Learning to Decode Collaboratively with Multiple Language Models

Shannon Zejiang Shen **Hunter Lang** **Bailin Wang** **Yoon Kim** **David Sontag**
Massachusetts Institute of Technology
{zjshen, hjl, bailinw, yoonkim, dsontag}@mit.edu

Abstract

We propose a method to teach multiple large language models (LLM) to collaborate by interleaving their generations at the token level. We model the decision of which LLM generates the next token as a latent variable. By optimizing the marginal likelihood of a training set under our latent variable model, the *base LLM* automatically learns when to generate itself and when to call on one of the "assistant" language models to generate, all without direct supervision. Token-level collaboration during decoding allows for a fusion of each model's expertise in a manner tailored to the specific task at hand. Our collaborative decoding is especially useful in cross-domain settings where a generalist base LLM learns to invoke domain expert models. On instruction-following, domain-specific QA, and reasoning tasks, we show that the performance of the joint system exceeds that of the individual models. Through qualitative analysis of the learned latent decisions, we show models trained with our method exhibit several interesting collaboration patterns, e.g., template-filling.[1]

1 Introduction

Techniques that combine the generations of multiple large language models (LLMs) at decoding time have benefits ranging from faster decoding speed (Leviathan et al., 2023), to more controllable generations (Liu et al., 2021; Yang and Klein, 2021), to more coherent, less repetitive text (Li et al., 2023a), and even enabling a large model to be "tuned" by combining its generations with those of a smaller model from the same family (Liu et al., 2024). A parallel thread of work has aimed to equip language models with the ability to infuse external tools into their generations, with the goal of incorporating outside knowledge and capabilities (Mialon et al., 2023). Language models are able to produce more faithful and accurate generations when equipped with external APIs (Schick et al., 2023; Qin et al., 2023; *i.a.*), search engines or retrievers (Izacard et al., 2022; Asai et al., 2023; Nakano et al., 2021; *i.a.*), or code executors (Gao et al., 2023; *i.a.*).

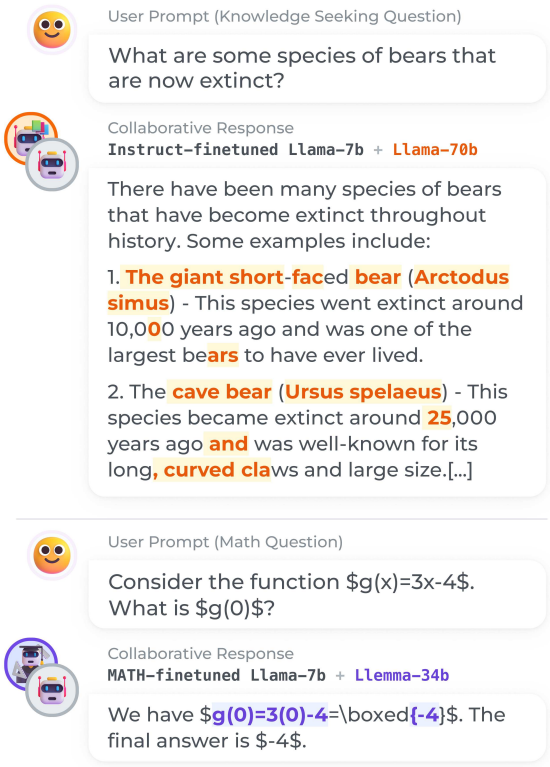

Figure 1: Example generations of our method, Co-LLM. *Top*: the base model generates the answer template and uses a larger LLAMA model to fill in factual knowledge; *Bottom*: the base model uses a math-specialized model as an "API" for computation. The assistant model generated the highlighted tokens because the base model learned to *defer* generation at those locations.

While powerful, these methods all require a prescription on *how* to combine the models and *when* to use the tools, either via specific formulas for combining the logits of multiple models, or through (weak) supervision on where to insert tool/API calls in the training data. In this work, we explore a different type of model combination

[1]Code: https://github.com/clinicalml/co-llm

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 12974–12990

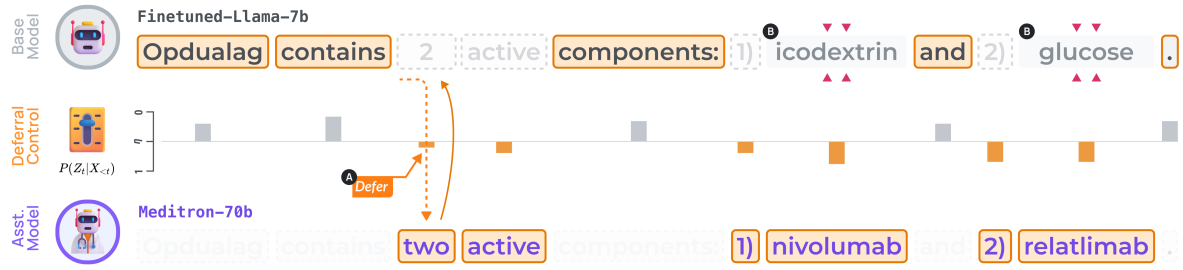

Figure 2: Illustration of the decoding procedure in Co-LLM, where a `base` (LLAMA-7B) and `assistant` model (MEDITRON-70B) collaborate to generate a correct response for a medical question. For each token, the `deferral control` predicts the probability of switching to the assistant model to decode the next token given the context: it *defers* when the probability is above some threshold η (indicated by A), and uses the decoded token as the context (highlighted with orange border). When using the base model alone, it may make factual mistakes (indicated by B); Co-LLM learns to use the assistant model at these positions to produce correct generations.

where the models learn to interleave their generations token-by-token. Each token is generated by one model, so the models *collaborate* to generate a token sequence together. We represent the decision of which LLM generates the next token as a latent variable, assuming no direct supervision on the decision of which model to use at each decoding step. This enables an effective collaboration pattern for a given task to be learned organically from data.

Figure 1 shows example generations from our method, Co-LLM. In the top example, LLAMA-7B collaborates with LLAMA-70B on instruction-following by generating a list template for the answer and then calling on LLAMA-70B to fill in each element of the list. Using the larger model as an assistant allows the smaller model to make effective use of a larger knowledge base and focus its efforts on learning the correct "scaffolding" for instruction responses. In the bottom example, LLAMA-7B collaborates with LLEMMA-34B (a domain-specific math model, Azerbayev et al., 2023) by treating the latter as an API call to fill in parts of a LaTeX formula. In both cases, the model predicts when to call the assistant by itself, behavior it *learns* from training without direct supervision on which contexts suit the assistant model well. This enables the emergence of qualitatively different collaboration methods (e.g., learning to scaffold, calling the large model as an API) based on what the task demands.

Section 2 describes our latent-variable model for collaboration during decoding, and Section 3 describes the training and decoding procedures for Co-LLM in this model. In, Section 4, we evaluate Co-LLM on instruction-following, mathematical reasoning, and domain-specific question-answering tasks. Our results indicate that teaching models to collaborate improves performance across all of these tasks compared to using the individual models alone, and can sometimes match or exceed the performance of fine-tuning the large model. By using chain-of-thought reasoning (Wei et al., 2022), Co-LLM can also be applied to classification tasks, where our experiments show that it boosts performance by enabling improved reasoning capability. Our results show that Co-LLM is especially useful in cross-domain settings where a generalist base LLM learns to invoke domain expert models and that Co-LLM can be effectively combined with other ensemble models, such as Mixture of Experts models (Shazeer et al., 2017).

2 Latent-Variable Framework for Collaborative Generation

Given a set of LMs with different expertise or sizes, we propose a latent-variable framework that enables their collaboration in a cost-efficient way. The framework centers around a finetunable *base model*, which itself is a relatively small LM. It decides which other *assistant models* (which are typically larger and/or more specialized models) to use per token. When the base model calls on an assistant to generate the next token, we say it *defers generation* for that token.

