Investigating the effectiveness of length based rewards in DPO for building Conversational Financial Question Answering Systems

Anushka Yadav[1*], **Sai Krishna Rallabandi**[2], **Parag Pravin Dakle**[2] **and Preethi Raghavan**[2]

[1]University of Massachusetts Amherst, [2]Fidelity Investments,
anushkayadav@umass.edu, {saikrishna.rallabandi, paragpravin.dakle, preethi.raghavan}@fmr.com

Abstract

In this paper, we address the numerical reasoning challenges of financial question-answering systems. We propose a two-stage approach where models first generate intermediate calculations and then produce the final answer. We perform two experiments to evaluate the performance of our approach. In the first, we compare single-step and multi-step approaches, demonstrating that incorporating intermediate calculations significantly improves numerical accuracy. In the second experiment, we compare traditional DPO and iterative DPO (iDPO) with length-regularized DPO. We show that while traditional DPO reduced parsing errors, it introduces verbosity; iDPO improves reasoning iteratively but faces diminishing returns. On the other hand, Length-regularized DPO reduces verbosity of intermediate calculation as well as enhances numerical accuracy across all models. These results highlight the potential of combining intermediate reasoning steps with domain-specific optimizations to build robust financial question-answering systems.

1 Introduction

Finance has emerged as a prominent area of focus for Large Language Models (LLMs)(Lee et al., 2024; Zhao et al., 2024; Desai et al., 2024; Nie et al., 2024; Xie et al., 2024b) since financial data (Zhao et al., 2022; Xie et al., 2024a) presents a unique set of complexities and challenges(Desai et al., 2024). These challenges arise from the intricate nature of financial texts and often require precise numerical calculations and an understanding of contextual dependencies that general-purpose LLMs struggle with. Common errors often include failure to perform precise numerical reasoning and generating inaccurate or irrelevant information due to an inadequate understanding of the context (Phogat et al., 2024).

In this paper, we focus on answering questions based on earnings reports(Chen et al., 2022b; Yang et al., 2023; Xie et al., 2024a; Zhao et al., 2022) and show that simple decomposition of the task of answering a financial question into two parts helps improve the reasoning capability. We build on the idea that arithmetic reasoning can benefit from generating a rationale (Wei et al., 2022; Cohen and Cohen, 2024) and fine-tune our LLMs to first output the arithmetic calculation required to answer the question. For this, we leverage Direct Preference Optimization (DPO)(Rafailov et al., 2024) and study the impact of introducing explicit rewards to incentivize the model to prioritize more accurate and contextually appropriate calculations. We then process the calculation and arrive at the final answer.

To demonstrate the effectiveness of our proposed approach, we perform extensive evaluations on the ConvFinQA(Chen et al., 2022b) dataset. Given the nature of numeric data and the annotation inconsistencies in the dataset, we also evaluate the approaches, considering a 0.1 percent threshold error. Our proposed Length-based regularization helps LLMs improve their performance and outperform GPT4o.

2 Length Regularization as Explicit Reward in DPO

Consider **R** to be a set of earnings reports, and r $\in$ **R** denotes an earnings report. Let q $\in$ **Q** denote a question about the report and y $\in$ **Y** denote the corresponding numeric answer to q. The task of question answering on earnings reports can be expressed as maximizing $P(y|q,r)$. In this paper, we fine-tuned our models to output a calculation c first and then arrive at the answer y based on c. Our approach can be expressed as maximizing $P(c|q,r)$.

A typical challenge in applying DPO to fine-

[*]Work completed during Fidelity Investments Internship

tuning LLMs is 'length bias'(Liu et al., 2024b; Park et al., 2024; Lu et al., 2024) - the tendency of models to generate unnecessarily long or convoluted outputs, especially when performing complex reasoning. In the context of our task, models fine-tuned using DPO without explicit length control exhibited this bias, frequently generating verbose and convoluted calculations for relatively straightforward financial questions. This over-generation introduces new errors, particularly in mathematical computations where simpler expressions are preferable.

Previous works have explicitly highlighted the importance of length-based regularization(Liu et al., 2024a; Park et al., 2024) in the context of classical RLHF pipelines and the DPO variants. Inspired by this, we introduced length regularization as an explicit reward in the DPO framework to mitigate this issue. Specifically, we penalize overly long calculations, encouraging the model to generate more concise outputs without sacrificing the necessary depth of reasoning. By incorporating this reward, we aim to balance the model's preference for providing comprehensive answers with the need for clarity and precision in financial contexts.

