A1 CP in Multi-Step Setting

Algorithm 1 Multi-step LLM planning with human help.

for time $t \leftarrow 0$ to $T-1$ **do**
  Observe input x^t_{test}
  Predict a set $C(x^t_{\text{test}})$
  if $C(x^t_{\text{test}})$ is a singleton **then**
    execute corresponding action
  else
    ask for help, which is always assumed to provide the correct label, or the human provide clarification only
  end if
end for

Proof of Claim 1: Suppose $\overline{y} \in \overline{C}(\overline{x}_{\text{test}})$. We have,

$$\overline{y} \in \overline{C}(\overline{x}_{\text{test}}) \iff \min_t \hat{f}(x^t_{\text{test}})_{y^t} \geq 1-\hat{q} \tag{A1}$$

$$\iff \hat{f}(x^t_{\text{test}})_{y^t} \geq 1-\hat{q}, \qquad \forall t \in [T] \tag{A2}$$

$$\iff y^t \in C^t(x^t_{\text{test}}), \qquad \forall t \in [T] \tag{A3}$$

$$\iff \overline{y} \in C(\overline{x}_{\text{test}}). \tag{A4}$$

Proof of Proposition 2: Since we can bound the probability that $\overline{y}_{\text{test}} \notin \overline{C}(\overline{x}_{\text{test}})$, we can also bound the probability that $\overline{y}_{\text{test}} \notin C(\overline{x}_{\text{test}})$. From the conformalization procedure, we have the following *dataset-conditional* guarantee: with probability $1-\delta$ over the sampling of the calibration set $\overline{Z}$, we have

$$\mathbb{P}\big(\overline{y}_{\text{test}} \in \overline{C}(\overline{x}_{\text{test}}) | \overline{Z}\big) \geq \text{Beta}^{-1}_{N+1-v,v}(\delta), \quad v = \lfloor (N+1)\hat{\epsilon} \rfloor \tag{A5}$$

$$\xRightarrow{\text{Claim 1}} \mathbb{P}\big(\overline{y}_{\text{test}} \in C(\overline{x}_{\text{test}}) | \overline{Z}\big) \geq \text{Beta}^{-1}_{N+1-v,v}(\delta), \tag{A6}$$

where $\hat{\epsilon}$ is chosen such that $\epsilon = 1 - \text{Beta}^{-1}_{N+1-v,v}(\delta)$. Hence, the following *marginal* guarantee also holds:

$$\mathbb{P}\big(\overline{y}_{\text{test}} \in \overline{C}(\overline{x}_{\text{test}})\big) \geq 1-\hat{\epsilon}$$

$$\xRightarrow{\text{Claim 1}} \mathbb{P}\big(\overline{y}_{\text{test}} \in C(\overline{x}_{\text{test}})\big) \geq 1-\hat{\epsilon}.$$

This result provides a bound on the task completion rate if $\overline{x}_{\text{test}}$ is drawn using the distribution $\mathcal{D}$. However, recall that the sequence $\overline{x}$ of augmented contexts as defined in Section 3.3 arises from having performed the *correct* actions in previous steps; incorrect actions may result in a distribution shift. In order to obtain a bound on the task completion rate, we consider three cases at any given timestep: (1) the prediction set is a singleton and contains the correct label, (2) the prediction set is not a singleton but does contain the correct label, and (3) the prediction set does not contain the true label. The robot performs the correct action in the first two cases (without help in (1) and with help in (2)), while CP bounds the probability of case (3). Thus, the CP bound translates to a bound on the task success rate.

