologies. Therefore, we propose a pipeline to solve the domain-specific calculation problems with **K**nowledge-**I**ntensive **P**rograms **G**enerator more effectively, named as **KIPG**. It generates knowledge-intensive programs according to the domain-specific documents. For each query, key variables are extracted, then outcomes which are dependent on domain knowledge are calculated with the programs. By iterative preference alignment, the code generator learns to improve the logic consistency with the domain knowledge. Taking legal domain as an example, we have conducted experiments to prove the effectiveness of our pipeline, and extensive analysis on the modules. We also find that the code generator is also adaptable to other domains, without training on the new knowledge.

1 Introduction

Large Language Models (LLMs) (Brown et al., 2020; Anil et al., 2023; Yang et al., 2023a; Touvron et al., 2023) have exhibited outstanding results on several tasks, covering both language understanding and generation, even without additional training. Based on which, domain-specific LLMs (Zhou et al., 2024; Yang et al., 2023b; Zhang et al., 2023) are developed via techniques such as continual pre-training and supervised fine-tuning (SFT).

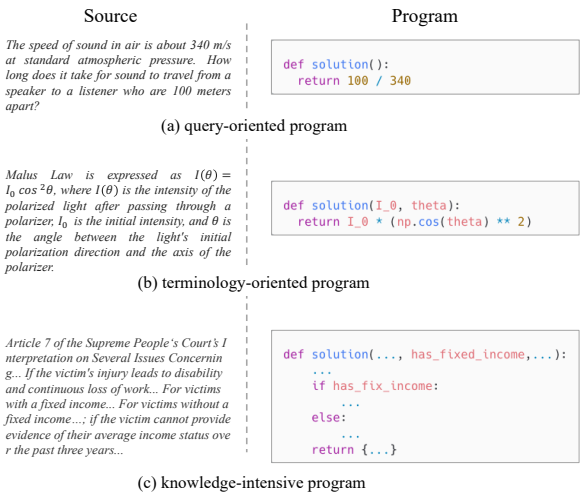

Figure 1: Comparison between program generation paradigms from different sources. (a) Query-oriented programs are generated to solve a specific query. (b) Terminology-oriented programs describe domain-specific concepts with several formulations, source from definitions of terminologies. (c) Knowledge-intensive programs follow more complex instructions with domain-specific conditions and rules.

Although the LLMs are enhanced by domain corpus, it still requires professional knowledge to facilitate the expertise for some domain-specific tasks (Yiquan et al.; Ma et al., 2024; Nay et al., 2023). In this paper, we mainly study at the domain-specific QA tasks involving math calculation.

Since LLMs are not good at numeric calculation in auto-regression, researchers adopt algebraic expressions (Imani et al., 2023) and programs (Zhao et al., 2024) to calculate precise numeric answers. The approaches can be classified by the source of generation, as shown in Figure 1. Zhao et al. (2024) conduct analysis on Program of Thought (PoT) for financial domain. The programs are generated to answer the query directly. We call them the **query-oriented programs**, as in Figure 1 (a). To introduce external knowledge, Ma et al. (2024) extracts

*Corresponding author.

†This work was done when Chengyuan Liu interned at Alibaba.

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 4776–4791
April 29 - May 4, 2025

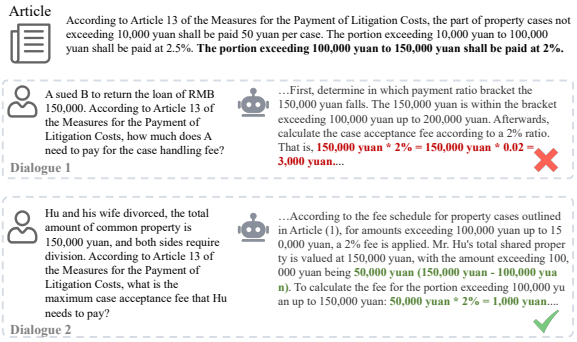

Figure 2: For the same article, the response may follow different calculation logic given different queries. Note that we only present part of the responses containing the logic concerned, i.e., "*The portion exceeding 100,000 yuan to 150,000 yuan shall be paid at 2%.*".

several definitions of terminologies from scientific domain as executable tools in Python language, and builds a dataset to train the LLM's ability to call the tools. As shown in Figure 1 (b), the programs describe domain-specific concepts and terminologies and return the value of the concept, named **terminology-oriented programs** in this paper.

From a pilot study, we find that it can be hard for LLMs to strictly follow the domain knowledge represented in documents involving instructions with complex conditions and constraints, rather than simple formulations. As illustrated in Figure 2, the understanding to the article rules shifts with different queries, thus the LLM may generate unstable answers. Therefore, we introduce **knowledge-intensive programs**, as in Figure 1 (c), which may contain hierarchical conditions involving diverse variables. Additionally, knowledge-intensive programs return all potential results from the knowledge, instead of a single scalar as query/terminology-oriented programs. In this way, the bias caused by queries can be removed by representing the essential logic of the article as knowledge-intensive program.

In this paper we design a pipeline to solve the domain-specific calculation problems with **K**nowledge-**I**ntensive **P**rograms **G**enerator step by step, called **KIPG**. For each query, key variables are extracted, then outputs dependent on domain knowledge are calculated with the programs. Given the outputs, the LLM is prompted to conclude the answer. We also propose a method to train a code generator, thus the knowledge-intensive programs can be generated automatically when facing new knowledge documents.

Additionally, we have constructed a QA dataset involving numeric calculation in legal domain with the help of human labor from legal experts. Various law articles are included covering a diversity of cases. From the experiments, our proposed method significantly outperforms the baselines for most of the cases. Given the gold article, KIPG surpasses baselines by a margin. We also find that the learned code generator can be generalized to other domains. In summary, our contributions are three folds:

- We have constructed a numeric calculation dataset involving domain-specific knowledge with complex conditions and instructions, rather than only single definitions of domain-specific terminologies.
- We introduce a pipeline, KIPG, to solve the domain-specific math word problems with knowledge-intensive programs generator, which learns to write the programs automatically according to the domain knowledge.
- We conduct experiments and analysis from multiple perspectives to demonstrate the effectiveness of our framework.

2 Related Work

Domain-specific LLMs Domain-specific LLMs are developed based on general LLMs for domain-specific tasks. With the help of continual pre-training (Ke et al., 2023; Çağatay Yıldız et al., 2024), knowledge from various domains is injected into the parameters of LLMs. Zhou et al. (2024) introduced LawGPT, the first open-source model specifically designed for Chinese legal applications. Yang et al. (2023b) presented an open-source large language model, FinGPT, for the finance sector. Zhang et al. (2023) presented HuatuoGPT, another LLM for medical consultation.