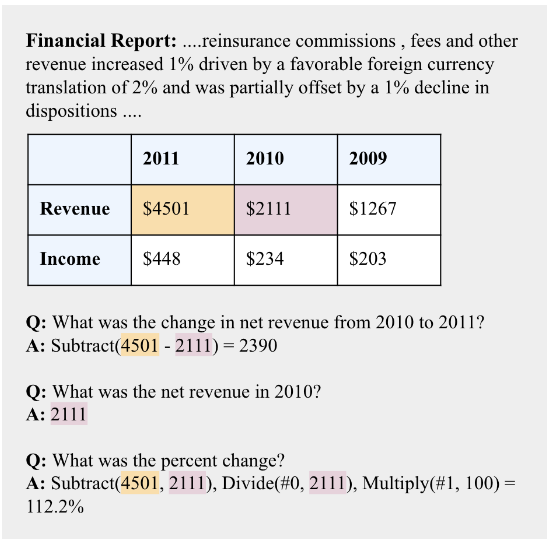
3 Experiments

In this section, we first describe our experimental setup and implementation details. We begin with off-the-shelf LLMs and fine-tune them using the following approaches: (a) Supervised Fine-tuning, (b) traditional DPO (one-step DPO), (c) Iterative DPO(Liu et al., 2024a; Fan et al.), and (d) DPO with Length Regularization as an explicit reward. Finally, we provide a detailed analysis of the effectiveness and impact of our proposed approach.

3.1 Experimental Setup

3.1.1 Data

ConvFinQA (Chen et al., 2022b) is a dataset designed to explore numerical reasoning in conversational question-answering tasks. Comprising 3,037 conversations derived from FinQA (Chen et al., 2022a), it emphasizes complex, multi-hop reasoning over financial reports from S&P 500 companies. A significant portion of the dataset features ambiguous questions with long dependencies, where content from previous answers is essential to resolve queries like *"What were they?"* or *"These years?"*. Figure 1 provides an example conversation from this dataset.



Financial Report: ....reinsurance commissions , fees and other revenue increased 1% driven by a favorable foreign currency translation of 2% and was partially offset by a 1% decline in dispositions ....

	2011	2010	2009
Revenue	$4501	$2111	$1267
Income	$448	$234	$203

Q: What was the change in net revenue from 2010 to 2011?
A: Subtract(4501 - 2111) = 2390

Q: What was the net revenue in 2010?
A: 2111

Q: What was the percent change?
A: Subtract(4501, 2111), Divide(#0, 2111), Multiply(#1, 100) = 112.2%

Figure 1: Example from **CONVFINQA** dataset

One major dataset challenge involves rounding percentage discrepancies during final answer generation. In most cases, rounding to the nearest integer resolves the mismatch; however, certain cases still result in inconsistencies. For instance, slight variations in how generative models handle precision and rounding lead to discrepancies even after applying conventional rounding techniques. To address this, we introduce a 0.1% error tolerance (0.1% ET) during evaluation to account for these minor differences. Importantly, this error-tolerant evaluation is applied exclusively to models that generate final answers directly rather than those that produce only calculation steps.

3.1.2 Models

We have employed the following models for our experiments: Mistral-7B-Instruct-v0.3[1], Llama-3.2-1B[2], Phi-3-Mini-128K[3], GPT-3.5 Turbo (*gpt-3.5-turbo-16k-0613*), and GPT-4o. We initially employed these models in a zero-shot setting to answer the numerical reasoning chains in the dataset conversations. Multiple fine-tuning experiments with various prompt configurations were conducted to facilitate an in-depth model analysis.

[1]https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3

[2]https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct

[3]https://huggingface.co/microsoft/Phi-3-mini-128k-instruct

Prompt	Model	No ET	0.1% ET
Final Answer Prompt	Mistral-7B	58.28	59.21
	Llama-3.2	57.21	59.17
	Phi-3	58.14	59.32
	GPT-3.5 Turbo	69.32	76.01
	GPT-4o	83.60	85.03
Calculation-Only Prompt	Mistral-7B	36.18	-
	Mistral-7B(SFT)	84.50	-
	Llama-3.2	35.11	-
	Llama-3.2(SFT)	82.10	-
	Phi-3	36.21	-
	Phi-3(SFT)	79.30	-
	GPT-3.5 Turbo	65.36	-
	GPT-4o	86.10	-

Table 1: Accuracies from Experiment 1: Single step vs Multi step under varying error tolerance settings. SFT denotes the supervised fintuned versions.