To generate a sequence of tokens $(X_1, \ldots, X_T)$, we represent the choice of which model generates token X_t as a discrete latent variable $Z_t \in \{0, 1, \ldots, M\}$, where $i = 0$ denotes the base model, and $i \in \{1, \ldots, M\}$ refers to the i-th of M assistant models. We assume access to the conditional distributions $P_i(X_t|X_{<t})$, $i \in \{1, \ldots, M\}$, of the assistant models,[2] and full access to the base model. Using these distributions, we represent the

[2]Conditionals can be obtained via either locally deployed models or remote API calls. At training time, we only require access to the assistant model's probability for the ground-truth tokens, not the full conditional distribution.

joint sequence-level likelihood as:

$$P(X, Z) = \prod_{t=1}^{T} \Big(P_\theta(Z_t|X_{<t}) P_{Z_t}(X_t|X_{<t}) \Big) \quad (1)$$

The (learned) categorical distribution P_θ models the token-level discrete decision Z_t. For efficiency, each Z_t is conditionally independent of $Z_{<t}$ given $X_{<t}$ in our design. The latent variable Z_t is designed in the same spirit as the defer variable in Mozannar and Sontag (2020) and classical Mixture-of-Experts models (Jordan and Jacobs, 1994) or ensemble models (Saunders et al., 2019).

Unsupervised Learning. In practice, the token level decisions Z_t are unknown, and collecting human annotation is not scalable. Our latent-variable framework offers a natural way of handling the issue with unsupervised learning. In particular, we aim to optimize the following marginal likelihood,

$$P(X) = \prod_{t=1}^{T} \Big(\sum_{Z_t=0}^{M} P_\theta(Z_t|X_{<t}) P_{Z_t}(X_t|X_{<t}) \Big), \quad (2)$$

which can be computed efficiently during training due to the conditional independence structure.

Collaborative Decoding. During inference time, our goal is to find the best sequence X along with the best decision Z on which assistant LM to use.

$$\hat{X}, \hat{Z} = \arg\max_{X,Z} P(X, Z) \quad (3)$$

The exact $\arg\max$ in Eq. (3) is intractable, so we follow the common practice of using the greedy strategy for decoding both Z_t and X_t in a token-by-token, autoregressive manner (see Fig. 2 for an example). In greedy decoding, for each token position, we first choose $\hat{Z}_t = \arg\max_{Z_t} P_\theta(Z_t|X_{<t})$ to determine which model to decode from, then decode greedily from that model: $\hat{X}_t = \arg\max_{X_t} P_{\hat{Z}_t}(X_t|X_{<t})$. Compared with standard greedy decoding for a single LM, decoding in this model is performed in collaboration, coordinated by P_θ. An alternative strategy to $\arg\max$ decoding is to marginalize out Z_t:

$$\hat{X}_t = \arg\max_{X_t} \sum_{Z_t} P_{Z_t}(X_t|X_{<t}) P_\theta(Z_t|X_{<t})$$

which closely aligns with the marginal likelihood training objective. However, this requires calling all $M + 1$ models every token, slowing down decoding. Our empirical results show that greedily choosing Z_t based on $P_\theta(Z_t|X_{<t})$ performs well and enables interpretable collaboration, since each token is generated by a single model.

Remark. The design of the collaborative decoding is natural from the probabilistic modeling perspective, and in this work we mainly focus on important empirical questions: how should we parameterize P_θ so that it can be learned in a data-efficient way; can the base model learn how to cooperate with (larger or domain-specific) assistant models with only access to conditional probabilities (no internal hidden states or weights); and what kind of collaboration pattern can be induced if the collaboration is learned without supervision? The latent-variable framework allows us to answer these questions via the exposed interpretable variable Z_t.

3 Co-LLM: Learning to Decode Collaboratively with LMs

In this work, we focus on the basic case where we only have one base model and one assistant model. In this setting, the base model needs to make a binary decision of whether to generate from itself or defer to the assistant model, i.e., $Z_t \in \{0, 1\}$. For clarity, we use P_{base} and P_{asst} to denote the base and assistant model, respectively. In the rest of this section, we explain our design for the parameterization of the model selector, and the training and inference procedure of the joint model.

3.1 Modeling $P_\theta(Z_t|X_{<t})$

Since we have full access to the base model, we build P_θ on top of the base model for parameter efficiency. Specifically, we represent θ as a linear binary classification head in the last layer. Formally, if $h_t(X_{<t}) \in \mathbb{R}^d$ is the base model's last hidden state at time step t for inputs $X_{<t}$, and $\theta \in \mathbb{R}^d$ is the weight vector, we set:

$$P_\theta(Z_t|X_{<t}) = \sigma(\langle \theta, h_t(X_{<t}) \rangle),$$

where $\sigma(\cdot)$ is the sigmoid function. This introduces only d new parameters to the base model, where d is the base model's hidden dimension size.

3.2 Training

Our training objective minimizes the negative log-marginal likelihood $-\sum_{t=1}^{T} \log P(X_t|X_{<t})$, where

$$P(X_t|X_{<t}) = P_{\text{base}}(X_t|X_{<t}) P_\theta(Z_t = 0|X_{<t}) + P_{\text{asst}}(X_t|X_{<t}) P_\theta(Z_t = 1|X_{<t}) \quad (4)$$

is the likelihood of the next token after marginalizing out the latent variable Z_t. The training procedure updates both θ and the base model parameters; it requires the forward pass of both the base

and assistant model to obtain next-token probabilities, and the backward pass of the base model for gradient computation. This implies that any commercial LLMs or locally deployed LMs exposing next-probability output can be used as the assistant LMs. The marginal likelihood aligns well with the typical pretrained objective of maximizing the next-token probs. Moreover, offloading "difficult tokens" can potentially alleviate hallucination issues of the base model, thus leading to better generalization even without much help from the assistant model, as we will show in the experiments.

Initialization of θ. In our experiments, we found that appropriately initializing θ helps encourage the base LM to quickly switch from generating by itself to generating collaboratively. Instead of collecting direct supervision for Z_t values to initialize θ, we use weak supervision to collect initial Z_t values, use these pseudolabels to initialize the parameters θ, then allow the Z_t's to change during the training procedure.

Intuitively, the prime location for $Z_t = 1$ is when the assistant model correctly predicts the target token X_t but the base model does not. To that end, we set the pseudolabels for Z_t as:

$$\hat{Z}_t := \mathbb{1}[X_t = \arg\max_{v \in \mathcal{V}} P_{\text{asst}}(v|X_{<t}) \wedge X_t \neq \arg\max_{v \in \mathcal{V}} P_{\text{base}}(v|X_{<t})], \quad (5)$$

and initialize the d parameters θ by maximizing the likelihood of $\log P_\theta(\hat{Z}_t|X_{<t})$ while holding the rest of the base model fixed. In our experiments, we compare to a baseline that combines this initialization with the usual language-model fine-tuning loss; our results show that the marginal likelihood objective leads to better performance by enabling the base model to learn better Z_t values from data (see ablation results in Section 5.2).

3.3 Decoding

In initial experiments, we found that performance of the joint model is sensitive to the choice of Z_t; due to the exposure of Z_t in our latent-variable framework, we can impose extra priors over Z_t for better performance. Specifically, we follow the general greedy decoding strategy, and set a threshold η for decoding Z_t:

$$\hat{Z}_t = \mathbb{1}[P_\theta(Z_t = 1|X_{<t}) > \eta], \quad (6)$$

which means that when $P_\theta(Z_t = 1) > \eta$, we execute the assistant model to predict the next token. The hyperparameter η is picked via grid search on a small validation set per dataset. The choice of threshold for the decoding probability also allows for inference-time control over the amount of collaboration, in contrast with other approaches such as DExperts (Liu et al., 2021), and our performance degrades gracefully as the threshold increases.