Math Problems for LLMs Auto-regression cannot ensure the correctness of numeric calculation. Researchers have paid much efforts to improve the calculation and reasoning abilities of LLMs. GSM8K (Cobbe et al., 2021) is one of the most popular datasets for math calculation. Wei et al. (2023) introduced Chain-of-Thought (CoT) to elicit reasoning and calculation abilities of LLMs by straightforward prompts. Imani et al. (2023) proposed MathPrompter, a technique that improves performance of LLMs on arithmetic problems

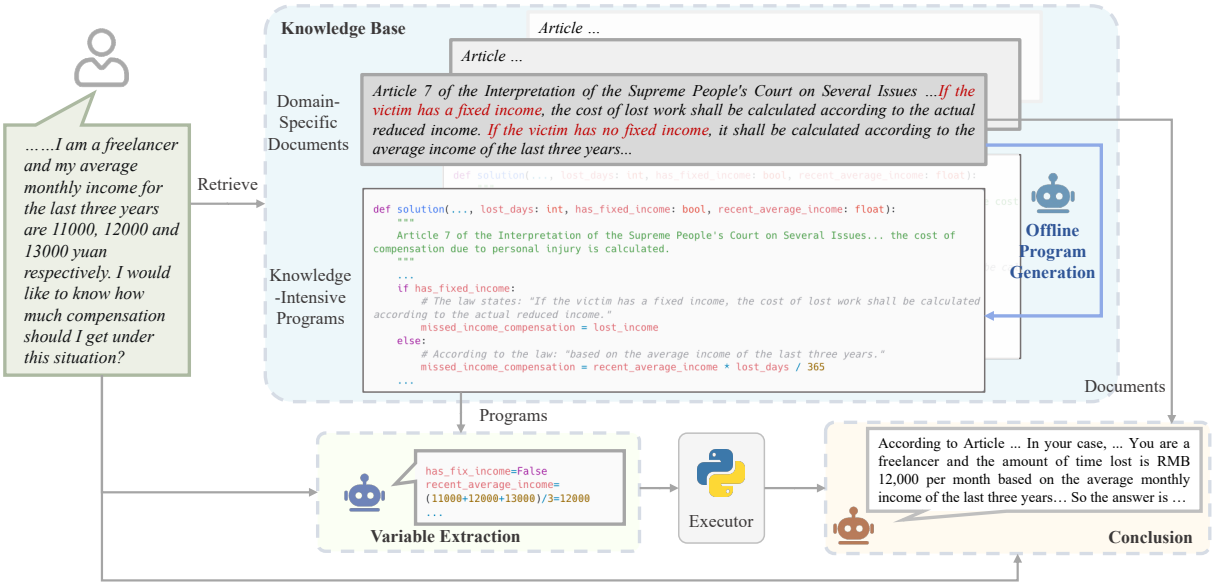

Figure 3: Framework of KIPG. The generator LLM writes programs for each domain-specific document. Given a query, KIPG extracts key variables and executes the programs to calculate the outcomes dependent on domain-specific knowledge step by step, finally concludes the answer.

along with increased reliance in the predictions. Kim et al. (2023) showed that a LLM agent can execute computer tasks guided by natural language using a simple prompting scheme where the agent Recursively Criticizes and Improves its output (RCI).

Domain-specific Math Word Problems Zhao et al. (2024) explored using query-oriented programs for solving knowledge-intensive math reasoning problems. Srivastava et al. (2024) introduced a novel prompting technique for financial tabular question-answering tasks involving math calculation. Nay et al. (2023) explored LLM capabilities in applying tax law to math calculation tasks. Ma et al. (2024) used terminology-oriented programs as tools for scientific problem solving, which mainly describe simple formulation of terminologies and concepts. While we utilize knowledge-intensive programs in this paper, which involve more complex conditions and instructions.

3 Dataset Construction

For experimental analysis, we construct a dataset in legal domain to evaluate the calculation ability of LLMs given domain-specific knowledge describing complex instructions. We also collect various knowledge to build another dataset in medical domain as a testset, to assess the generalization. Details of the dataset construction are described in Appendix E.

Domain	Type	# SubType	# Article	# Instance
Legal	Compensation	12	17	220
	Tax	11	6	216
	Other Fees	8	10	201
	Penalties	2	552	1277
	Traffic Violations	1	6	136
Medical	Term	1	4	153
	indicator	1	6	68
	Medicine	1	5	139

Table 1: Statistics of the constructed dataset.

Knowledge Selection To improve the challenge of the task, we focus on practical domain knowledge, which mostly contains detailed instructions for different cases. It is non-trivial for LLMs to strictly follow the complex knowledge document to perform calculation given queries, especially considering that it requires domain knowledge to identify the corresponding conditions. For legal domain, we have engaged professional lawyers with expertise in Chinese law to identify satisfactory law articles that involve mathematical computations. For medical domain, we search for medication instructions and indicators involving computations from the Internet to ensure the correctness of the documents. Some statistics are provided in Table 1.

Query and Responses The queries should ask a specific question and expect a single number as the answer. The queries are forbidden to provide the clues to the required domain-specific knowledge.

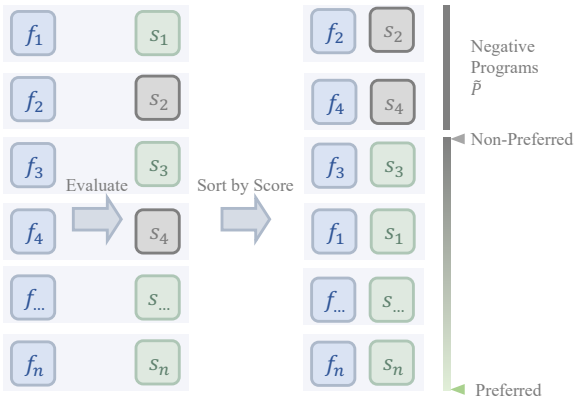

Figure 4: Ranking guideline of DPO data generation.

For the responses, the annotators are prompted to write the reasoning chains as detailed as possible, which help to explain and check the results. A single numeric answer is expected at the end for each query. For evaluation, we extract the numeric answer with regex and calculate the accuracy.

Quality Review To precisely assessing the calculation results of LLMs, we also carefully review the built dataset after the first round of annotation. We correct the improper selected documents and wrong calculations.

Extension by LLM Since the human labor is expensive, we adopt GPT-4 to extend the dataset scale given hand-written instances as examples for each case. We also apply manual quality review before adding the instances into the dataset.

4 Method

In this Section, we introduce the framework of KIPG, as illustrated in Figure 3.

4.1 Knowledge-Intensive Programs

Our pipeline works with a program generator LLM, denoted as θ_G. θ_G writes knowledge-intensive programs for each article. To provide necessary information for variable extraction and knowledge understanding, we inject the description to the knowledge source, input/output parameters and program comment to the programs. Details about the above components are available in Appendix G.