3.2 Experiment 01: Single step vs Multi step based approaches to final answer

We first built two systems to evaluate how the models that perform multi step numeric reasoning with calculation as the intermediate step compare to their out of the box single step variants. To do this, we have employed two prompts: (1) **Final Answer Prompt**, which directly generates the final answer for each question, and (2) **Calculation-Only Prompt**, which focuses exclusively on generating intermediate calculations required for reasoning. For conversational questions, the current question was provided along with contextual information derived from the previous question-answer pair within the conversation flow. The prompt details can be found in Appendix A.

3.2.1 Hyperparameters

We applied LoRA with 4-bit quantization. The fine-tuning parameters included a rank $r = 32$, an alpha value of $\alpha = 64$, and an initial learning rate of 5×10^{-6}, which decayed to 1.1×10^{-6} using a cosine schedule by the end of the training period. We set the batch size to 1 and employed a LoRA dropout of 0.05. The supervised fine-tuning (SFT) was carried out for a total of 1000 steps across all prompt variations.

3.2.2 Results and Observations

Table 1 summarizes the results of this experiment under varying error tolerance settings. For the Calculation-Only Prompt, no error tolerance was applied, as the generated calculations required an exact match with the gold-standard answer, including decimal precision.

It can be observed that SFT significantly enhances the ability of models to generate syntactically correct and logically consistent calculations. This finetuning step helped the models better adhere to the expected syntax and improved their overall reasoning.

3.2.3 Error Analysis

Errors in the outputs generated by the SFT Calculation-Only prompt based systems are typically deviations in the required format such as incorrect operator placement or incomplete expressions. Here are a few examples of such expressions:

- **add (multiply (0.09,3), 0.08)** : This expression has the presence of a nested multiplication and is in the wrong format.
- **41029, subtract (28422)**: This expression is in the wrong format. The correct format should have been subtract(41029, 28422)
- **divide (#0,5)**: This expression has ambiguous and unresolved variable #0.

3.3 Experiment 02: DPO vs iDPO vs Length regularized DPO

To alleviate the errors by the models in supervised fine tuning stage and further improve the reasoning, we implemented Direct Preference Optimization (DPO) as a refinement step. DPO was performed using poorly generated calculations from the SFT model alongside the gold-standard calculations in the dataset. A total of 600 poorly generated samples, extracted from the SFT outputs, were used for training. We have built three different systems that performed DPO:

1. A traditional one-step DPO system of 100 steps applied to the model to address errors observed from SFT.
2. **Iterative DPO (iDPO)**: A system consisting of two consecutive DPO sessions of 100 steps each. In iDPO, the model iteratively learns from errors in the previous session, progressively improving its ability to generate accurate calculations.
3. **Length-Regularized DPO (LDPO)**: Explicitly length-regularized DPO using the length regularization term *length_alpha* (Park et al., 2024) in the loss function. This approach penalizes overly long or verbose calculations to encourage conciseness.

Appendix C.1 shows the overall flow of training experiments.

3.3.1 Hyperparameters

For all DPO iterations, the learning rate was set to 5×10^{-6}, decaying to 9×10^{-7} at the 100th step and further to 2×10^{-9} at the 500th step. We used a batch size of 1, applied a LoRA dropout of 0.05, set the β value to 0.1, and assigned a value of 0.01 to *length_alpha*.

3.3.2 Results and Observations

Table 2 shows that DPO improved the overall performance across all models by addressing logical inconsistencies and refining calculation accuracy. Figure 2 illustrates some of the model responses from DPO experiments. It can be observed that traditional one step DPO successfully eliminated many parsing errors such as nested function calls. However, models finetuned with traditional DPO also introduced new challenges such as overly complex responses for simple queries. An example scenario is the DPO finetuned model generating [divide(1,B), multiply(A,#0)] instead of the simpler divide(A,B). The models often produced unnecessarily verbose responses, increasing complexity and reducing parsing accuracy.