4 Experimental Setup

In our experiments, we fine-tune models for specific tasks and we test the models in-domain, comparing the end-task performance between Co-LLM and multiple single- or multi-model baselines. We test on 4 datasets ranging from instruction-following to solving expert problems, trying to understand when and how model collaboration can be beneficial. We investigate the collaboration between different models (e.g., between Llama models of multiple scales, and between models fine-tuned on different domains). Overall, we find that Co-LLM can learn a successful collaboration between different base and reference models, leading to better results than tuning base models alone.

Models Used. Our primary experiments are concerned with whether smaller models can collaborate with expert models that have been specialized to different domains. In Section 5.1, we experiment with collaboration between the finetuned LLAMA-7B and the LLEMMA family (Azerbayev et al., 2023, finetuned for math and reasoning), as well as the MEDITRON family (Chen et al., 2023, finetuned for biomedicine). It is also possible to use differently-sized models from the same family with Co-LLM: in Section 5.2 we experiment with the 7B and the 70B model from the same LLAMA-2 family (Touvron et al., 2023) as the base and assistant model, respectively.

Datasets. We train on the full Tülu v2 mix data (Wang et al., 2023) for instruction following, GSM8k (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021), each with 7.5k samples, for reasoning and math problem solving, and the BioASQ (Tsatsaronis et al., 2015) (4.7k samples) for medical question answering. We train and test the model on the corresponding data and evaluation suites separately.

Evaluation. We only compare greedy decoding results for all models, as they are commonly used in real-world applications. We evaluate the instruction following model using the AlpacaEval (Li

et al., 2023b) benchmark; we use the GPT-4 annotator and compute the win rate of the testing model as judged by `GPT-4-0613` when compared to the outputs produced by `Davinci-003`. For GSM8k, following Wang et al. (2023),we extract the last numerical answer from the model's output, and calculate the exact match of the model prediction on a 200-example subset of the test set. The MATH dataset provides 5 levels of math problems from 7 categories: similar to Wu et al. (2023), we skip the geometry category since it largely relies on Asymptote code, and sample 25 questions per level per category for evaluation, resulting in a 750-example subset. We adopt the prompting and evaluation code from Azerbayev et al. (2023) and Lewkowycz et al. (2022), extracting the last generated math after "The answer is". [3] The BioASQ comes with 330 test examples of 4 categories: factoid, list, summary, and yes/no questions, evaluated using strict accuracy (SAcc.), F1, accuracy (Acc.), and Rouge-2 (R2). We test on 310 examples (saving 20 for validation) and reimplement the evaluation code.[4]

4.1 Baselines

Single models. The performance of the base and assistant models can inform whether the learned collaboration is beneficial. We report 0-shot performance of the original untuned models and their finetuned counterparts. The same data and hyperparameters are used for model finetuning; for 70B models, we fine-tune using QLoRA (Dettmers et al., 2023) with the hyperparameters in Appendix A.

Other collaborative models. We use two collaborative strategies from the literature. Contrastive Decoding (Li et al., 2022b; O'Brien and Lewis, 2023, CD) combines the output of the untuned "expert" (e.g., a 70B model) and "amateur" (e.g., a 7B model) models by subtracting their logits and sampling from the resulting distribution. We follow the setup in O'Brien and Lewis (2023), setting the same $\alpha = 0.1$ and $\beta = 0.5$, and use unmodified LLAMA-70B as the expert model and LLAMA-7B as the amateur model.[5] Proxy Tuning (Liu et al., 2024, PT) proposes to approximate finetuning a (large) base model $\mathcal{M}$ by composing the outputs of smaller expert $\mathcal{M}^+$ and anti-expert models $\mathcal{M}^-$ with $\mathcal{M}$. We include CD and PT results as ways to enhance untuned models and to simulate finetuning 70B models, respectively. Both CD/PT require calling the smaller and larger models at each time step. In contrast, Co-LLM may generate multiple tokens from the base model before calling the large model, which can be seen as a form of speculative decoding (Leviathan et al., 2023). For example, for a sequence of length L, Proxy Tuning makes L calls[6] to the large model and $2L$ calls to the small model, whereas Co-LLM makes fL calls to the large (assistant) model and L calls to the small (base) model, where $0 \leq f \leq 1$ is the empirical frequency of deferral, the percent of $Z_t = 1$ tokens.

Ablated Co-LLM. We consider different variants of Co-LLM to verify the necessity of a learned model selector P_θ. First, we consider two simple heuristics as model selectors: Co-LLM-Random randomly chooses the base or the assistant model to produce a token with probability $p = 0.5$; Co-LLM-Greedy runs both models in parallel for each token and selects the token with the higher probability from either model. This is a strong baseline since it requires observing next-token probabilities from both models at every decoding step. Similar to our default setting, the base model is finetuned on target datasets while the assistant model is frozen.

Weakly-supervised Co-LLM. Finally, we consider a different weakly supervised training procedure for Co-LLM. This baseline is inspired by the process used to derive tool-use labels in Toolformer (Schick et al., 2023): the weak supervision for when to call the tool is chosen and *fixed* before updating the language model parameters. Specifically, the training procedure is two-stage: first, we collect pseudo-labels $\hat{Z}_t$ using Eq. (5). Second, we jointly train $P_\theta(Z_t|X_{<t})$ and the base model by optimizing the weighted sum of $\log P(\hat{Z}_t|X_{<t})$ and the usual language modeling loss. This trains the base model to defer to the assistant when $\hat{Z}_t = 1$ while also fine-tuning it on the full training set. In contrast, our marginal likelihood training Section 3.2 only uses the $\hat{Z}_t$ values to *initialize* θ and allows the Z_t values to evolve during training.

[3] We use greedy decoding for GSM8k and MATH; some literature refers to this as "maj@1".

[4] See details in Appendix A.

[5] In their paper, O'Brien and Lewis (2023) use a 1.5B parameter amateur model; as this model is not released, we use the 7B model instead. Different from their paper, we use 0- or 1-shot prompting in congruence with our other results.

[6] Here each "call" corresponds to a token-level decoding step, which is a sensible unit with which to measure inference latency as the context encoding portion can be easily parallelized and thus grows slowly with respect to context length in small batch regimes.

Math and reasoning tasks	GSM	MATH
LLEMMA-7B	4.0	2.0
LLEMMA-34B	14.5	6.3
Finetuned LLAMA-7B	34.5	7.6
Finetuned LLAMA-70B (QLoRA)	52.5	11.7
PT (LLEMMA-34B + LLAMA-7B)	30.0	20.9
PT (LLEMMA-34B + LLEMMA-7B)	58.5	23.7
CO-LLM-7B + LLEMMA-7B	40.0	17.2
CO-LLM-7B + LLEMMA-34B	43.5	24.5

BioASQ tasks	Factoid	List	Yes/No	Summ.	Avg.
MEDITRON-7B	0.00	2.7	70.4	18.6	22.9
MEDITRON-70B	17.2	16.1	80.2	21.1	33.7
Finetuned LLAMA-7B	23.7	13.8	76.5	18.1	33.0
Finetuned LLAMA-70B (QLoRA)	24.7	20.7	75.3	21.1	35.5
PT (MEDITRON-70B + LLAMA-7B)	26.9	10.7	80.2	7.3	31.3
PT (MEDITRON-70B + MEDITRON-7B)	26.9	23.5	82.7	11.0	35.6
CO-LLM-7B + MEDITRON-7B	17.2	16.0	72.8	19.8	31.4
CO-LLM-7B + MEDITRON-70B	21.5	18.6	81.5	20.6	35.6

Table 1: CO-LLM enables collaboration between models trained on different domains: using the expert model trained for the domain (e.g., LLEMMA for math and reasoning, and MEDITRON for biomedical tasks) during decoding boosts performance compared to the fine-tuned base model, and sometimes performs even better than fine-tuned LLAMA-70B. Proxy Tuning (Liu et al., 2024, PT) only performs well when all three of its component models ($\mathcal{M}, \mathcal{M}^+, \mathcal{M}^-$) are pretrained on the same domain mix.