4.2 KIPG

Given a query Q, the related domain-specific knowledge D can be obtained by retrieval. Here, we focus on how to understand and strictly follow the instructions and rules with the documents as background knowledge. An extraction model, denoted as θ_E, is adopted to extract the key variables from the query, according to the knowledge-intensive program f. The variables sever as input arguments to the program. Then, the program is executed by the system executor.

$$\begin{aligned} I &= \theta_E(Q, f) \\ O &= f(I) \end{aligned} \tag{1}$$

The variables are packed together as (I, O), which contain domain-specific information derived from the document and are appropriate for the case of Q. The response can be concluded by another model θ_C given the deduced information as

$$Y = \theta_C(Q, D, (I, O)) \tag{2}$$

The original document D helps θ_C to understand and review the variables, and the query Q provides the calculation target of conclusion. Note that (I, O) merely provide the answer derived from target document, not the direct answer to Q. For example, an article may stipulate the total amount to be reimbursed in the event of an accident, but the user inquires about the amount payable per month if the repayment is spread over 12 installments. In this way, it requires the θ_C to do post-calculation according to the query.

4.3 Code Generation

To improve the logic consistency with the original documents, we propose to enhance θ_G with iterative Direct Preference Optimization (DPO) (Rafailov et al., 2024). Preference alignment is studied across all time, hindered by the fuzzy standard of "helpful, useful, harmless" responses. In our task, we are also facing the challenge of identifying the best programs, since even the programs generated by GPT-4 may also be problematic. Fortunately, we notice that evaluating the correctness of the generated programs is a rather simple task without demand of human labor, different from traditional preference alignment.

The ranking guideline of DPO data generation process is illustrated in Figure 4. Specifically, for each article, we sample n different programs with diverse beam search (Vijayakumar et al., 2018), denoted as $P = [f_1, f_2, \ldots, f_n]$. The programs are evaluated by KIPG respectively. We calculate the correctness when using a program f_i to solve all queries requiring the corresponding document, and the score is denoted as s_i. There are also potential

Type	Qwen-Max	Vanilla	EEDP	CoT	PoT	Algebraic	RaR	ICL	RCI	KIPG
					Qwen2-7B					
Compensation	34.55	44.55	45.00	46.82	39.09	40.45	31.36	38.18	38.64	**50.91**
Tax	73.15	60.65	62.04	64.35	30.56	59.72	40.28	57.87	60.19	**73.15**
Other Fees	61.19	57.71	60.70	52.24	33.83	57.71	38.81	53.73	54.23	**64.68**
Penalties	87.86	81.91	90.92	86.06	71.42	72.04	62.18	82.54	83.87	**92.17**
Traffic Violations	62.50	48.53	62.50	55.88	45.59	55.15	38.97	50.00	54.41	**68.38**
AVG	63.85	58.67	64.23	61.07	44.10	57.01	42.32	56.46	58.27	**69.86**
					Qwen2-72B					
Compensation	-	49.55	52.73	50.00	**57.27**	52.27	28.18	51.36	51.36	56.82
Tax	-	68.98	75.00	76.12	75.46	61.57	39.35	70.00	73.61	**77.78**
Other Fees	-	47.26	61.19	48.94	61.19	56.22	30.35	62.69	58.71	**71.14**
Penalties	-	77.76	93.81	83.16	90.37	83.32	54.78	90.27	86.03	**95.69**
Traffic Violations	-	47.79	65.44	61.97	53.68	69.85	19.85	50.76	58.09	**72.79**
AVG	-	58.27	69.63	64.04	67.59	64.65	34.50	65.02	65.56	**74.84**

Table 2: Main results given oracle documents in legal domain. **The training dataset only includes the first 3 types, while the calculation problems of the "Penalties" and "Traffic Violations" are never seen during training.** We first conduct experiments on Qwen2-7B-Instruct, denoted as "*Qwen2-7B*", which is additionally compared with "Qwen-Max" (200B parameters). Then we supplement the calculated variables from Qwen-7B directly to Qwen2-72B-Instruct, denoted as "*Qwen2-72B*". "Vanilla" indicates directly using the document without any prompting or advanced skills.

programs failed to be executed, whose scores are assigned as -1, and the program set is called negative programs, represented by $\tilde{P} \subseteq P$. We filter out $\tilde{P}$ by the following aspects: 1) If f_i can not be executed and throw an runtime exception, then $f_i \in \tilde{P}$. 2) If the parameter definitions in the comment of f_i are fuzzy[1], then $f_i \in \tilde{P}$. 3) If f_i contains words like "assuming", it may generate hallucinations, fabricating content that does not exist, then $f_i \in \tilde{P}$.

All valid programs are sorted by the scores in ascending order. Then all later programs are more preferred than previous programs. Besides, executable programs are more preferred than negative programs. Gathering each pair of programs, we can build the dataset D_G, then train the code generator θ_G with DPO loss:

$$\mathcal{L}_G = -\mathbb{E}_{(d,p_w,p_l)\sim D_G}\left[\log\sigma\left(\beta\log\frac{\theta_G\left(p_w|d\right)}{\theta_{\text{ref}}\left(p_w|d\right)} - \beta\log\frac{\theta_G\left(p_l|d\right)}{\theta_{\text{ref}}\left(p_l|d\right)}\right)\right] \tag{3}$$

where $\langle p_w, p_l\rangle$ represents a pair of positive program and negative program.

By iterative inference with new parameters, KIPG gets different groups of programs and scores, consequently constructs new training data. Thus the overall accuracy improves step by step.

4.4 Initialization

From experiments, we find that vanilla extraction LLM θ_E struggles to follow the extraction instruction. Mostly, it outputs the extraction in random formats, hindering the following usage of the variables. Hence, we initialize θ_E with supervised fine-tuning on extraction instances generated by GPT-4.

5 Experiments

We have conducted extensive experiments comparing KIPG with several competitive baselines, including vanilla LLMs (Yang et al., 2024), CoT (Wei et al., 2023), PoT (Chen et al., 2023), Algebraic (Imani et al., 2023), InContext-Learning (Dong et al., 2024), SciAgent (Ma et al., 2024), RaR (Deng et al., 2024), RCI (Kim et al., 2023), EEDP (Srivastava et al., 2024), and Qwen-Max[2] with 200B parameters. We also try training techniques with different training datasets. Detailed baseline descriptions can be found in Appendix A. We train the models on the types of "Compensation", "Tax" and "Other Fees", while the calculation problems of the "Penalties" and "Traffic Violations" are served as out-of-distribution test questions. We mainly report the accuracy in our ex-

[1]We prompt the generator to include descriptions of parameter units, meanings and data-types in Python language, within the comment. If any of these components is missing, the parameter definition is considered "fuzzy", which makes it difficult for the extraction model to identify appropriate variables and their units.