The models employing iDPO improved on the traditional DPO based counterparts by addressing some of these issues, as can be seen by superior performance across the model variants employed. iDPO however led to repetitive errors in subsequent iterations and therefore the improvements stalled after the first iteration.

Length Regularized DPO model variants were the most appropriate and led to the best performance across the model variants. While Mistral 3 variant with LDPO achieved the highest accuracy, it has to be noted that all of the model variants that employed LDPO outpeformed their other DPO counterparts.

Model	OOB	SFT	DPO	iDPO	LDPO
GPT-4o	86.10	-	-	-	-
GPT-3.5 Turbo	65.36	-	-	-	
Mistral 3	36.18	84.5	84.5	87.01	**87.2**
Llama 3.2	35.11	82.1	82.8	86.05	87.14
Phi 3	36.21	79.3	81.1	84.03	86.19

Table 2: Accuracies from Experiment 2: DPO vs iDPO vs LDPO

Gold Calculation: subtract(multiply(7963, 36%), multiply(10014, 27%))
Generated Calculation: multiply(7963, 36%), multiply(10014, 27%), subtract(#0, #1) ❌

Gold Calculation: add(19400, 2500), add(#0, 1100), add(#1, 200), divide(19400, #2)
Generated Calculation: add(19400, 2500), add(#0, 1100), add(#1, 200), divide(#2, 4), divide(19400, #3) ❌

Gold Calculation: add(15636, 14710)
Generated Calculation: subtract(15688, 14663), divide(#0, 14663), add(15636, 14710) ✅

Gold Calculation: subtract(279, 100)
Generated Calculation: 279, subtract(#0, 100) ✅

Figure 2: Examples of complex responses generated after the initial DPO step.

4 Limitations

- **Dependency on Clean Annotations**: Our approach relies on the availability of calculation annotations in the training data. Real-world financial datasets might not contain annotations or have noisy or incomplete annotations which could hinder the model's performance.
- **Potential Oversimplification**: While length regularization effectively reduces verbosity, it may lead to oversimplification of reasoning paths. This could result in the model missing nuanced steps required for answering complex financial queries.
- **Generalization Challenges**: Length-based regularization may not generalize well across other domains or question-answering tasks that require different forms of reasoning or context interpretation.

5 Conclusion

In this study we have investigated the effectiveness of employing length regularzed DPO in multi step fashion towards answering financial questions. Our first experiment demonstrated that introducing intermediate calculations before generating final answers improved multi-step reasoning and accuracy. Our second experiment showed that traditional DPO reduced parsing errors but introduced verbosity, while iterative DPO (iDPO) iteratively improved reasoning but faced diminishing returns. Length-regularized DPO emerged as the most effective approach, balancing concise outputs with reasoning depth, reducing verbosity, improving numerical accuracy, and enhancing efficiency across all tested models. These findings underscore the importance of domain-specific strategies to improve reliability and precision in financial question-answering systems.

References

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2022a. Finqa: A dataset of numerical reasoning over financial data. *Preprint*, arXiv:2109.00122.

Zhiyu Chen, Shiyang Li, Charese Smiley, Zhiqiang Ma, Sameena Shah, and William Yang Wang. 2022b. Convfinqa: Exploring the chain of numerical reasoning in conversational finance question answering. *Preprint*, arXiv:2210.03849.

Cassandra A Cohen and William W Cohen. 2024. Watch your steps: Observable and modular chains of thought. *arXiv preprint arXiv:2409.15359*.

Akshar Prabhu Desai, Ganesh Satish Mallya, Mohammad Luqman, Tejasvi Ravi, Nithya Kota, and Pranjul Yadav. 2024. Opportunities and challenges of generative-ai in finance. *arXiv preprint arXiv:2410.15653*.

Ying Fan, Fei Deng, Yang Zhao, Sahil Singla, Rahul Jain, Tingbo Hou, Kangwook Lee, Feng Yang, Deepak Ramachandran, and Qifei Wang. Iterative dpo with an improvement model for fine-tuning diffusion models.

Jean Lee, Nicholas Stevens, Soyeon Caren Han, and Minseok Song. 2024. A survey of large language models in finance (finllms). *arXiv preprint arXiv:2402.02315*.