5 Results

5.1 Collaboration across domains

Table 1 shows that CO-LLM enables collaboration between LLAMA and domain-specific models and that this collaboration improves performance compared to the individual models themselves. For example, being able to utilize LLEMMA as an assistant leads to improved performances on math and reasoning tasks. On the MATH dataset, even invoking a small, 7B-scale LLEMMA assistant (17.2) outperforms fine-tuned LLAMA-7B (7.6), fine-tuned LLAMA-70B (11.7), and LLEMMA-34B (6.3). Similarly, cooperation with MEDITRON models leads to performance gains on some BioASQ subtasks (e.g., List, Summary), and outperforms fine-tuned LLAMA-7B, fine-tuned LLAMA-70B, and base MEDITRON-70B on average.

In addition, CO-LLM with LLAMA-7B and LLEMMA-34B can achieve similar performance as fine-tuned LLEMMA-7B, which scores 43.5 on GSM8k and 23.5 on MATH. CO-LLM allows the base 7b model access to collaborate with a domain expert (LLEMMA-34B), which surprisingly leads to similar performance as performing a large amount of domain-specific fine-tuning plus further task-specific fine-tuning on the base model (finetuned LLEMMA-7b).

These results suggest that CO-LLM enables a modular approach to continued pretraining and task-specific finetuning: one can pretrain a large model on a domain-specific corpus, then fine-tune smaller models with CO-LLM to leverage the knowledge from the larger models and attain improved performance on the downstream tasks.

Comparison against Proxy Tuning. While PT and our work are differently motivated and constructed, they both leverage multiple models during generation. Table 1 also provides an in-depth comparison between the two methods in the context of combining models from different domains. PT only performs well when all three models ($\mathcal{M}, \mathcal{M}^+$, $\mathcal{M}^-$) are pretrained on the same domain mix (compare, e.g. "LLEMMA + LLAMA" to "LLEMMA + LLEMMA"). This is due to the implicit assumption that the difference between the base model $\mathcal{M}$ and a hypothetical, tuned version of the base model is the same as the difference between the smaller expert $\mathcal{M}^+$ and the anti-expert $\mathcal{M}^-$. Our results show that CO-LLM is more effective at enabling collaboration between models from different domains. PT also requires more calls to the larger model, thus resulting in slower inference. CO-LLM makes fewer calls to both large and small models.

In the following section, we show that in addition to enabling collaboration across domains, CO-LLM also allows collaboration across model scales.

5.2 Collaboration across scales

Table 2 shows that using CO-LLM leads to a successful collaboration between the base and assistant models of different sizes within the LLAMA family. In particular, compared to using the (fine-tuned) base model LLAMA-7B alone, CO-LLM-7B + LLAMA-70B, which occasionally calls the unmodified assistant model during decoding, consistently achieves significantly better performance across all tasks and datasets (2.6, 10.5, 7.5, and 3.3 absolute improvements for the 4 datasets, respectively). CO-LLM is sometimes better than the QLoRA finetuned assistant model (on MATH and BioASQ), suggesting that our method can effectively combine the best of the models and achieve better performance than the "sum" of their parts. Training with CO-LLM does not hurt the perfor-

		AlpacaEval	GSM	MATH	BioASQ[a]				
		(% Win)	(Acc.)	(EM)	Factoid (SAcc.)	List (F1)	Yes/No (Acc.)	Summ. (R2)	Avg.
Untuned	LLAMA-7B	-	7.0	0.3	4.3	4.9	71.6	17.2	24.5
	LLAMA-70B	11.6	13.5	2.1	11.8	14.9	77.8	18.6	30.8
	LLAMA-70B+7B (CD)	-	11.5	1.3	11.8	9.0	71.6	17.5	27.5
Finetuned	LLAMA-7B (Finetuned)	69.3	34.5	7.6	23.7	13.8	76.5	18.1	33.0
	LLAMA-70B (QLoRA)	78.6[b]	52.5	11.7	24.7	20.7	75.3	21.1	35.5
	LLAMA-70B+7B (PT)	72.3	52.5	17.3	29.0	16.8	85.2	21.3	38.1
Collaboration	CO-Random	46.3	17.0	6.1	6.5	1.9	30.9	17.5	14.3
	CO-Greedy	64.1	38.0	8.1	29.0	16.6	76.5	20.2	35.6
	Weak Supervision	56.7	40.0	12.3	22.6	14.6	80.2	17.5	33.7
	Co-LLM-7B (Base Only)	70.6	33.0	6.4	20.4	11.2	79.0	18.1	32.2
	Co-LLM-7B + LLAMA-70B	71.9	45.0	15.1	24.7	18.0	82.7	20.4	36.5

[a] For BioASQ, we use 1-shot prompting for LLAMA-7B, -70B, and CD experiments to inform the model of the output format.
[b] We report the results obtained by Ivison et al. (2023) in Table 5.

Table 2: Results of using Co-LLM for LLAMA models of different sizes. Occasionally[7] calling the Llama-70B model to generate a few tokens, Co-LLM-7B is able to significantly outperform the finetuned Llama-7B model in all tasks, and sometimes even performs better than the QLoRA-finetuned Llama-70B model.

Math and reasoning tasks	GSM	MATH
MISTRAL-7B	21.5	7.2
MIXTRAL-8×7B (MoE)	38.5	16.2
Finetuned MISTRAL-7B	51.0	13.9
Co-LLM MISTRAL-7B + MIXTRAL-8×7B	**57.0**	**20.0**

Table 3: Co-LLM can be applied among models of different architectures like a dense LLM (MISTRAL-7B) and a sparse Mixture of Experts (MoE) model (MIXTRAL-8×7B). The learned collaboration leads to strong performance improvements on both GSM and MATH tasks.

mance of the base model: when we prohibit using the assistant model during inference, performance is comparable to the base model finetuned with the usual language modeling objective (LLAMA-7B (Finetuned)): for example, getting 33.0 and 6.4 for GSM8k and MATH, respectively. Co-LLM thus degrades gracefully as the amount of deferral is changed, which we explore further in §5.4.

Comparing with the model collaboration baselines, we show that interleaving two model generations is not a trivial task: randomly switching between the two models lead to worse than single-model performances in Co-LLM-Random. Even if running two models in parallel, Co-LLM-Greedy does not consistently yield better performance than using either model alone, and in some cases, they are worse (e.g., in the case of GSM8k). As discussed in Section 5.1, Proxy Tuning performs very well when used with models of different sizes in the same family, but Co-LLM also performs well despite using far fewer calls to the language models. Finally, the Toolformer-style baseline with fixed, weakly supervised values of Z_t leads to overall worse performance compared to Co-LLM, indicating the benefits of our latent variable formulation and marginal likelihood loss, which allow the best deferral patterns to be learned from data.