[2]https://huggingface.co/spaces/Qwen/Qwen-Max-0428

Method	Compensation	Tax	Other Fees	Penalties	Traffic Violations	AVG
Zero/Few-Shot						
-	18.64	33.33	30.85	31.95	24.26	27.81
CoT	19.55	27.78	29.35	33.36	16.91	25.39
ICL	26.82	42.13	46.27	41.74	24.26	36.24
Supervised Fine-Tuning						
-	26.36	37.50	26.87	36.81	18.38	29.18
CoT	25.45	40.74	28.36	36.41	16.91	29.57
L1	28.18	31.94	32.33	27.88	19.85	28.44
L2	31.82	39.81	37.81	39.08	11.76	32.06
$\text{MixTraining}_{\text{DA}}$	36.82	47.69	51.74	45.42	21.32	40.60
$\text{MixTraining}_{\text{GSM8K}}$	26.36	37.50	30.85	36.10	19.85	30.13
ICL	27.27	38.89	39.30	44.24	17.65	33.47
SciAgent	34.09	43.98	46.27	32.73	16.91	34.80
Retrieve with LLM						
-	31.36	40.28	40.80	48.71	16.91	35.61
Algebraic	28.64	38.43	37.81	42.52	13.97	32.27
PoT	30.45	25.93	22.89	44.01	14.71	27.60
CoT	30.45	39.81	40.30	51.68	16.17	35.68
ICL	28.64	43.52	39.80	48.08	17.65	35.54
RaR	27.27	25.93	23.38	43.70	17.65	27.59
EEDP	30.91	41.67	42.79	50.90	22.06	37.67
RCI	32.73	37.50	30.35	46.99	22.06	33.93
Retrieve with SLM						
-	41.82	58.33	56.22	66.09	21.32	48.76
Algebraic	37.27	56.94	56.22	61.55	24.26	47.25
PoT	37.27	29.17	35.82	59.51	19.12	36.18
CoT	40.91	61.11	52.74	70.40	26.47	50.33
ICL	32.27	54.63	55.22	67.89	16.18	45.24
RaR	32.27	39.35	38.81	51.61	20.59	36.53
EEDP	46.82	59.26	62.19	75.10	23.53	53.38
RCI	40.00	56.48	52.74	68.36	26.47	48.81
KIPG	**47.27**	**72.22**	**62.69**	**75.72**	**30.15**	**57.61**

Table 3: Results without providing the label knowledge. We explore the potential internal domain-specific knowledge under zero/few-shot scenarios. We also try different training source and skills ("Supervised Fine-Tuning"). Additionally, we turn to RAG considering the LLM itself and language model with smaller scale as the retriever respectively. Detailed baselines are available in Appendix A.

periments, calculated by regex from the sentences[3].

5.1 Main Results

Method	Indicator	Term	Medicine	AVG
Vanilla	44.20	65.36	51.08	53.55
CoT	45.59	63.40	57.55	55.51
PoT	36.76	52.29	48.20	45.75
Algebraic	69.93	52.78	55.40	52.56
RaR	7.35	26.80	23.74	19.30
RCI	51.47	58.17	51.80	53.81
EEDP	48.53	71.90	61.87	60.77
KIPG	**52.94**	**74.51**	**64.75**	**64.07**

Table 4: Cross domain performance given the oracle context.

[3]https://github.com/QwenLM/Qwen/blob/main/eval/evaluate_chat_gsm8k.py

Oracle Context We report the main results **given the gold document in legal domain as the context** in Table 2, i.e., the context article contains the necessary domain-specific knowledge for solving the problem. θ_G and θ_E are always Qwen2-7B models, while using 7B and 72B models as θ_C (the conclusion model) respectively. **1)** When we consider *Qwen2-7B-Instruct* as θ_C, the knowledge-intensive programs significantly improve the overall performance under all types of cases. Especially, KIPG on 7B model achieves equal or higher accuracy than Qwen-Max. The average accuracy is 69.86%, which is 8.77% relatively higher than the best baseline, EEDP. **2)** Then we prompt *Qwen2-72B-Instruct* to perform as θ_C. We omit the comparison with Qwen-Max under this setting. The accuracy of KIPG is mostly the highest over other methods.

Type	GPT-4	Code Qwen	Vanilla	SFT	Ours
Compensation	49.09	44.09	43.18	**52.27**	50.91
Tax	72.69	52.78	0.00	71.30	**73.15**
Other Fees	49.75	53.23	51.74	52.74	**64.68**
Penalties	87.94	90.37	91.31	**93.42**	92.17
Traffic Violations	55.88	61.76	1.47	64.71	**68.38**
AVG	63.07	60.45	37.54	66.89	**69.86**

Table 5: Comparison between different code generators during calculation. CodeQwen is a specific LLM developed for code relative tasks. "Vanilla" indicates the general Qwen2-7B-Instruct model. "SFT" indicates the Qwen model initialized with the GPT-4 generated codes, followed by DPO training.

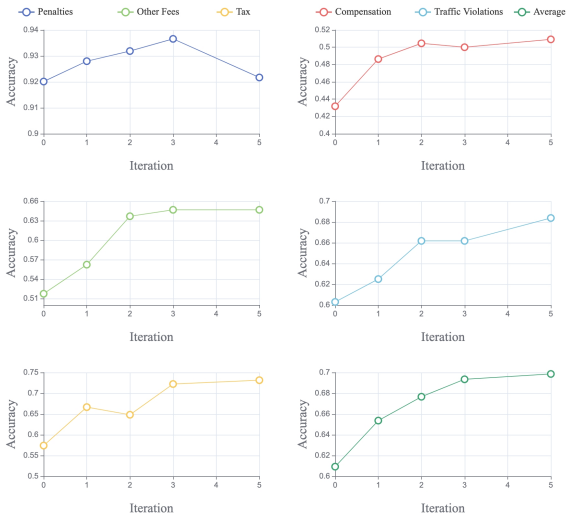

Figure 5: Accuracy over different training iterations under each type of cases.

Method	Indicator	Term	Medicine	AVG
Vanilla	23.53	32.03	34.53	30.03
CoT	26.47	24.84	38.13	29.81
PoT	29.41	56.86	46.04	44.10
Algebraic	26.47	30.72	43.88	33.69
RaR	35.29	28.10	26.62	30.00
RCI	32.35	45.10	48.92	42.12
EEDP	32.35	51.63	36.69	40.22
KIPG	**39.71**	**65.36**	**51.80**	**52.29**

Table 6: Cross domain performance in English conducted on Llama3 model with 8B parameters.

KIPG outperforms the baselines by a large margin. Additionally, by comparing the different model-scales, the 72B model is more skilled at leveraging the calculated domain-specific variables to generate the answers obviously. The average accuracy improves from 69.86% to 74.84%. It indicates the knowledge-intensive programs from smaller LLMs also benefit larger LLMs. **3)** Although KIPG is never trained on calculation cases of "Penalties" or "Traffic Violations", it also achieves the best results, indicating there are common features to exploit to solve the domain-specific calculation problems.