As seen in Eq. (3), we have from [10, Thm. 1], that we achieve the smallest average set size among all possible sequence-level prediction schemes, $\overline{C}$, if $\hat{f}$ models the prediction uncertainty accurately,

$$\min_{\overline{C} \in \overline{\mathcal{C}}} \mathop{\mathbb{E}}_{(\overline{x},\cdot) \sim \mathcal{D}} \big[|\overline{C}(\overline{x})|\big], \text{ subject to } \mathbb{P}\big(\overline{y} \in \overline{C}(\overline{x})\big) \geq 1-\hat{\epsilon}. \tag{A7}$$

A2 CP in Settings with Multiple Acceptable Options Per Step

Proposition 3 (Multi-label uncertainty alignment) *Consider a setting where we use CP with coverage level $1-\epsilon$ to construct the prediction set when there are multiple true labels and seek help whenever the set is not a singleton at each timestep. With probability $1-\delta$ over the sampling of the calibration set, the task completion rate over new test scenarios drawn from $\mathcal{D}$ is at least $1-\epsilon$.*

Proof: We have a dataset of $Z = \{(\tilde{x}_i, Y_i), \ldots\}_{i=1}^N$ sampled i.i.d. from a data distribution $\mathcal{D}$ for calibration (we use the same notation $\mathcal{D}$ as in the single-label setting here), where $Y_i := \{y_{i,j}\}_{j=1}^{J_i}$ is the set of true labels for a single trial. For each label, we use the same heuristic notion of confidence, $\hat{f}(x)_y \in [0,1]$.

We define an operator $\beta: \mathcal{X} \times \mathcal{Y}^J \to \mathcal{Y}$ where $\mathcal{X}$ is the space of contexts and $\mathcal{Y}$ is the space of labels:

$$\beta(x, Y) := \underset{y \in Y}{\operatorname{argmax}} \hat{f}(x)_y, \tag{A8}$$

which takes the true label with the highest confidence value from the true label set.

If we consider applying β to every point in the support of $\mathcal{D}$, a new distribution $\mathcal{D}'$ is induced. We also consider the induced dataset of samples $S' = \{(x_i, y_i^{\max})\}_{i=1}^N$, where $y_i^{\max} := \beta(x_i, Y_i)$. Then we can perform the usual conformalization and obtain the guarantee that with

$$C(x_{\text{test}}) := \{y | \hat{f}(x_{\text{test}})_y \geq 1 - \hat{q}\}, \tag{A9}$$

the following *marginal* guarantee holds,

$$\mathbb{P}(y_{\text{test}}^{\max} \notin C(x_{\text{test}})) \leq \hat{\epsilon}, \tag{A10}$$

$$\Rightarrow \mathbb{P}(\underset{y \in Y_{\text{test}}}{\operatorname{argmax}} \hat{f}(x_{\text{test}})_y \notin C(x_{\text{test}})) \leq \hat{\epsilon}, \tag{A11}$$

$$\Rightarrow \mathbb{P}(\beta(x_{\text{test}}, Y_{\text{test}}) \notin C(x_{\text{test}})) \leq \hat{\epsilon}, \tag{A12}$$

and the following *dataset-conditional* guarantee holds when we choose $\hat{\epsilon}$ such that $\epsilon = 1 - \text{Beta}^{-1}_{N+1-v,v}(\delta)$ where $v = \lfloor (N+1)\hat{\epsilon} \rfloor$,

$$\mathbb{P}(\beta(x_{\text{test}}, Y_{\text{test}}) \in C(x_{\text{test}}) | Z) \geq 1 - \epsilon. \tag{A13}$$

Hence, $C(x_{\text{test}})$ contains the true label with the highest confidence with probability at least $1 - \epsilon$.

At test time, we sample $(x_{\text{test}}, Y_{\text{test}})$ from $\mathcal{D}$ that is i.i.d. with samples in S — for the guarantee to hold for $\beta(x_{\text{test}}, Y_{\text{test}})$, we need to show $\beta(x_{\text{test}}, Y_{\text{test}})$ is a sample from $\mathcal{D}'$ that is i.i.d. with samples in S'. This is true since functions of independent random variables are independent, and functions of identically distributed random variables are identically distributed if the functions are measurable.