5.3 Collaboration across architectures

Since Co-LLM only assumes access to token logprobs, it can easily be used for collaboration between models of different architectures. For example, Table 3 shows that Co-LLM can be adopted to collaborate between a dense model (MISTRAL-7B) and a sparse MoE model (MIXTRAL-8×7B), and the joint model achieves strong accuracy gains compared to either the finetuned MISTRAL-7B model or the MIXTRAL-8×7B model. These results indicate that Co-LLM is still useful when used together with other model combination methods like MoE.

5.4 Qualitative Analysis

The exposure of the interpretable variable Z_t in our latent-variable framework makes it easy to visualize the model composition patterns. As shown in Figs. 1 and 2, our unsupervised training can induce an interesting template-filling collaboration strategy: the base model learns to generate a template with slots that requires deeper knowledge (in question answering) or reasoning (in math problems) for assistant models to fill in. Table 4 further illustrates how the collaboration between the assistant

[7] See Section 5.4 for a detailed analysis of the frequency of invoking the assistant model and the end performance.

Deferral frequency $f = 0$	$f = 0.3$	$f = 0.4$	$f = 0.9$	$f = 1.0$
We have $a^3 = 5^3 = 125$, and $a^2 = 5^2 = 25$, so $a^3 \cdot a^2 = \boxed{125}$. The final answer is 125.	We have $a^3 = 5^3 = 125$, and $a^2 = 5^2 = 25$, so $a^3 \cdot a^2 = \boxed{3000}$. The final answer is 3000.	We have $a^3 = 5^3 = 125$, and $a^2 = 5^2 = 25$, so $a^3 \cdot a^2 = \boxed{3125}$. The final answer is 3125.	We can use the power rule to simplify this expression. We have that $a^3 \cdot a^2 = a^{3+2} = a^5$. Now we can substitute $a = 5$ to get $5^5 = 3125$. Therefore, the final answer is $\boxed{3125}$. The final answer is 3125.	### 1 Given a mathematics problem, determine the answer. Simplify your answer as much as possible. You can use latex to format your answer and you should state your final answer as "The final answer is $(final - answer)$." Problem:

Table 4: Model answers with different rates of deferral to the question "Evaluate the expression $a^3 \cdot a^2$ if $a = 5$" (answer: 3125). In this example, we use the finetuned LLAMA-7B as the base model and LLEMMA-34B model as the assistant. We show the 0-shot model answers at different deferral frequencies f, with yellow background to indicate the token is produced by the assistant model.

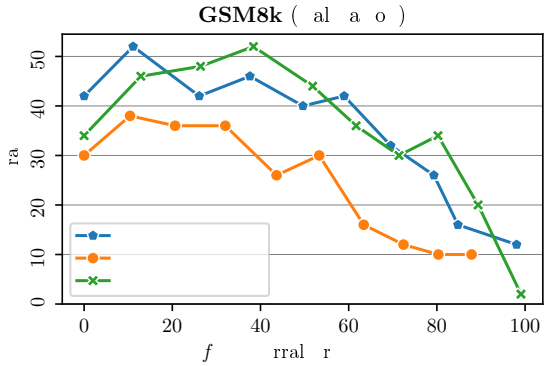

Figure 3: Performance of Co-LLM at different frequencies of deferral on GSM8k. There exists an optimal f that the joint model achieves better performance than using either of them alone. Similar trend is observed in MATH and BioASQ, shown in Fig. 4 in Appendix.

(LLEMMA-34B) and base (fine-tuned LLAMA-7B) model leads to correct responses while either model alone fails to do so. When the base model is tasked to solve the math question alone (i.e, deferral frequency $f = 0$), it fails to produce a valid answer, generating 125 rather than 3125. As we increase the frequency of deferral by lowering the threshold η, we observe Co-LLM starts to invoke the assistant model to generate latex code and compute the results, and when $f = 0.4$, the joint model produces the correct answer. More deferral in this case does not yield better generations. When we fully rely on the assistant model ($f = 1$), since it is not tuned or aligned, it produces no helpful solutions.

We also evaluate the joint model at different deferral frequencies on small validation sets for GSM8k, MATH, and BioASQ, and plot the results in Fig. 3 and Fig. 4 in the Appendix. Across different domains and scales, the models exhibit a similar concave performance curve: at some optimal deferral frequencies, the joint model can achieve better performance than using either of the models alone, with decreased performance at both extremes. The optima vary across different datasets, corresponding to the different patterns of invoking assistant models (e.g., API-call or "leading" style). In practice, one can pick the proper η to balance the accuracy and the efficiency/cost.

6 Related Work

Learning to compose models. Composing expert models has been a recurring strategy to improve large models, and the way of composing expert models is largely related to the underlying learning settings. Mixture of Experts (Shazeer et al., 2017; Jiang et al., 2024; Dai et al., 2024; Xue et al., 2024, MoE) requires that all experts are trained simultaneously using the same data. In contrast, post-hoc methods assume the presence of pretrained language models but usually pose implicit constraints on the model families to compose. Proxy Tuning (Liu et al., 2024, 2021) works best when all models are all pretrained on the same data mixture; PHATGOOSE (Muqeeth et al., 2024) requires that all models are LoRA-finetuned (Hu et al., 2021) models; Contrastive Decoding (Li et al., 2022b, CD) requires an amateur model, which is not clear for tasks such as math reasoning. Co-LLM is more flexible, mainly because our base model *learns* to interact with assistant models instead of relying on prescribed strategies. Our experiments also indicate that Proxy Tuning (concurrent work) only performs well when all models are pretrained in the same domain, whereas Co-LLM can effectively combine general and domain-specific models. Compared to CD and Proxy Tuning, Co-LLM also makes fewer calls to the language models at inference time, as described in Section 4.1.

Perhaps most similar to our work are speculative decoding (Leviathan et al., 2023) and CombLM (Ormazabal et al., 2023). Like our approach, speculative decoding generates some tokens with one model and some tokens with another, but our method differs in that the goal is to improve generation quality rather than to sample more quickly from a large model by approximating its generations with a smaller one. CombLM also learns to combine the probability distributions of multiple models, but their combination is not time-varying, and they mainly demonstrate wins on perplexity.

Our approach could be seen as a special case of Toolformer (Schick et al., 2023), where the assistant model is the tool/API called by the base model. However, our latent variable formulation allows fine-grained deferral decisions to evolve over the course of fine-tuning, whereas Toolformer derives a fixed dataset prescribing which tool gets called (and where) *before* training. Co-LLM enables varying the frequency of calling the assistant model, whereas Toolformer has no provision for flexibly adjusting the amount of tool use at inference time.

Is this just Mixture of Experts? In Mixture of Experts (MoE) LLMs (Zhou et al., 2022; Li et al., 2022a; Jiang et al., 2024; Xue et al., 2024; Dai et al., 2024), the goal is to train a model that can be partially executed during inference time. A common choice in MoEs is to decompose FFN layers (feed forward network) in a transformer into modular "experts". These experts are subnetworks—they cannot be used standalone, and the experts are typically expected to have the same size (parameter count). MoE also requires gradient access to all experts and assumes every expert has access to the same training data.

In contrast, we aim to learn how to collaborate with existing off-the-shelf models. First, we do not assume gradient access to assistant models, and we only fine-tune the base model. This saves in training cost, but poses a more difficult learning problem, since we only assume access to logprobs from the assistant model. Co-LLM is able to learn effective collaboration patterns between models despite this limited access to the assistants. Second, because of our less restrictive assumptions on assistant model access and architecture, Co-LLM can be applied in more constrained scenarios. For example, Co-LLM can still be applied when the assistant is trained on proprietary data (e.g., due to copyright issues (Duetting et al., 2023)). Third, our framing allows flexible collaboration between models of different sizes or even architectures. Table 3 shows performance gains from combining a dense model with an MoE model using Co-LLM, indicating that the gains are orthogonal to MoE. Finally, in our approach, each "expert" is a full-fledged LLM rather than a sub-network. This allows for interpretable generation patterns (as illustrated in Figs. 1 and 2), as well as flexible inference-time tradeoffs between how much each model is used (e.g., in Table 4, we show Co-LLM can work well at different frequencies of deferral).