Without Oracle Context We conduct experiments without providing the golden knowledge under this setting. We prompt the models to either *recall relative rules with inner knowledge* ("zero/few-shot" and "supervised fine-tuning") or *retrieve the most helpful document with RAG techniques* ("retriever with LLM" and "SLM" such as BERT). The results are listed in Table 3. **1)** It exhibits better overall performance when retrieving the label documents using SLM than LLM. KIPG shows the highest accuracy for all types of cases. For "Tax", KIPG outperforms the second best by 11.11%. KIPG also achieves the highest average accuracy of 57.61%. **2)** Besides, fine-tuning improves the accuracy of calculation generally. RAG further helps the models to perform better by providing relative documents. We find that there are hallucinations when considering LLM as retriever.

5.2 Generalization across Domain

To investigate the cross domain generalization, we test KIPG on constructed medical calculation dataset after training in legal domain. The results are presented in Table 4, where "Term" indicates the terminologies and concepts in legal domain, which is relatively simple. Correspondingly, it is observed that most methods have achieved better performance on this type of cases. KIPG reaches the accuracy of 74.51% on "Term" and 64.07% on average, surpassing the second best baseline EEDP.

5.3 Iterations

We illustrate the accuracy changes over different iterations in Figure 5. Given that the types of "Other Fees", "Tax" and "Compensation" appear in the training dataset, their accuracy improves rapidly to a high level. While for "Traffic Violations", there is a significant optimization at iteration 5. But the iteration also leads to reduction sometimes especially for "Penalties", which is unobserved during training. Generally, the accuracy presents an ascending trend with the iteration increasing.

5.4 Code Models

We also compare different code generators within KIPG inference framework in Table 5. Although GPT-4 is the largest model, it is not good at gen-

Method	Compensation	Tax	Other Fees	Penalties	Traffic Violations	AVG
KIPG	50.45	**64.81**	**63.68**	**93.19**	**66.18**	**67.66**
w/o init	46.82(-3.63)	63.43(-1.38)	59.70(-3.98)	92.95(-0.24)	66.18(0.00)	65.82(-1.84)
w/o syntax	**51.82(+1.37)**	54.17(-10.64)	54.73(-8.95)	92.09(-1.10)	63.97(-2.21)	63.36(-4.30)
w/o correctness	49.55(-0.90)	55.56(-9.25)	53.73(-9.95)	92.48(-0.71)	65.44(-0.74)	63.35(-4.31)
w/o training	43.18(-7.27)	57.47(-7.34)	51.74(-11.94)	92.01(-1.18)	60.29(-5.89)	60.93(-6.73)

Table 7: Ablation study at the second training iteration. "w/o init" indicates removing the initialization of extraction LLM. When constructing preference pairs, we classify programs with grammar and comment errors as negative samples ("syntax review"). Additionally, programs with lower scores are treated as negative samples relative to those with higher scores ("correctness review"). "w/o syntax" and "w/o correctness" indicate removing the syntax review and correctness review respectively during the DPO data ranking. "w/o training" indicates the setting without training θ_G.

erating the knowledge-intensive programs in legal domain. CodeQwen[4] is specifically developed for code relative tasks, while the average accuracy is only 60.43%. From the above analysis, it can be seen that when it comes to domain knowledge, the generation mode of knowledge-intensive programs is different from that of ordinary codes. Our proposed training process achieves the best results on most types of cases, and it is better for some types when initializing θ_G by SFT.

5.5 Different Language and Base LLM

To verify the effectiveness of KIPG on different base LLMs and languages, we train Llama3-8B-Instruct (AI@Meta, 2024) on legal dataset in English and test it on translated English medical-domain dataset. The results are listed in Table 6. Surprisingly, PoT exhibits outstanding performance over other baseline methods under this setting, although it struggles from runtime errors in legal domain. KIPG has significant advantage among all approaches, with 8.19% margin over PoT in average.

5.6 Ablation Study

The ablation study is shown in Table 7. We list the results after **the second iteration**. KIPG exhibits the best overall accuracy, while removing the initialization of θ_E slightly reduces the performance from 67.66% to 65.82%. Then, it seems that code review on syntax and correctness have the same overall significance, while the accuracy of "compensation" is higher when ignoring syntax during DPO ranking. By removing training θ_G, the accuracy is not satisfactory, with the dramatic reduction of 11.94% on "Other Fees", and 6.73% accuracy drop on average.

[4]https://qwenlm.github.io/blog/codeqwen1.5

5.7 Logical Complexity

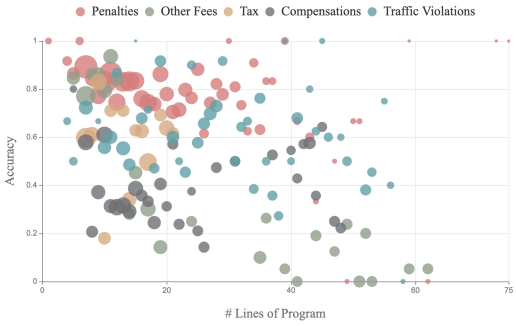

Figure 6: Number of lines in the programs versus the program accuracy under different types. The point size indicates the number of such programs.

We investigate the distribution of the accuracy versus the complexity of the documents and programs. Intuitively, a document describing complex rules and instructions corresponds to a program with more lines. Thus we utilize the number of program lines as a simplified indicator of the complexity of document. The distribution is illustrated in Figure 6. In general, more complex the document is, there are more lines in the program, and the corresponding accuracy is lower. Thus the plot shows a descending trend as the number of lines increases. Additionally, it is reasonable that the type of "Other Fees" covers a wide range along the x-axis, since it contains instructions of various fees. The programs of "Penalties" mostly scattered on the left-top section, which explains why its overall accuracy is relatively high.

6 Conclusion

In this paper, we present the task of domain-specific calculation involving knowledge of complex rules and conditions, and the usage of knowledge-intensive programs. Additionally, we also intro-

duce the framework named KIPG to solve the calculation task. We construct datasets in legal and medical domains with human labor and LLMs in a carefully designed pipeline. Empirical results illustrate the superiority of our method from multiple perspectives. KIPG outperforms other baselines with different base LLMs and different scales, in both Chinese and English.

Limitations

In this paper, we propose the framework of KIPG to solve the domain-specific calculation problems involving documents describing complex instructions and conditions. Although KIPG presents the superiority given gold document, but our method have not directly optimized the retrieval. It can be seen that the average performance drop is more than 10% without utilizing the label document. Obviously, all methods in our experiments dependent on RAG are all sensitive to the outcomes by retrieval. Optimizing the accuracy of retrieval is not our target in this paper, thus it may be remained for further studies.