A3 CP in Multi-Step Setting with Multiple Acceptable Options Per Step

Proposition 4 (Multi-step, multi-label uncertainty alignment) *Consider a multi-step setting where we use CP with coverage level $1 - \epsilon$ to causally construct the prediction set when there may be multiple true labels at any step and seek help whenever the set is not a singleton at each timestep. With probability $1 - \delta$ over the sampling of the calibration set, the task completion rate over new test scenarios drawn from $\mathcal{D}$ is at least $1 - \epsilon$.*

Proof: For the multi-step setting, each trial now involves a sequence of contexts $\overline{x}$ and a set of sequences of true labels:

$$\overline{Y} = \{\overline{y}_1, \overline{y}_2, ..., \overline{y}_M\}, \tag{A14}$$

where $\overline{y}_m := (y_m^0, y_m^2, ..., y_m^{T-1})$. For example, $\overline{Y}$ can contain the sequence of "blue block, yellow block, green block", "green block, blue block, yellow block", ..., for the task of picking up three blocks. We collect a dataset of $\overline{Z} = \{(\overline{x}_i, \overline{Y}_i)\}$ of i.i.d. samples from the data distribution $\overline{\mathcal{D}}$.

Unlike the single-step setting, here we cannot apply β to the set of true labels in each step since we are reasoning over a *set of sequences*, and not a *sequence of sets* of true labels. Notably, the true label set at time step t depends upon the sequence of previously chosen true labels.

Let $Y^t[x^0, \bar{y}^{t-1}]$ denote the set of true labels at timestep t, *conditioned* upon the initial context x^0 and a partial sequence of past true labels $\bar{y}^{t-1} := (y^0, ..., y^{t-1})$ extracted from $\overline{Y}$. We then autoregressively define the following sequence:

$$\overline{\beta}_0(\overline{x}, \overline{Y}) := \underset{y \in Y^0}{\operatorname{argmax}} \hat{f}(x^0)_y, \quad Y^0 := \{y_1^0, ..., y_M^0\} \tag{A15}$$

$$\overline{\beta}_t(\overline{x}, \overline{Y}) := \overline{\beta}_{t-1}(\overline{x}, \overline{Y}) \bigcup \underset{y \in Y^t[x^0, \overline{\beta}_{t-1}(\overline{x}, \overline{Y})]}{\operatorname{argmax}} \hat{f}(x^t)_y, \quad t = 1, ..., T-1. \tag{A16}$$

For convenience, we denote $\overline{\beta}_t(\overline{x}, \overline{Y})[\tau]$ the τ element in $\overline{\beta}_t(\overline{x}, \overline{Y}), \tau \leq t$. An intuitive interpretation is that, we can consider $\overline{Y}$ forming a tree of valid executions (all possible actions that can be taken by choosing each of true labels). Hence, at each time step t, $\overline{\beta}_t(\overline{x}, \overline{Y})$ prunes the tree to a single branch by taking the true label with the highest heuristic value $\hat{f}(x^t)$. This reduces the tree of all possible sequences of true

labels to a single branch of true labels with highest confidence. Given this single branch of true labels, we can now perform CP as shown in the multi-step setting in Section A1.

We apply $\overline{\beta}_{T-1}$ to every point in the support of $\overline{\mathcal{D}}$, and a new distribution $\overline{\mathcal{D}}'$ is induced. We consider $\overline{S}' = \{(\overline{x}_i, \overline{y}_i^{\max})\}$, where $\overline{y}_i^{\max} := \overline{\beta}_{T-1}(\overline{x}_i, \overline{Y}_i)$. Let $\overline{Y}_{\text{test}}$ be the set of sequences of true labels for $\overline{x}_{\text{test}}$. Suppose we get the *marginal* bound with $\overline{\beta}_{T-1}$ as the labels:

$$\mathbb{P}(\overline{\beta}_{T-1}(\overline{x}_{\text{test}}, \overline{Y}_{\text{test}}) \notin C(\overline{x}_{\text{test}})) \leq \hat{\epsilon}, \tag{A17}$$

and *dataset-conditional* bound when we choose $\hat{\epsilon}$ such that $\epsilon = 1 - \text{Beta}^{-1}_{N+1-v,v}(\delta)$ where $v = \lfloor (N+1)\hat{\epsilon} \rfloor$,

$$\mathbb{P}(\overline{\beta}_{T-1}(\overline{x}_{\text{test}}, \overline{Y}_{\text{test}}) \notin C(\overline{x}_{\text{test}}) | \overline{Z}) \leq \epsilon, \tag{A18}$$

which states that at test time, given a context sequence $\overline{x}_{\text{test}}$, we produce a prediction set of sequences; if we consider a sequence consisting of the true label with the highest score at each step, the probability of this sequence covered by $C(\overline{x}_{\text{test}})$ is lower bounded by $1-\epsilon$. However, we need to be careful of following $\overline{\beta}_t$ at each step at test time. Conside the three cases:

- (1) At a given time-step, the prediction set $C^t(x^t_{\text{test}})$ does not contain the true label, $\overline{\beta}_t(\overline{x}, \overline{Y})[t]$.
- (2a) The prediction set is a singleton and does contain the true label.
- (2b) The prediction set is not a singleton (but does contain the correct label).

We already bound the probability of (1) happening with the CP bound; (2a) is fine since the LLM will take the correct action; (2b) is more challenging — in this case the robot asks the human for help, and we need to make sure the human "follows" the true label, by choosing the true label in the prediction set with the highest confidence by $\hat{f}$. In practice, we present the labels ranked by $\hat{f}$ and ask the human to choose the true label with the highest rank.

Now let's derive the bound in Eq. (A17) and Eq. (A18). Again we need to consider the causal construction issue. As seen in Section 3.3, we construct the prediction set $\overline{C}(\overline{x}_{\text{test}})$ non-causally using the score function $s_i = 1 - \hat{f}(\overline{x}_i)_{\overline{y}_i^{\max}}$ (taking minimum over steps). For a test sequence $\overline{x}_{\text{test}}$, we apply $\overline{\beta}_{T-1}$ to the true label set of sequences $\overline{Y}_{\text{test}}$ to get $\overline{y}^{\max}_{\text{test}} = \overline{\beta}_{T-1}(\overline{x}_{\text{test}}, \overline{Y}_{\text{test}})$. Now suppose $\overline{y}^{\max}_{\text{test}} \in \overline{C}(\overline{x}_{\text{test}})$, then we can show $\overline{y}^{\max}_{\text{test}} \in C(\overline{x}_{\text{test}})$ with the same proof as the single-label setting, which gives us the bound.

Lastly we need to show the sampled test sequence from $\overline{D}$ leads to a sample from $\overline{\mathcal{D}}'$ i.i.d. with $\overline{S}'$. This is true with the same argument that functions of independent random variables are independent.

A4 Additional Experiment Setting: Real Bimanual Setup

In this example, a real bimanual setup with two Kuka IIWA 7 arms move objects on the table, with one bin at each side (Fig. 5 right). The reachable workspace of each arm is limited so that one arm cannot reach the other end of the table or the other bin. Thus, there can be ambiguities in the choice of the arm depending on the task; e.g., Fig. 5 shows the human asking the robot to pass over the mango, but not specifying where the human is. KNOWNO is able to capture such ambiguities and triggers clarification. We design a scenario distribution with all instructions being ambiguous (thus requiring high human intervention rate): with $\epsilon = 0.15$, the robot achieves 84% plan success with 92% help. With 10 real trials, the robot succeeds 9 times while triggering help for 9 times.

A5 LLM Prompt Setup

Next we detail the LLM prompt setup for MCQA applied in KNOWNO. We will use the Mobile Manipulation setting from Section 4.3 as the example.