Learning to defer. A large body of literature focuses on the related problems of *prediction with rejection* (Cortes et al., 2016), where the goal is to train a model that can predict on some inputs and decline to predict on others, and *learning with deferral* (Mozannar and Sontag, 2020; Mozannar et al., 2023), where the goal is to train a model to make predictions on some inputs and defer to a human expert on others. Mohri et al. (2023) combine prediction with example-level rejection and LLMs on the text decontextualization problem. We use a latent variable formulation inspired by Mozannar et al. (2023), initially developed for learning to defer to a human expert on classification problems. We replace the human expert with a fixed LLM assistant model and extend the loss to allow for token-level rather than sequence-level deferral.

7 Conclusion

We propose Co-LLM, a latent variable approach that learns to collaboratively decode using multiple LMs. Co-LLM can interleave token generations from different LMs in patterns most suitable for the task at hand (e.g., scaffolding the generation, or using the other model like an API), and learns the collaboration pattern organically from data. We empirically show that Co-LLM can produce generations of better quality compared to using either of the models alone, in tasks ranging from math reasoning, medical question answering, and instruction following. The latent "defer" variable offers a flexible and interpretable way for adjusting the frequency for invoking other LMs at inference time without re-training. In the future, we plan to extend Co-LLM to integrate more than two LMs and investigate potentially more complex collaboration strategies emerging from this setting.

Limitations

As shown in Fig. 4 and Table 6 of the Appendix, the optimal deferral threshold may be different across different datasets and models. Co-LLM thus requires picking the deferral threshold per task, which can be inconvenient in practice. However, the threshold also enables inference-time control over the amount of collaboration. Second, not every deferral matters: for some positions, the assistant model may generate an identical token as the base model does. This suggests the development of more fine-grained control of deferral strategies, potentially via more sophisticated modeling of deferral model parameters θ.

Another limitation in our method comes from fully relying on an assistant model at some point in decoding. For example, if the assistant model is not well-tuned or aligned, it may unintentionally break the generation due to occasional mistakes. As shown in the example below, one erroneous token might lead to a cascade of errors, causing repetition patterns or generating irrelevant content. One future work is to develop a more robust deferral strategy that allows backtracking when the assistant model fails to generate a proper response.

```
Here's a recipe for Kubdari, a traditional
Georgian dish:
Ingredients:
* 1 lb ground beef
* 1 onion, finely cho pped
* 2 cloves garlic, minced
* 1 cup chopped parsley
* 1 cup chopped cilantro
* 1 cup chopped dill
* 1 cup chopped ...
```

[...repeating the same pattern...]

Acknowledgements

SS and DS were supported by the National Science Foundation (NSF award no. IIS-2205320 Conceptualizing ML for Dynamic Information Retrieval of EHR Notes). HL was funded by an NDSEG fellowship. BW and YK were partially supported by MIT-IBM Watson AI and an Amazon award. We thank CloudBank (Norman et al., 2021) for supplying computational resources, which is supported by the National Science Foundation under award #1925001. And thanks to Jacob Andreas, Lucas Torroba Hennigen, Hussein Mozannar, Ilker Demirel, Andreas Haupt, Tiwalayo Eisape, Alex Gu, Ruochen Zhang, and Doug Downey for feedback on the draft of this paper.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2023. Llemma: An open language model for mathematics.

Zeming Chen, Alejandro Hernández Cano, Angelika Romanou, Antoine Bonnet, Kyle Matoba, Francesco Salvi, Matteo Pagliardini, Simin Fan, Andreas Köpf, Amirkeivan Mohtashami, Alexandre Sallinen, Alireza Sakhaeirad, Vinitra Swamy, Igor Krawczuk, Deniz Bayazit, Axel Marmet, Syrielle Montariol, Mary-Anne Hartley, Martin Jaggi, and Antoine Bosselut. 2023. Meditron-70b: Scaling medical pretraining for large language models.

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.

Corinna Cortes, Giulia DeSalvo, and Mehryar Mohri. 2016. Learning with rejection. In *Algorithmic Learning Theory: 27th International Conference, ALT 2016, Bari, Italy, October 19-21, 2016, Proceedings 27*, pages 67–82. Springer.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *ArXiv*, abs/2401.06066.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Paul Duetting, Vahab Mirrokni, Renato Paes Leme, Haifeng Xu, and Song Zuo. 2023. Mechanism design for large language models. *arXiv preprint arXiv:2310.10826*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset.

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. *arXiv preprint arXiv:2106.09685*.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023. Camels in a changing climate: Enhancing lm adaptation with tulu 2.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts.

Michael I Jordan and Robert A Jacobs. 1994. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214.

Anastasia Krithara, Anastasios Nentidis, Konstantinos Bougiatiotis, and Georgios Paliouras. 2023. Bioasq-qa: A manually curated corpus for biomedical question answering. *Scientific Data*, 10(1):170.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models.

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A. Smith, and Luke Zettlemoyer. 2022a. Branch-train-merge: Embarrassingly parallel training of expert language models.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and M. Lewis. 2022b. Contrastive decoding: Open-ended text generation as optimization. pages 12286–12312.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2023a. Contrastive decoding: Open-ended text generation as optimization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12286–12312, Toronto, Canada. Association for Computational Linguistics.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023b. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. 2024. Tuning language models by proxy.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021. DExperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6691–6706, Online. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. 2023. Augmented language models: a survey.

Christopher Mohri, Daniel Andor, Eunsol Choi, and Michael Collins. 2023. Learning to reject with a fixed predictor: Application to decontextualization. *arXiv preprint arXiv:2301.09044*.

Hussein Mozannar, Hunter Lang, Dennis Wei, Prasanna Sattigeri, Subhro Das, and David Sontag. 2023. Who should predict? exact algorithms for learning to defer to humans. In *International Conference on Artificial Intelligence and Statistics*, pages 10520–10545. PMLR.

Hussein Mozannar and David Sontag. 2020. Consistent estimators for learning to defer to an expert. In *International Conference on Machine Learning*, pages 7076–7087. PMLR.

Mohammed Muqeeth, Haokun Liu, Yufan Liu, and Colin Raffel. 2024. Learning to route among specialized experts for zero-shot generalization. *arXiv preprint arXiv:2402.05859*.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Michael Norman, Vince Kellen, Shava Smallen, Brian DeMeulle, Shawn Strande, Ed Lazowska, Naomi Alterman, Rob Fatland, Sarah Stone, Amanda Tan, et al. 2021. Cloudbank: Managed services to simplify cloud access for computer science research and education. In *Practice and Experience in Advanced Research Computing*, pages 1–4.

Sean O'Brien and Mike Lewis. 2023. Contrastive decoding improves reasoning in large language models.

Aitor Ormazabal, Mikel Artetxe, and Eneko Agirre. 2023. Comblm: Adapting black-box language models through small fine-tuned models. *arXiv preprint arXiv:2305.16876*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Danielle Saunders, Felix Stahlberg, A. Gispert, and B. Byrne. 2019. Domain adaptive inference for neural machine translation. *ArXiv*, abs/1906.00408.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

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

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, et al. 2015. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC bioinformatics*, 16(1):1–28.

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

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.

Yiran Wu, Feiran Jia, Shaokun Zhang, Hangyu Li, Erkang Zhu, Yue Wang, Yin Tat Lee, Richard Peng, Qingyun Wu, and Chi Wang. 2023. An empirical study on challenging math problem solving with gpt-4.

