

## APPENDIX

**Table of Contents**


---

<b>A</b>	<b>SPRINT Pseudocode</b>	<b>16</b>
<b>B</b>	<b>Large Language Model Prompt</b>	<b>16</b>
<b>C</b>	<b>Baselines and Implementation</b>	<b>17</b>
C.1	Language-conditioned Behavior CLoning . . . . .	19
C.2	Actionable Models (AM) . . . . .	19
C.3	SPRINT . . . . .	19
<b>D</b>	<b>Dataset and Environment Details</b>	<b>20</b>
D.1	Dataset Details . . . . .	20
D.2	Evaluation Tasks . . . . .	21
<b>E</b>	<b>Extended Experiments, Results, and Analysis</b>	<b>25</b>
E.1	LLM Summary Examples . . . . .	25
E.2	Comparing Summaries from Different LLMs . . . . .	27
E.3	Step by Step Zero-shot evaluations . . . . .	27
E.4	Extended Ablation Study . . . . .	28
E.5	Qualitative Comparison Results . . . . .	28

---

## A SPRINT PSEUDOCODE

---

### Algorithm 1 SPRINT Algorithm

---

**Require:** Dataset  $\mathcal{D}^L$  w/ language instruction labels, LLM

- 1: **AGGREGATESKILLS**( $\mathcal{D}^L$ , LLM) ▷ Automated LLM skill relabeling (Sec. 4.2)
- 2: **while** not converged **do**
- 3:    $\tau_z \leftarrow \mathcal{D}^L$ : Sample an annotated skill (sub-)trajectory
- 4:   Train offline RL on  $\tau_z$
- 5:
- 6:    $\tau_z^{\text{chain}} \leftarrow \text{CROSSCHAINSKILLS}(\mathcal{D}^L)$  ▷ Cross-trajectory skill chaining (Sec. 4.3)
- 7:   Train offline RL on  $\tau_z^{\text{chain}}$
- 8:
- 9:    $\tau_{\text{agg}_1}, \tau_{\text{agg}_2} \leftarrow \text{CROSSAGGREGATESKILLS}(\mathcal{D}^L, \text{LLM})$  ▷ Cross-traj. aggregation (Sec. 4.3)
- 10:   Train offline RL on  $\tau_{\text{agg}_1}, \tau_{\text{agg}_2}$
- 11: **end while**
- 12:
- 13: **procedure** **AGGREGATESKILLS**( $\mathcal{D}^L$ , LLM)
- 14:   **for** composite trajectory  $\tau_z$  in  $\mathcal{D}^L$  **do**
- 15:     Separate  $\tau_z$  into language annotated sub-trajectories  $[\tau_{z_1}, \dots, \tau_{z_N}]$
- 16:     **for** all adjacent sub-trajectories  $[\tau_{z_i} \dots \tau_{z_j}]$  **do**
- 17:       Assign name from LLM:  $\text{LLM}(z_i \dots z_j) = \hat{z}_{i:j}$
- 18:        $\tau_{\hat{z}_{i:j}} \leftarrow \text{Concat}[\tau_{z_i}, \dots, \tau_{z_j}]$  and relabel with  $\hat{z}_{i:j}$
- 19:        $T \leftarrow \text{length of } \tau_{\hat{z}_{i:j}}$
- 20:        $R(s_T, a_T, \hat{z}_{i:j}) = 1$  ▷ Label last reward with 1 (Eq. 2).
- 21:        $\mathcal{D}^L = \mathcal{D}^L \cup \{\tau_{\hat{z}_{i:j}}\}$
- 22:     **end for**
- 23:   **end for**
- 24: **end procedure**
- 25:
- 26: **procedure** **CROSSCHAINSKILLS**( $\mathcal{D}^L$ )
- 27:   Sample random (sub)trajectories  $\tau_{z_1}, \tau_{z_2} \sim \mathcal{D}^L$
- 28:   Sample random endpoint  $j$  in  $\tau_{z_1}$
- 29:    $\tau_{z_2}^{\text{chain}} \leftarrow [(s_0, a_0, 0, z_2), \dots, (s_j, a_j, Q^\pi(s_j, a_j, z_2), z_2)]$  ▷ Relabel reward w/ Eq. 4.
- 30:   **return**  $\tau_{z_2}^{\text{chain}}$
- 31: **end procedure**
- 32:
- 33: **procedure** **CROSSAGGREGATESKILLS**( $\mathcal{D}^L$ , LLM)
- 34:   Sample random (sub)trajectories  $\tau_{z_1}, \tau_{z_2} \sim \mathcal{D}^L$  with lengths  $T_1, T_2$
- 35:   Assign new name:  $\hat{z} = \{z_1\} \cdot \{z_2\}$
- 36:    $\tau_{\text{agg}_1} \leftarrow [(s_0, a_0, 0, \hat{z}), \dots, (s_{T_1}, a_{T_1}, Q^\pi(s_{T_1}, a_{T_1}, \hat{z}), \hat{z})]$  ▷ Relabel reward w/ Eq. 5
- 37:    $\tau_{\text{agg}_2} \leftarrow [(s_0, a_0, 0, \hat{z}), \dots, (s_{T_2}, a_{T_2}, 1, \hat{z})]$  ▷ Relabel reward w/ Eq. 5.
- 38:   **return**  $\tau_{\text{agg}_1}, \tau_{\text{agg}_2}$
- 39: **end procedure**