Multiple choice generation. Given a scenario, we first prompt the LLM to generate four options for possible actions. We apply few-shot prompting as shown in Fig. A1 below with zero temperature. In this scenario, there is a Coke, a bottled tea, and a Pepsi on the counter, and the task is to put the Coke in the top drawer but the choice of drawer is under-specified ("Put the Coke in the drawer please.").

```
We: You are a robot operating in an office kitchen. You are in front of a counter with two closed drawers,
a top one and a middle one. There is also a landfill bin, a recycling bin, and a compost bin.

We: On the counter, there is an orange soda, a Pepsi, and an apple.
We: Put that drink in the top drawer.
You:
A) open the top drawer and put the orange soda in it
B) open the middle drawer and put the Pepsi in it
C) open the middle drawer and put the orange soda in it
D) open the top drawer and put the Pepsi in it

We: On the counter, there is an energy bar, a banana, and a microwave.
We: Put the snack next to the microwave.
You:
A) pick up the energy bar and put it next to the microwave
B) pick up the banana and put it next to the energy bar
C) pick up the banana and put it next to the microwave
D) pick up the energy bar and put it next to the banana

We: On the counter, there is a Coke, a Sprite, and a sponge.
We: Can you dispose of the can? It should have expired.
You:
A) pick up the sponge and put it in the landfill bin
B) pick up the Coke and put it in the recycling bin
C) pick up the Sprite and put it in the recycling bin
D) pick up the Coke and put it in the landfill bin

We: On the counter, there is a bottled water, a bag of jalapeno chips, and a bag of rice chips.
We: I would like a bag of chips.
You:
A) pick up the bottled water
B) pick up the jalapeno chips
C) pick up the kettle chips
D) pick up the rice chips
(The correct option is either B or D, since either jalapeno chips or rice ships are fine.)

We: On the counter, there is a Coke, a bottled unsweetened tea, and a Pepsi.
We: Put the Coke in the drawer please.
You:
```

Figure A1: Prompt used for multiple choice generation in the Mobile Manipulation setting.

After the LLM generates four options, we append an additional option 'an option not listed here' to the four generated ones and then randomize the order to further prevent bias. We then use a zero-shot prompt in Fig. A2 for querying next-token probabilities ('A', 'B', 'C', D', 'E').

```
We: You are a robot operating in an office kitchen. You are in front of a counter with two closed drawers,
a top one and a middle one. There is also a landfill bin, a recycling bin, and a compost bin.

We: On the counter, there is a Coke, a bottled unsweetened tea, and a Pepsi.
We: Put the Coke in the drawer please.
You:
A) pick up the coke
B) pick up the coke and put it in the top drawer
C) pick up the coke and put it in the bottom drawer
D) a different option not listed here
E) pick up the pepsi
We: Which option is correct? Answer with a single letter.
You:
```

Figure A2: Prompt used for next-token prediction with generated multiple choices in the Mobile Manipulation setting.

A6 Additional Experiment Details

Environments. In addition to Fig. 1 and Fig. 5, here Fig. A3 shows the office kitchen environment with the set of drawers and bins used in the Mobile Manipulator experiments (left), and the bimanual setup with the set of objects used on the mat (right). There is another set of drawers used in the mobile manipulation experiments underneath a much bigger countertop not shown here.

Tasks and instructions. Next, we provide more details on the task settings, in particular, the possible ambiguities:

- Attribute ambiguities in Simulated setting: besides non-ambiguous terms like "green", "yellow", "blue", "block" and "bowl" ("put green block in yellow bowl"), refer to the block as one of "cube", "cuboid", "box", "square object", to the bowl as one of "container", "round object", "receptacle", or to either block or bowl as one of "object", "item", "thing" ("move the blue object in yellow bowl"); refer to "blue" as one of "cyan", "navy", to "green" as one of "greenish", "grass-colored", and to "yellow" as "orange" or "gold".
- Numeric ambiguities in Simulated setting: besides non-ambiguous terms like "a", "one", "a single of", "two", "a pair of", "three", "all" ("put a block in yellow bowl"), refer to either two or three numerically with one of "a few", "a couple of", "some", "a handful of" ("put some blocks in the green bowl").