Fuzhao Xue, Zian Zheng, Yao Fu, Jinjie Ni, Zangwei Zheng, Wangchunshu Zhou, and Yang You. 2024. Openmoe: An early effort on open mixture-of-experts language models. *ArXiv*, abs/2402.01739.

Kevin Yang and Dan Klein. 2021. FUDGE: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3511–3535, Online. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon. 2022. Mixture-of-experts with expert choice routing.

A Additional Experimental Details

A.1 Model Training and Computation

Training details. In our experiments, we finetune the base model and learn the latent variable parameters θ jointly: i.e., we optimize both the "base" and θ parameters by optimizing the marginal likelihood from Eq. (4) with AdamW (Loshchilov and Hutter, 2018). We train our models primarily using 4 A100 80G GPUs, and we use FlashAttention (Dao, 2023) and DeepSpeed ZeRO Stage 2 (Rasley et al., 2020) during training to reduce the GPU memory usage.

We follow Ivison et al. (2023) and Liu et al. (2024), using similar hyperparameters and settings detailed in Table 5. In all training experiments, we compute the marginal likelihood loss on only target tokens (i.e., we mask out the input tokens). It takes around 2 hours to finish the finetuning experiments for GSM8k, and we estimate a total of 3,000 GPU hours used for all experiments.

Datasets and prompts. Table 8 details the sizes of the training and evaluation datasets. We format the data using the same simple prompts during training and evaluation, with examples shown in Table 11.

Model and data licenses. We use three different LLMs in our experiments, i.e., LLAMA (Touvron et al., 2023), LLEMMA (Azerbayev et al., 2023), and MEDITRON (Chen et al., 2023), and they all share the same LLaMA 2 community license.[8] The licenses for the dataset are listed in Table 9.

Threshold search. For all methods with a learnable model selector (including Co-LLM), we choose the best η for Eq. (6) using a small validation set before testing, described in Algorithm 1. For GSM8k, MATH, and BioASQ, we conduct the search in-domain, using a small hold-out subset for that dataset; for AlpacaEval, we use a subset of a separate instruction following benchmark MT-Bench (Zheng et al., 2023) to search and pick the $\eta*$. We report the used values under different settings in Table 6 (for Co-LLM) and Table 6 (for Weakly-supervised Co-LLM).

A.2 BioASQ Evaluation

There are four different substasks in the BioASQ dataset, and they are evaluated using different met-

[8] https://ai.meta.com/llama/license/.

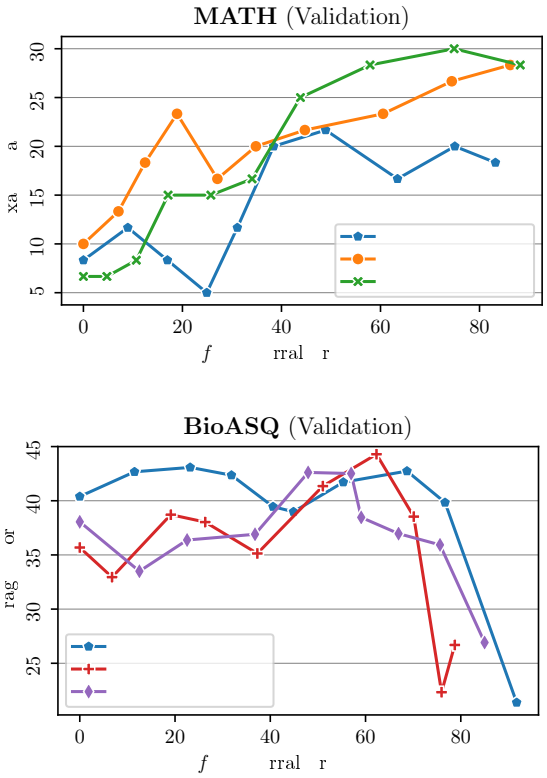

Figure 4: Performance of Co-LLM at different frequencies of deferral on GSM8k, MATH and BioASQ. There exists an optimal f that the joint model achieves better performance than using either of them alone.

Algorithm 1: Find Optimal Deferral Th. η

Input : Base Model, Asst. Model, Model Selector ϕ, Validation Dataset $\mathcal{D}$

Let $\mathcal{P} = \{\}$

for $i = 0$ **to** $|\mathcal{D}|$ **do**

  Given the input prompt in $\mathcal{D}_i$, generate response $X^{(i)}$ using the base model

  Predict per-token deferral probability $P_\phi(Z_t^{(i)} = 1 \mid X_{<t}^{(i)})$, and append it to $\mathcal{P}$

end

Sort $\mathcal{P}$ in ascending order

Set current best threshold $\eta = 0$ and evaluation score $s = 0$

for $j = 0$ **to** 100 **by** 10 **do**

  Get the j-th quantile p_j in $\mathcal{P}$ and use it as the deferral threshold η_j

  Generate responses $X^{(i)}$ for i in from 0 to $|\mathcal{D}|$ using the base and asst. model controlled by $P_\phi(Z_t^{(i)} = 1 \mid X_{<t}^{(i)}) > \eta_j$

  Evaluate the responses, and if the evaluation score is better than s, set $\eta = \eta_j$ and s to the new score

end

Return η

Hyperparameter	Configuration
Default and Co-LLM Finetuning for 7B Models[a]	
Training Epoch	2
Max Sequence Length	2048
Effective Batch Size	128
Gradient Accumulation Steps	16
Learning Rate	2e-5
Warmup Ratio	0.04
Weight Decay	0
AdamW β_1, β_2	0.9, 0.999
QLoRA Finetuning for 70B models[b]	
Training Epoch	2
LoRA Rank	64
LoRA Alpha	16
LoRA Dropout	0.1
Learning Rate	1e-4
Warmup Ratio	0.03
Weak Supervision Experiment	
λ for binary classifier loss	0.5
Positive Class Weight in binary loss	8 or 5[c]

[a] The settings are similar to the ones used by Liu et al. (2024).
[b] We adopt the same values as Ivison et al. (2023).
[c] 8 for Tülu-v2-mixture training and 5 for the rest, set based on the class imbalance in the training data.

Table 5: Training hyperparameters for our experiments.

rics according to the guideline:[9]

- **Factoid**: It require a particular entity name (e.g., of a disease, drug, or gene, or a number) to be generated per the question, which might not appera in the original question. The model may generate a list of candidate, and we pick the first generation (as the model often only generates one) and search for matching among the allowed candidate answers. This is the Strict Accuracy (SAcc.) metric in Krithara et al. (2023).
- **List**: Similar to Factoid questions but the model is expected to generate a list of entities. The model is required to produce the answer in a bullet list format, and we use F1 score to evaluate the performance.
- **Summary**: The answer is expected to be a long-form text like the description of a treatment. Following Krithara et al. (2023), we report ROUGE-2 (Lin, 2004) (using the implementation from the Huggingface Evaluate library[10]) to measure the textual overlapping between the generation and the ground truth.
- **Yes/No**: The model needs to provide a binary

[9] http://participants-area.bioasq.org/general_information/Task9b/
[10] https://huggingface.co/spaces/evaluate-metric/rouge

	Base	Asst.	η_*	f
AlpacaEval $N = 2048$	LLAMA-7B	LLAMA-70B	0.80	0.1
GSM8k $N = 512$	LLAMA-7B	LLAMA-70B	0.17	0.1
	LLAMA-7B	LLEMMA-7B	0.08	0.1
	LLAMA-7B	LLEMMA-34B	0.05	0.3
MATH $N = 512$	LLAMA-7B	LLAMA-70B	0.57	0.6
	LLAMA-7B	LLEMMA-7B	0.05	0.9
	LLAMA-7B	LLEMMA-34B	0.30	0.8
BioASQ $N = 512$	LLAMA-7B	LLAMA-70B	0.38	0.2
	LLAMA-7B	MEDITRON-7B	0.07	0.5
	LLAMA-7B	MEDITRON-70B	0.20	0.5

Table 6: For each dataset, we show the max number of generated tokens N (used for all models) and the optimal deferral threshold η_* (corresponding frequency f) used to generate the responses (available only for Co-LLM).