---

## B LARGE LANGUAGE MODEL PROMPT

We list the full large language model summarization prompt in Figure 8. The examples in the prompt are fixed for all summarization queries. These examples are selected from the ALFRED validation dataset (which is not otherwise used in our work) at random: We spell out the primitive skill annotations in the “Task Steps:” part of each prompt example. Then, the “Summary” for each of these is the high-level, human-written annotation for that trajectory from ALFRED. We repeatedly sampled these trajectories until each example mentioned a different object to prevent biasing the LLM towards certain types of objects.

We note that the “Look at the vase under the light” example is important to make the LLM give reasonable summaries for similar tasks in ALFRED where the agent picks something up and turns on a light. This is because most of the human labels for turning on the lamp do not mention the object in the previous step, making it difficult for the LLM to realize that the task has to do with looking at the held object under a lamp.

<p>Instructions: summarize the following ordered steps describing common household tasks.</p> <p>Task Steps: 1: Pick up the smaller knife on the counter to the left of the stove. 2: Slice the tomato with the smaller knife. 3: Put the knife in the sink. 4: Pick up a slice of tomato from the countertop. 5: Heat up the slice of tomato in the microwave, removing it afterwards. Summary: Microwave the tomato slice after slicing it with the smaller knife on the counter.</p> <p>Task Steps: 1: Pick up the vase. 2: Turn on the lamp. Summary: Look at the vase under the light.</p> <p>Task Steps: 1: Grab the pencil off of the desk. 2: Put the pencil in the bowl. 3: Grab the container off of the desk. 4: Put the container down at the back of the desk. Summary: Put a bowl with a pencil in it on the desk.</p> <p>Task Steps: 1: Pick up the bar of soap from the back of the toilet. 2: Put the bar of soap in to the sink, turn on the faucet to rinse off the soap, pick up the soap out of the sink. 3: Put the soap in the cabinet under the sink and on the left. Summary: Put a rinsed bar of soap in the cabinet under the sink.</p> <p>Task Steps: 1: [SKILL 1]. 2: [SKILL 2]. 3: [SKILL 3]. ... N: [SKILL N]. Summary:</p>
--

Figure 8: The full prompt that we use for summarization. Following the suggestions of Ahn et al. (2022) for prompt design, we explicitly number each step. The LLM completion task begins after “Summary:”. For brevity, we omit the new line characters between all numbered steps.

## C BASELINES AND IMPLEMENTATION

We implement IQL (Kostrikov et al., 2022) as the base offline RL algorithm for all goal-conditioned offline RL pretraining baselines and ablations due to its strong offline and finetuning performance on a variety of dense and sparse reward environments. At a high level, IQL trains on in-distribution  $(s, a, s', r, a')$  tuples from the dataset rather than sampling a policy for  $a'$  to ensure that the Q and value functions represent accurate estimated returns constrained to actions in the dataset. The value function is trained with an expectile regression loss controlled by a hyperparameter  $\tau$ , where  $\tau = 0.5$  results in standard mean squared error loss and  $\tau \rightarrow 1$  approximates the max operator, resulting in a more optimistic value function that can better “stitch” together trajectories to obtain distant reward in sparse reward settings. The IQL policy is trained to maximize the following objective:

$$e^{\beta(Q(s,a)-V(s))} \log \pi(a|s),$$

which performs advantage-weighted regression (Peng et al., 2019) with an inverse temperature term  $\beta$ . In practice, the exponential advantage term is limited to a maximum value to avoid numerical overflow issues. We detail shared training and implementation details below, with method-specific information and hyperparameters in the following subsections.

**Observation space.** The state space of the ALFRED environment consists of  $300 \times 300$  RGB images. Following the baseline method in ALFRED (Shridhar et al., 2020), we preprocess these images by sending them through a frozen ResNet-18 encoder (He et al., 2016) pretrained on ImageNet (Deng et al., 2009). This results in a  $512 \times 7 \times 7$  feature map that we use as the observation input to all networks. Furthermore, as ALFRED is a partially observable, egocentric navigation environment, we concatenate the last 5 frames as the full observation, resulting in an observation that is of the shape  $(512 * 5) \times 7 \times 7$ .

**Action space.** The agent chooses from 12 discrete low-level actions. There are 5 navigation actions: MoveAhead, RotateRight, RotateLeft, LookUp, and LookDown and 7 interaction actions: Pickup, Put, Open, Close, ToggleOn, ToggleOff, and Slice. For interaction actions the agent additionally selects one of 82 object types to interact with, as defined by Pashevich et al. (2021). In total, the action space consists of  $5 + 7 * 82 = 579$  discrete action choices. Note that this action space definition is different from the action space in Shridhar et al. (2020), which

used a pixel-wise mask output to determine the object to interact with. In contrast to Shridhar et al. (2020) we aim to train agents with *reinforcement learning* instead of imitation learning and found the discrete action parametrization more amenable to RL training than dense mask outputs. For all methods, due to the large discrete action space, we perform some basic action masking to prevent agents from taking actions that are not possible. For example, we do not allow the agent to `Close` objects that aren’t closeable nor can they `ToggleOn` objects that can’t be turned on.