Acknowledgements

This work was supported in part by National Natural Science Foundation of China (62441605, 62376243, 62037001, U20A20387), National Key Research and Development Program of China (2022YFC3340900), the StarryNight Science Fund of Zhejiang University Shanghai Institute for Advanced Study (SN-ZJU-SIAS-0010), Alibaba Group through Alibaba Research Intern Program, Project by Shanghai AI Laboratory (P22KS00111), Program of Zhejiang Province Science and Technology (2022C01044), the Fundamental Research Funds for the Central Universities (226-2024-00170).

References

AI@Meta. 2024. Llama 3 model card.

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. 2023. Palm 2 technical report. *Preprint*, arXiv:2305.10403.

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. *Preprint*, arXiv:2005.14165.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Preprint*, arXiv:2211.12588.

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. *Preprint*, arXiv:2110.14168.

Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. 2024. Rephrase and respond: Let large language models ask better questions for themselves. *Preprint*, arXiv:2311.04205.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui.

2024. A survey on in-context learning. *Preprint*, arXiv:2301.00234.

Duanyu Feng, Bowen Qin, Chen Huang, Zheng Zhang, and Wenqiang Lei. 2024. Towards analyzing and understanding the limitations of dpo: A theoretical perspective. *Preprint*, arXiv:2404.04626.

Neel Guha, Julian Nyarko, Daniel Ho, Christopher Ré, Adam Chilton, Alex Chohlas-Wood, Austin Peters, Brandon Waldon, Daniel Rockmore, Diego Zambrano, et al. 2024. Legalbench: A collaboratively built benchmark for measuring legal reasoning in large language models. *Advances in Neural Information Processing Systems*, 36.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Shima Imani, Liang Du, and Harsh Shrivastava. 2023. MathPrompter: Mathematical reasoning using large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 37–42, Toronto, Canada. Association for Computational Linguistics.

Zixuan Ke, Yijia Shao, Haowei Lin, Tatsuya Konishi, Gyuhak Kim, and Bing Liu. 2023. Continual pre-training of language models. *Preprint*, arXiv:2302.03241.

Geunwoo Kim, Pierre Baldi, and Stephen McAleer. 2023. Language models can solve computer tasks. *Preprint*, arXiv:2303.17491.

Yubo Ma, Zhibin Gou, Junheng Hao, Ruochen Xu, Shuohang Wang, Liangming Pan, Yujiu Yang, Yixin Cao, Aixin Sun, Hany Awadalla, and Weizhu Chen. 2024. Sciagent: Tool-augmented language models for scientific reasoning. *Preprint*, arXiv:2402.11451.

John J. Nay, David Karamardian, Sarah B. Lawsky, Wenting Tao, Meghana Bhat, Raghav Jain, Aaron Travis Lee, Jonathan H. Choi, and Jungo Kasai. 2023. Large language models as tax attorneys: A case study in legal capabilities emergence. *Preprint*, arXiv:2306.07075.

Arka Pal, Deep Karkhanis, Samuel Dooley, Manley Roberts, Siddartha Naidu, and Colin White. 2024. Smaug: Fixing failure modes of preference optimisation with dpo-positive. *Preprint*, arXiv:2402.13228.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

Pragya Srivastava, Manuj Malik, Vivek Gupta, Tanuja Ganu, and Dan Roth. 2024. Evaluating llms' mathematical reasoning in financial document question answering. *Preprint*, arXiv:2402.11194.

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

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R. Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search: Decoding diverse solutions from neural sequence models. *Preprint*, arXiv:1610.02424.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, Fan Yang, Fei Deng, Feng Wang, Feng Liu, Guangwei Ai, Guosheng Dong, Haizhou Zhao, Hang Xu, Haoze Sun, Hongda Zhang, Hui Liu, Jiaming Ji, Jian Xie, JunTao Dai, Kun Fang, Lei Su, Liang Song, Lifeng Liu, Liyun Ru, Luyao Ma, Mang Wang, Mickel Liu, MingAn Lin, Nuolan Nie, Peidong Guo, Ruiyang Sun, Tao Zhang, Tianpeng Li, Tianyu Li, Wei Cheng, Weipeng Chen, Xiangrong Zeng, Xiaochuan Wang, Xiaoxi Chen, Xin Men, Xin Yu, Xuehai Pan, Yanjun Shen, Yiding Wang, Yiyu Li, Youxin Jiang, Yuchen Gao, Yupeng Zhang, Zenan Zhou, and Zhiying Wu. 2023a. Baichuan 2: Open large-scale language models. *Preprint*, arXiv:2309.10305.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren,

Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. 2023b. Fingpt: Open-source financial large language models. *Preprint*, arXiv:2306.06031.

Wu Yiquan, Liu Yuhang, Liu Yifei, Li Ang, Zhou Siying, and Kuang Kun. wisdominterrogatory. Available at GitHub.

Hongbo Zhang, Junying Chen, Feng Jiang, Fei Yu, Zhihong Chen, Jianquan Li, Guiming Chen, Xiangbo Wu, Zhiyi Zhang, Qingying Xiao, Xiang Wan, Benyou Wang, and Haizhou Li. 2023. Huatuogpt, towards taming language model to be a doctor. *Preprint*, arXiv:2305.15075.

Yilun Zhao, Hongjun Liu, Yitao Long, Rui Zhang, Chen Zhao, and Arman Cohan. 2024. Financemath: Knowledge-intensive math reasoning in finance domains. *Preprint*, arXiv:2311.09797.

Zhi Zhou, Jiang-Xin Shi, Peng-Xiao Song, Xiao-Wen Yang, Yi-Xuan Jin, Lan-Zhe Guo, and Yu-Feng Li. 2024. Lawgpt: A chinese legal knowledge-enhanced large language model. *Preprint*, arXiv:2406.04614.

Çağatay Yıldız, Nishaanth Kanna Ravichandran, Prishruit Punia, Matthias Bethge, and Beyza Ermis. 2024. Investigating continual pretraining in large language models: Insights and implications. *Preprint*, arXiv:2402.17400.

A Baselines and Setting

CoT Wei et al. (2023) proposed CoT, which starts from "Let's think step by step".

ICL In-Context Learning (Dong et al., 2024) provides several examples within the contexts. For simplification, in our implementations, we provide 2 specific example of solving the task, ignoring the types of cases.

L1 and L2 We attempt to utilize the normalization methods to maintain the general capabilities during training on domain-specific corpus.

MixTraining Training on the mixture of different data source is proved to be beneficial for continual pre-training and SFT. We tried the combination of legal articles with GPT-4 extended calculation problems (denoted as "MixTraining$_{\text{DA}}$") and GSM8K (Cobbe et al., 2021) (denoted as "MixTraining$_{\text{GSM8K}}$").

SciAgent Ma et al. (2024) introduced SciAgent to retrieve several well-prepared functions (represented by python programs). In our experiment, there are no ready tools, thus we prepare the programs of numeric calculation instead (such as sum, subtraction, multiplication, division, maximum and minimum).