Figure A3: (Left) Office kitchen environment with drawers and bins for the Mobile Manipulation setting. (Right) Bimanual setup with the set of objects used in the experiments.

- Spatial ambiguities in Simulated setting: besides non-ambiguous terms like "in front of", "behind", "to the left", and "to the right" ("put the green block to the left of green bowl"), refer to any of the four possible directions with "near", "close to", "beside", "next to", refer to either left to right with "lateral to", and refer to either front or behind with "along the line of sight".
- Real Tabletop Rearrangement setting: we split the 28 toy items (Fig. A4) into two categories of human liking them or disliking them: the things the human likes include corn, avocado, celery, carrot, tomato, lettuce, apple, orange, pear, lemon, peanut butter, sunny-side-up egg, egg, and pea; the human dislikes pretzel, cracker, waffle, mustard, ketchup, pizza, meat patty, cheese, chicken drumstick, peach, mango M&M, Skittles, and donut.
- Real Mobile Manipulator setting: please refer to `https://tinyurl.com/robot-help` for the full list.
- Bimanual setting: please refer to `https://tinyurl.com/robot-help` for the full list.

Next we provide more details on some of the baselines.

Baselines - Ensemble Set. Admittedly, our ensemble-based method is a weaker method than the traditional model-based ensemble where multiple copies of neural network are trained and inferred with; however, this is infeasible with the LLM we use. In our work, we randomize over the few-shot examples in the prompt as the ensemble. We select a pool of 20 possible MCQA examples (see examples in Fig. A1), and then randomly sample a certain amount from it for each inference. Note that in this case, Ensemble Set actually has advantage over KnowNo and Simple Set that, for the same data, it has seen many more examples than the fixed ones in the prompt used in KnowNo and Simple Set. We only apply ensemble for next-token prediction; the same set of multiple choices generated is used.

Baselines - Prompt Set. MCQA is also applied. Similar to Fig. A1, in the few-shot examples in the prompt, we show the true possible labels. For example, "We: Which options are possibly correct? You: A, C, D."

Baselines - Simple Set. Instead of MCQA, the LLM is first prompted to give the most likely action (e.g., "We: Put the Coke can in the drawer. You: I will" as the prompt). And then we attach the generated response to the same prompt, and ask LLM to label "Certain/Uncertain:", given few-shot examples.

A7 Additional Implementation Details

While the focus of KnowNo is mainly on providing uncertainty alignment for the LLM-based planner, below we provide details of the perception and action modules applied in all examples.

Perception. For all settings except for the Mobile Manipulator, we use either MDETR [52] (UR5 tabletop setting) or Owl-ViT [53] (Simulated and Bimanual settings) open-vocabulary object detector for recognizing the objects in the environment and obtaining the object locations for low-level action. In Simulated and Bimanual settings, the variations of the object types are limited, and with general prompting, the objects are detected without issue. In the UR5 tabletop setting, since we are use a wide variety of toy items (Fig. A4 right), the detector has issues often differentiating objects like peanut butter and meat patty that are both darker colors. We modify the scenario distributions to avoid using such items together in one scenario. In addition, we apply the Segment Anything model [54] to extract the object segmentation masks (shown overlaid in Fig. A4 left), and then use the `polylabel` algorithm [55] to find the most distant internal point of the mask as the suction point (shown as red dots).