	Base	Asst.	η_*	f
AlpacaEval	LLAMA-7B	LLAMA-70B	0.11	0.1
GSM8k	LLAMA-7B	LLAMA-70B	1.00	0.0
MATH	LLAMA-7B	LLAMA-70B	0.44	0.1
BioASQ	LLAMA-7B	LLAMA-70B	1.00	0.0

Table 7: Similar to Table 6, we report the the optimal deferral threshold η_* and frequency used for Weakly-supervised Co-LLM.

	# of Samples		
Dataset	**Train**	**Dev**	**Test**
Tülu-v2-mixture	326,115		
MT-Bench		24	
AlpacaEval			805
GSM8k	7,473	50	200
MATH	7,498	60	750
BioASQ	4,719	20	310
		(Factiod)	93
		(List)	61
		(Summary)	75
		(Yes/No)	81

Table 8: The training and evaluation dataset sizes. For the instruction following task, we train a model on the Tülu mixture, uses a small validation set from the MT-Bench dataset to pick the deferral threshold η, and evaluate on the AlpacaEval dataset. For the other tasks, we train and test in-domain.

Dataset Name	License
AlpacaEval	CC BY-NC 4.0
GSM8k	MIT
MATH	MIT
BioASQ	CC BY 2.5
Tülu v2 mix	ODC BY

Table 9: The licenses of the datasets used.

answer yes or no to the given question, and the classification accuracy is reported.

B Additional Generation Examples

Fig. 2 shows an simplified version of generation for clarity, and we show the original generation in Table 10. An additional example generation on MATH is shown in Table 12.

`Opdu al ag contains two active components: 1) n iv ol umab and 2) rel at lim ab. The final answer is: 1) nivolumab 2) relatlimab`

Table 10: The original (token-level) model generation for the example in Fig. 2. Fig. 2 shows the generation at the word level rather than the token level for simplicity.

Dataset	Prompt
AlpacaEval	<|user|> What are some species of bears that are now extinct? <|assistant|>
GSM8k	Please solve the following math problem with detailed steps. Question: Tom's ship can travel at 10 miles per hour. He is sailing from 1 to 4 PM. He then travels back at a rate of 6 mph. How long does it take him to get back? Answer:
MATH	Given a mathematics problem, determine the answer. Simplify your answer as much as possible. You can use latex to format your answer and you should state your final answer as "The final answer is $(final-answer)$." Problem: Consider the function $g(x)=3x-4$. What is $g(0)$? Solution:
BioASQ	You are a medical expert answering real-world biomedical questions. Based on your understanding of basic and clinical science, medical knowledge, and mechanisms underlying health, disease, patient care, and modes of therapy, write a list of entity names, numbers, or short expressions that answer the following question. Question: Which genes are related to Emery-Dreifuss Muscular Dystrophy (EDMD)? Answer:

Table 11: Prompts used for formatting the datasets during training and evaluation. For training on the Tülu mixture, we adopt the same chat template as Wang et al. (2023). The prompt for GSM8k and MATH is based on the prompt by Wei et al. (2022) and Hendrycks et al. (2021). We modify the prompt for MedMCQA questions in the Meditron paper (Chen et al., 2023) for BioASQ, and adapt it for different tasks.

f	**Generation (Source TEX)**	**Rendered**
GT	`Completing the square in $x$ and $y$, we get \[9(x + 4)^2 + 4(y - 1)^2 = 324.\] Then \[\frac{(x + 4)^2}{36} + \frac{(y - 1)^2}{81} = 1.\]Thus, the center of the ellipse is $\boxed{(-4,1)}.$`	Completing the square in x and y, we get $$9(x+4)^2+4(y-1)^2=324.$$ Then $$\frac{(x+4)^2}{36}+\frac{(y-1)^2}{81}=1.$$ Thus, the center of the ellipse is $\boxed{(-4,1)}$.
0.0	`Completing the square in $x$ and $y$, we get \[9x^2 + 72x + 4y^2 - 8y - 176 = 0 \Rightarrow 9x^2 + 72x + 4y^2 - 8y - 176 = 9(x - 2)^2 + 72(x - 2) + 4(y - 2)^2 - 8(y - 2) - 176.\]Then \[\boxed{2} = \frac{1}{9}(9(x - 2)^2 + 72(x - 2) + 4(y - 2)^2 - 8(y - 2) - 176).\] The final answer is $2$.`	Completing the square in x and y, we get $$9x^2+72x+4y^2-8y-176=0\Rightarrow$$ $$9x^2+72x+4y^2-8y-176=$$ $$9(x-2)^2+72(x-2)+4(y-2)^2$$ $$-8(y-2)-176.$$ Then $$\boxed{2}=\frac{1}{9}(9(x-2)^2+72(x-2)$$ $$+4(y-2)^2-8(y-2)-176).$$ The final answer is 2.
0.4	`Completing the square in x and y, we get \[9x^2 + 72x + 4y^2 - 8y - 176 = 0 \Rightarrow 9(x - 4)^2 + 4(y - 1)^2 = 1.\]Then \[\frac{9(x - 4)^2}{1} + \frac{4(y - 1)^2}{1} = 1.\]Hence, \[a = \sqrt{9} = 3,\]and $b = \sqrt{4} = 2.$ Then the center of the ellipse is $\boxed{\textbf{(4, 1)}}.$ The final answer is $\textbf{(4, 1)}$.`	Completing the square in x and y, we get $$9x^2+72x+4y^2-8y-176=0\Rightarrow$$ $$9(x-4)^2+4(y-1)^2=1.$$ Then $$\frac{9(x-4)^2}{1}+\frac{4(y-1)^2}{1}=1.$$ Hence, $$a=\sqrt{9}=3,$$ and $b=\sqrt{4}=2$. Then the center of the ellipse is $\boxed{\textbf{(4, 1)}}$. The final answer is **(4, 1)**.
0.8	`Completing the square in $x$ and $y$, we get \[9(x^2 + 8x) + 4(y^2 - 2y) = 176.\]Then \[\begin{aligned} 9(x^2 + 8x + 16) + 4(y^2 - 2y + 1) &= 176 \\ 9(x + 4)^2 + 4(y - 1)^2 &= 176. \end{aligned}\]There fore, the center of the ellipse is $\boxed{(-4, 1)}.$ The final answer is $(-4, 1)$.`	Completing the square in x and y, we get $$9(x^2+8x)+4(y^2-2y)=176.$$ Then $$\begin{aligned} 9(x^2+8x+16)+4(y^2-2y+1)&=176\\ 9(x+4)^2+4(y-1)^2&=176. \end{aligned}$$ There fore, the center of the ellipse is $\boxed{(-4,1)}$. The final answer is $(-4,1)$.

Table 12: Model answers with different deferral frequencies to the question "Find the center of the ellipse whose equation is $9x^2 + 72x + 4y^2 - 8y - 176 = 0$." In this example, the base model is a LLAMA-7B model fine-tuned on the MATH dataset, and the reference model is the LLEMMA-34B model. We show the ground-truth answer in the first row, and the 0-shot model answers with different deferral ratios, with the original outputs and rendered. We use yellow background to indicate the token is produced by the reference model rather than the base model.