Algebraic Inspired by Imani et al. (2023), we utilize the mentioned algebraic formulation to replace the calculation by LLM itself.

PoT Chen et al. (2023) proposed PoT, which writes programs directly to solve the queries.

RaR Deng et al. (2024) allows LLMs to rephrase and expand questions and provide responses in a single prompt, named RaR.

EEDP Srivastava et al. (2024) proposed to prompt the LLM in the order of Elicit, Extract, Decompose, Predict, to solve the calculation task involving domain-specific knowledge, and the technique is called EEDP.

RCI Kim et al. (2023) utilized LLMs to execute computer tasks guided by natural language using a simple prompting scheme where the agent Recursively Criticizes and Improves its output (RCI).

Oracle Context Under this setting, we provide the label document as the context to the LLMs, thus they can directly generate the solution from the knowledge. For the Qwen2-72B model, we don't train a brand new code generator. Instead, the variables extracted by Qwen2-7B model are directly given, to investigate that whether the outcomes from smaller model could benefit the inference of larger LLM.

Without Oracle Context Under this setting, we assume that the label document is unobserved, then the LLMs have to recall the related knowledge either by internal parameters or retrieval from external sources. For SFT, we prepare legal article QA as the basic domain-specific corpus, based on which, we attempt to mix GSM8K and calculation instances generated by GPT-4 respectively into the training dataset. For "Retrieve with LLM", we adopt a LLM fine-tuned on article QA as the retriever, prompting it to recall the related article content given queries. For "Retrieve with SLM", we adopt a BERT model as the retriever to rank the articles according to embedding distances.

Cross Type For all scenarios involving training in legal domain, we keep the types of "Penalties" and "Traffic Violations" as the test types, which indicates that their calculation instances are unobserved during training. In this way, we can investigate the cross-type performance of the methods.

Cross Domain The medical dataset is kept for cross domain experiments. Specifically, we train the code generator in legal domain, then prompting it to generate the corresponding programs given documents in medical domain.

Llama in English To verify KIPG in English, we translate the related documents and instances into English with GPT-4, and develop KIPG based on Llama3-8B-Instruct. We also present the results of cross-domain performance in medical domain, since llama has rather limited knowledge about Chinese legal articles.

B Implementation Details

We use LoRA (Hu et al., 2021) for all training settings with rank 8. The learning rate is set to 5×10^{-5}, and training batch size as 16. For each iteration of DPO, we set the epochs to 3 and β to 0.1. The diverse beam search samples 8 programs for each document. We adopt Qwen2-7B-Instruct as the base LLM under most of the settings and download the parameters from HuggingFace [5]. During training, the calculation instances only involve the types of "Compensation", "Tax" and "Other Fees", while the other two types "Penalties" and "Traffic Violations" solely presents their articles. We leave the medical domain dataset for cross-domain experiments. To verify the effectiveness in English, we also adopt Llama3-8B-Instruct[6] for cross domain experiment. We adopt bf16 and train the models on 4 A100 80G GPUs.

C Efficiency Discussion

We discuss the efficiency of KIPG in this Section. First of all, we notice that the program generation is only applied in an offline manner. Thus the pre-generated programs can be directly called, given a query during inference. Additionally, we propose to filter the programs based on only a small set to reduce the calculation cost. Figure 7 illustrates

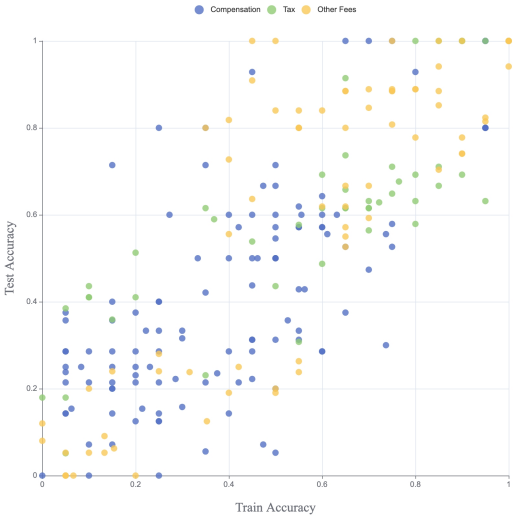

Figure 7: Accuracy of programs within the training set versus testing set.

the distribution of programs accuracy in the large-scale testing set (2050 samples) and a small-scale training set (330 randomly selected samples) under different types of cases. It can be seen that there is a high consistency between the two accuracy. It indicates that the calculation cost can be greatly reduced by remaining only the programs that works well on the small-scale training set. Finally, KIPG doesn't change the behaviour of normal text generation, thus it is adaptable to the acceleration techniques, such as vLLM and Flash-Attention.

D Several Findings

Larger beam size during training is important. We have tried different beam size during training, which means the scale of the exploration when generating knowledge-intensive programs. It is observed that larger beam size rapidly improves the conclusion performance with several iterations, and raises the upper limit after convergence. Larger exploration space directly increase the scale and diversity of the DPO training set, thus enhances the model performance.

Extraction LLM cannot be replaced by simple IE models. Extraction LLM θ_E extracts variables from the original query according to the comment of the programs. We attempt to replace LLM with a small language model for information extraction (IE), but it fails especially when the input variables requires simple general calculation and the text doesn't explicitly appear in the query. For example, the query presents the salary per year, while the program requires the salary per month. In this way,

[5] https://huggingface.co/Qwen/Qwen2-72B-Instruct
[6] https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

IE models cannot directly extract the correct value since the text span is not in the question.

Merging previous DPO instances does not help. We have also tried to merge the DPO data from previous iterations, to enlarge the scale of DPO training data and prevent over-fitting. However, the idea has only negative effects. We notice that the probabilities of both the chosen and rejected programs are descending, which may cause the model hard to generate the chosen one. Feng et al. (2024) and Pal et al. (2024) also reported similar phenomenon. We assume the failure is caused by the nature of DPO.

Diverse documents help even without corresponding calculation examples. There are only 33 articles for our training types. We notice that by adding the articles of "Compensation" and "Traffic Violations" at the first iteration, the initial significantly improves. The direct consequence of introducing new articles is the larger DPO data about only program syntax, while it leads to the ultimate improvement in correctness surprisingly.

E Guidelines of Dataset Construction

Knowledge Selection To improve the challenge of the task, we focus on practical domain knowledge, which mostly contains detailed instructions for different cases. For example, according to Article 13 of the Measures for the Payment of Litigation Costs, *the part of property cases not exceeding 10,000 yuan shall be paid 50 yuan per case. The portion exceeding 10,000 yuan to 100,000 yuan shall be paid at 2.5%. The portion exceeding 100,000 yuan to 150,000 yuan shall be paid at 2%.* It is non-trivial for LLMs to strictly follow the complex knowledge document to perform calculation given queries, especially considering that it requires domain knowledge to identify the corresponding conditions. For legal domain, we ask a team of lawyers with expertise in Chinese law to search satisfactory articles, given several examples. The articles cover a wide range of cases. For medical domain, we search for professional formulations and medication instructions on the Internet to ensure the correctness of the documents. Some statistics are provided in Table 1.