Jie Liu, Zhanhui Zhou, Jiaheng Liu, Xingyuan Bu, Chao Yang, Han-Sen Zhong, and Wanli Ouyang. 2024a. Iterative length-regularized direct preference optimization: A case study on improving 7b language models to gpt-4 level. *arXiv preprint arXiv:2406.11817*.

Wei Liu, Yang Bai, Chengcheng Han, Rongxiang Weng, Jun Xu, Xuezhi Cao, Jingang Wang, and Xunliang Cai. 2024b. Length desensitization in directed preference optimization. *arXiv preprint arXiv:2409.06411*.

Junru Lu, Jiazheng Li, Siyu An, Meng Zhao, Yulan He, Di Yin, and Xing Sun. 2024. Eliminating biased length reliance of direct preference optimization via down-sampled kl divergence. *arXiv preprint arXiv:2406.10957*.

Yuqi Nie, Yaxuan Kong, Xiaowen Dong, John M Mulvey, H Vincent Poor, Qingsong Wen, and Stefan Zohren. 2024. A survey of large language models for financial applications: Progress, prospects and challenges. *arXiv preprint arXiv:2406.11903*.

Ryan Park, Rafael Rafailov, Stefano Ermon, and Chelsea Finn. 2024. Disentangling length from quality in direct preference optimization. *arXiv preprint arXiv:2403.19159*.

Karmvir Singh Phogat, Sai Akhil Puranam, Sridhar Dasaratha, Chetan Harsha, and Shashishekar Ramakrishna. 2024. Fine-tuning smaller language models for question answering over financial documents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10528–10548, Miami, Florida, USA. Association for Computational Linguistics.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Qianqian Xie, Weiguang Han, Xiao Zhang, Yanzhao Lai, Min Peng, Alejandro Lopez-Lira, and Jimin Huang. 2024a. Pixiu: A comprehensive benchmark, instruction dataset and large language model for finance. *Advances in Neural Information Processing Systems*, 36.

Qianqian Xie, Dong Li, Mengxi Xiao, Zihao Jiang, Ruoyu Xiang, Xiao Zhang, Zhengyu Chen, Yueru He, Weiguang Han, Yuzhe Yang, et al. 2024b. Open-finllms: Open multimodal large language models for financial applications. *arXiv preprint arXiv:2408.11878*.

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

Huaqin Zhao, Zhengliang Liu, Zihao Wu, Yiwei Li, Tianze Yang, Peng Shu, Shaochen Xu, Haixing Dai, Lin Zhao, Gengchen Mai, et al. 2024. Revolutionizing finance with llms: An overview of applications and insights. *arXiv preprint arXiv:2401.11641*.

Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. 2022. Multihiertt: Numerical reasoning over multi hierarchical tabular and textual data. *arXiv preprint arXiv:2206.01347*.

A Appendix A

Prompt for Experimentation

The following is the prompt used for `Final Answer Only Prompt` in our experiments:

Final Answer Only Prompt

You are a highly intelligent bot. You can have conversations with the user to answer a series of questions over a financial report. Later questions may depend on previous questions to answer.
Here is the financial report: $report
I will be asking questions over it next. Understood?

The following is the prompt used for `Calculation-Only Prompt` in our experiments:

Calculation-Only Prompt

You are an expert financial analyst who analysis financial reports of various organizations. You will be given a financial report of an organization and your manager will be asking a series of connected questions based on that report.
Your objective is to:

- Understand the given financial report and its associated information provided in tables.
- Answer the given questions turn by turn using the information from the report and relevant context from your responses to previous turns.
- The answer to a question is the calculation over the values stated in the report.The calculation might depend on answers to previous questions in the series.

Criteria for Answering:

1. Use the *Operations Table* below to perform any operations needed to answer the question.
2. Calculation should include the operations (if any) performed for answering the question. Include all the calculations needed to answer the current question in response.
3. If the answer is just getting extracted from the report, output the answer directly.
4. Use `#` to refer to the result of a previous step where necessary. Example: `add(1,2)`, `multiply(#0,3)`.
5. Respond with Calculation only, do not give the final answer.

Operations Definition Table:

- `add(number1, number2)` → add two numbers: `number1 + number2`
- `subtract(number1, number2)` → subtract two numbers: `number1 - number2`
- `multiply(number1, number2)` → multiply two numbers: `number1 · number2`
- `divide(number1, number2)` → divide two numbers: `number1 / number2`
- `exp(number1, number2)` → `number1 ^ number2`
- `greater(number1, number2)` → boolean comparison: `number1 > number2`

Your response should look like this for each question:
Calculation:
Example Report and Questions: $ex
Here is the financial report: $report
Answer the questions based on the report. Understood?