Low-level action. In Simulated setting and UR5 tabletop setting, simple pick-and-place actions are executed based on object locations and solving the inverse kinematics. In Bimanual setting, the reachability of the Kuka arm is limited, and the pick-and-place action trajectories are solved using Sequential Quadtratic

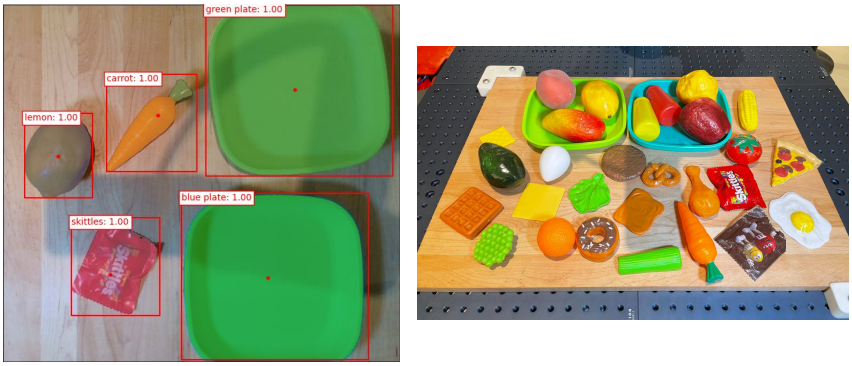

Figure A4: (Left) MDTER [52] object detection with Segment Anything [54] and most distant internal point (red dots) for UR5 tabletop setting. (Right) The total 28 toy items used for the experiments.

Programming (SQP) instead [56]. In the Mobile Manipulator setting, for most of the tasks that involve simple pick-and-place and opening the drawers, the action is from an end-to-end policy from the RT-1 policy (please refer to [57] for details), which takes in the raw observation. For some of the hard tasks such as putting the plastic bowl in the microwave and putting the metal bowl on the bowl, object locations are assumed known and we use scripted action policies.

A8 Additional Discussions

Sentence-level score leads to worse performance. In Section 2 we hypothesize that the distribution of probabilities (perplexity) of LLM outputs $p(y)$ is highly sensitive to the output length. Here we explore the effect of using sentence output and the perplexity score for CP. We still apply multiple choice generation first to obtain the possible options from LLM, and then query LLM scoring, for example, the probability of "put the blue block in the green bowl" with the prompt ending with "I will". Table A1 shows that for all three settings, using CP with perplexity leads to worse performance, and performance degradation correlates with variance of the multiple choice lengths.

Setting	Variance	Method	Set Size	Help
Attribute	1.52	KnowNo	1.18	0.18
		CP w/ Perplexity	1.33	0.32
Spatial	2.81	KnowNo	2.23	0.69
		CP w/ Perplexity	2.50	0.82
Numeric	8.51	KnowNo	2.17	0.79
		CP w/ Perplexity	4.06	1.00

Table A1: Comparison of KnowNo with CP with sentence output and perplexity score in the three settings in the Simulated setting. $\epsilon = 0.15$.

Potentially stronger baselines with model fine-tuning. In Section 4 we introduce the two prompt-based baselines Prompt Set and Binary, and demonstrate them being (1) inflexible (not allowing controlling the target success rate) and (2) do not properly model the uncertainty. We note that these two baselines can be potentially strengthened by fine-tuning the LLM to better predict the binary uncertainty or the uncertainty set, if the true labels can be properly defined. In fact, some recent work [37, 36] have explored model fine-tuning and exhibiting the effectiveness of Binary for uncertainty calibration. We also explored fine-tuning the GPT3 model (*davinci*) from OpenAI, which is the most powerful one from OpenAI available for fine-tuning. However, we find the model performing at very low accuracy with MCQA, and fine-tuning the model always results in overfitting to the dataset, even with thousands of data and varying hyperparameters (including ones from [37]). We suspect that our scenarios exhibit high complexity and variance, and it is non-trivial to fine-tune the model well with our dataset. Nonetheless, we do hope to have future work looking into better training the model for proper uncertainty, and then applying CP on top of it to achieve set-based calibration.