Query and Responses In our dataset, queries are questions from users, and responses are the corresponding explanations including an answer from consultants. The human annotators are instructed to question in a manner of speaking as casual as possible. The queries should ask a specific question and expect a single number as the answer, without providing the clues to the required domain-specific knowledge. For the responses, the annotators are prompted to write the reasoning as detailed as possible, which help to understand and check the results. A single numeric answer is expected at the end for each query. For the domain-specific documents describing a range of possible results, which is not rare, we turn to asking the maximum or minimum value in a general way. The results are format with 4 decimal places if they cannot be represented as integers. Additionally, we ask annotators to specify the unit of the answer, which is essential for evaluation.

Quality Review To precisely assessing the calculation results of LLMs, we also review the built dataset after the first round of annotation. Our review corrects the flaw including: 1) Incorrect question target, including asking multiple targets or requiring unrelated knowledge to the documents. 2) Wrong answers caused by incorrect calculation or low precision in intermediate variables. 3) Unclear explanations and reasoning. 4) Wrong citation to related articles.

Extension by LLM Since the human labor is expensive, we adopt GPT-4 to extend the dataset scale given hand-written instances as examples for each case. We provide detailed instructions and examples to GPT-4. We also specify the response with a explicit template "*According to {knowledge}, {reasoning}. {Analyse the case.} So the answer is {answer}.*" We also apply data review before adding the instances into the dataset.

F Detailed Legal Documents

Detailed types of legal cases are listed in Table 8.

G Components in Knowledge-Intensive Programs

Knowledge Source We add the knowledge source to the start of the top comment, such as "*Calculate the compensation fees for personal injury in accordance with Article 7 of the Interpretation of the Supreme People's Court on Several Issues Concerning the Application of Law in the Trial of Cases Involving Compensation for Personal Injury.*"

Input and Output Arguments In the top comment of the programs, we prompt the generator to

Type	SubType
Compensation	Funeral Allowance Burial Expenses Medical Malpractice Compensation Work Injury Benefits Work-related Death Funeral Allowance Death Compensation Disability Compensation Economic Compensation Lost Wages Compensation Compensation Payment Non-work-related Death Funeral Allowance and Consolation Payments
Tax	Personal Income Tax Taxable Income for Personal Income Tax Urban Maintenance and Construction Tax Stamp Tax Assessed Taxable Price of Taxable Vehicles Tobacco Leaf Tax Environmental Protection Tax Tax Arrears Penalty Cultivated Land Occupation Tax Vehicle Purchase Tax Interest on Debt During the Period of Delayed Performance
Other Fees	Unemployment Insurance Premium Deposit Trade Union Funds Nursing Expenses Application Fee Dependent's Living Expenses Litigation Costs Preparation Fees
Penalties	Penalties
Traffic Violations	Traffic Violations

Table 8: Types and sub-types in legal cases.

supplement with detailed descriptions to the input and output arguments, including the definitions, units and data-types in Python language. Different from query/terminology-oriented programs, our programs return all potential outcomes that can be calculated from the document in a dictionary, instead of a scalar representing a single concept or the direct answer.

Comment Augmentation To improve the logic consistency with the original document, we cite the sentences from the document before important calculation and "if" clauses. For example, the sentences start from "*The law states*" or "*According to the law*" for legal domain.

H Dataset Examples

We provide some examples of our constructed dataset in Table 9.

I Comparison with Related Work

Compared to LegalBench (Guha et al., 2024), our dataset places greater emphasis on combining calculation capability with domain knowledge (especially complex rules and logic), rather than focusing solely on simple mathematical problems that rely on little legal knowledge. Our tasks cover a wide range of areas, not just tax-related Q&A as in LegalBench. Not only does LegalBench, but also other open-source datasets, fall short in matching the strong reliance on domain-specific knowledge with complex rules and logical descriptions required by our task setting. This limitation diminishes the practicality and usability of these tasks, which is why we have developed our own dataset. We believe that the tasks and corresponding data we propose are more specialized and can provide significant research value for studies on integrating domain knowledge with numerical computation

Type	Query	Response
Compensation	Zhang, a migrant worker, was hospitalized for 5 days due to his infringement, but he could not provide proof of fixed income or the average income of the last three years. It is known that the average salary of employees in the same or similar industries where Zhang lived in the previous year was 80,000 yuan/year. How much should Zhang get for lost work?	Zhang has no fixed income and cannot provide proof of the average income of the last three years, which is calculated according to the average salary of employees in the same or similar industries in the previous year. Therefore, Zhang deserves 80,000 yuan /365 days x 5 days =1,095.89 yuan.
Penalties	A construction unit illegally built a small hydropower project on the Qinghai-Tibet Plateau, with a total investment of 5 million yuan. If the local people's government at or above the county level orders the construction to stop and restore to the original state, what is the maximum fine that the construction unit may face according to the law?	According to the fine range stipulated in Article 57 of the Qinghai-Tibet Plateau Ecological Protection Law of the People's Republic of China, the total investment amount of the construction project is determined to be 5 million yuan. According to laws and regulations, new small hydropower projects in violation of the provisions of this Law will be imposed a fine of not less than 1 percent but not more than 5 percent of the total investment of the construction project. Calculate the maximum fine: 5 million yuan $\times$5% = 250,000 yuan.
Other Fees	My wife and I are in the midst of divorce proceedings involving division of property. The total amount of our joint property is 3 million yuan. How much litigation fees should we pay in this case?	According to Article 13 (2) of the "Measures for the Payment of Litigation Costs", each divorce case pays 50 yuan to 300 yuan, if it involves the division of property and the total amount of property exceeds 200,000 yuan, it is paid in accordance with 0.5%. In this case, the total amount of property is 3 million yuan, and the part exceeding 200,000 yuan is 2.8 million yuan. Therefore, it is necessary to pay 2.8 million yuan x 0.5% = 1,400 yuan. With the minimum divorce case fee of 50 yuan, the total payment fee is 1400 yuan + 50 yuan = 1450 yuan.

Table 9: Dataset Examples.

capability.

J Notations

Notation	Description
θ_G	The program generator.
θ_E	The extraction model.
θ_C	The conclusion model.
$P = [f_1, f_2, \cdots, f_n]$	List of programs sampled from θ_G.
Q	The user query.
I, O	The input and output variables of a program.
$\tilde{P} \subseteq P$	The negative subset of P. Its programs are not executable because of errors.

Table 10: Descriptions of the notations in this paper.

We clarify our notations in Table 10.