B Appendix B

Hybrid Reasoning Prompt Configuration

The following is the prompt used for `Hybrid Reasoning Prompt` in our experiments:

Hybrid Reasoning Prompt

You are an expert financial analyst who analyzes financial reports of various organizations. You will be given a financial report of an organization, and your manager will ask a series of connected questions based on that report.
Your objective is to:

- Understand the given financial report and its associated information provided in tables.
- Answer the given questions turn by turn using the information from the report and relevant context from your responses to previous turns.
- Handle questions where the answer may depend on previous answers in the series.

Criteria for Answering:

1. Use the *Operations Table* below to perform any operations needed to answer the question.
2. Include all calculations performed in your response to provide the final numerical answer.
3. Use `#` to refer to the result of a previous step where necessary. Example: `add(1,2)`, `multiply(#0,3)`.
4. Do not generate further content after outputting the answer.

Operations Definition Table:

- `add(number1, number2)` → add two numbers: `number1 + number2`
- `subtract(number1, number2)` → subtract two numbers: `number1 - number2`
- `multiply(number1, number2)` → multiply two numbers: `number1 · number2`
- `divide(number1, number2)` → divide two numbers: `number1 / number2`
- `exp(number1, number2)` → `number1 ^ number2`
- `greater(number1, number2)` → boolean comparison: `number1 > number2`

Your response should look like this for each question:
Calculation:
Answer:
Example Report and Questions: $ex
Here is the financial report: $report
Answer the questions based on the report. Understood?

Table 3 summarizes the performance of models on the Hybrid Reasoning Prompt, which combines intermediate calculation generation with final answer generation. This approach

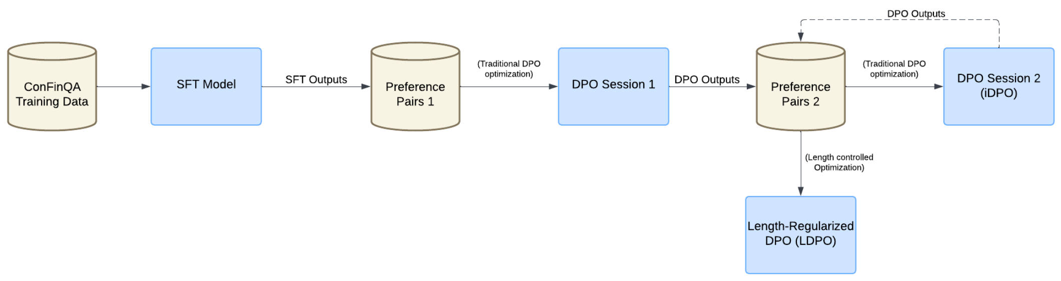


Figure 3: Flowchart showing the different training stages carried out during experimentation.

Prompt	Model	No ET	0.1% ET
Hybrid Reasoning Prompt	Mistral-7B	63.53 (48.89)	65.19
	Llama-3.2	61.22 (48.27)	64.59
	Phi-3	62.97 (47.77)	65.02
	GPT-3.5 Turbo	61.34 (60.2)	62.61
	GPT-4o	85.97 (84.81)	86.31

Table 3: Baseline performance of models Hybrid Reasoning Prompt setting

consistently improved final answer accuracy across all models compared to the Final Answer Prompt, highlighting the benefits of incorporating reasoning steps. The only exception was GPT-3.5, which occasionally failed to produce a final answer after completing the calculation, leading to a slight drop in accuracy. Also, the calculation execution accuracy, shown in parentheses, remains lower due to parsing issues and formatting errors.

C Appendix C

C.1 Training Flowchart

Figure 3 shows how training was carried out during experimentation. First, Supervised Fine-tuning (SFT) was performed using the ConvFinQA dataset. This model is then fine-tuned using DPO with incorrect predictions of the SFT model and ground truth from the ConvFinQA dataset. After a single DPO fine-tuning session, we explore two training variations: (1) Iteratively applying multiple DPO sessions and (2) Length-Regularized DPO (LDPO).