**Language embedding.** We embed language instructions with the `all-mpnet-base-v2` pretrained sentence embedding language model from the `SentenceTransformers` python package (Reimers & Gurevych, 2019). This produces a 768-dimensional language embedding which is used as input for language-conditioned policy and critic functions, as detailed below.

**Policy and critic networks.** We train a discrete policies with two output heads of size 12 and 82 for the action and interaction object outputs respectively. Critic networks are conditioned on both the observation and the discrete action output of the policy. In all policy and critic networks, we process the ResNet feature observation inputs with 4 convolutional layers. In networks with language input, we flatten the output of the convolutional layers and concatenate the observation features with the 768-dim language embedding, before passing the concatenated image-language features through a series of fully connected layers. Additionally, we use FiLM (Perez et al., 2018) to condition convolutional layers on the language embeddings.

**Pre-training hyperparameters.** A hyperparameter search was performed first on the language-conditioned BC-baseline to optimize for training accuracy. These hyperparameters were carried over to the IQL implementation, and another search for IQL-specific hyperparameters were performed on a baseline IQL policy conditioned on semantic instructions. With these parameters fixed, we performed one more hyperparameter search specific to Actionable Models but for the final implementation of SPRINT we re-used the same hyperparameters and only selected SPRINT-specific parameters heuristically. Hyperparameters for each method are detailed in a separate table. Shared hyperparameters for all methods (where applicable) are listed below:

Param	Value
Batch Size	1024
Learning Rate	2e-3
Optimizer	AdamW
Dropout Rate	0.2
Weight Decay	0.05
Discount $\gamma$	0.98
Q Update Polyak Averaging Coefficient	0.005
Q-Network Discrete Action Embedding Size	48
Q-Network Discrete Object Selection Action Embedding Size	24
Policy and Q Update Period	Once every training iteration
Batch Norm	True
Nonlinearity	ReLU
IQL Advantage Clipping	[0, 100]
IQL Advantage Inverse Temperature $\beta$	10
IQL Quantile $\tau$	0.9

**Finetuning details and hyperparameters.** We perform finetuning experiments for Language-conditioned BC, Actionable Models, and SPRINT. For all models, we finetune the model on only newly collected task data by running online IQL (without any of the chaining or aggregation steps). Each method is finetuned on every task in the *EVAL\_UNSEEN* task set individually; that is, we pre-train once and then finetune ten times, once for every task in the task set. We then average returns over all tasks, then report metrics averaged over all random seeds. For each task, we define a maximum rollout time horizon of 2 timesteps per environment action required by the expert planner.

When not specified, finetuning parameters are identical to pre-training parameters. Finetuning hyperparameters are specified below:

Param	Value
Dropout Rate	0
# Initial Rollouts	50
Training to Env Step Ratio	0.5
$\epsilon$ in $\epsilon$ -greedy action sampling	0.25: annealed down to 0.05
# Parallel Rollout Samplers	4

### C.1 LANGUAGE-CONDITIONED BEHAVIOR CLONING

Our language-conditioned behavior cloning (L-BC) comparison method is inspired by and replicates BC-Zero (Jang et al., 2021) and LangLp (Lynch & Sermanet, 2021). BC-Zero performs FiLM-conditioned semantic imitation learning (Perez et al., 2018) and both BC-Zero and LangLp have an additional image/video-language alignment objective. In BC-Zero, their video alignment objective aligns language embeddings with videos of humans performing tasks related to those the BC-Zero robot agent trains on. LangLp’s image-language alignment objective allows their policy to accept both image and natural language goals as input due to only having a subset of their data labeled with hindsight language labels. As we don’t have human videos of these tasks and our entire dataset is labeled with language labels, we do not add a video or image alignment objective.

Therefore, we implement L-BC by using the same architecture as described above with just a single policy network that trains to maximize the log-likelihood of actions in the dataset. As our entire dataset consists of expert trajectories, this baseline ideally learns optimal actions for the instructions.

Hyperparameters for the L-BC baseline are identical to the shared parameters above, where applicable.

### C.2 ACTIONABLE MODELS (AM)

Actionable Models (Chebotar et al., 2021) pretrains a goal-conditioned Q function conditioned on randomly sampled image goals and also performs a goal-chaining procedure very similar to our semantic skill chaining procedure. We implement AM by modifying the base IQL policy and critic networks to take in image goals instead of natural language embeddings as goals. These goals are provided in the same way as the observations, i.e., as a concatenated stack of 5 frames (the last 5 frames in the trajectory) processed by a frozen ResNet-18. Therefore, goals are the same shape as observations:  $(512 * 5) \times 7 \times 7$ .

To allow for fair comparison between our approach and AM, we implement AM with the same powerful offline RL algorithm, IQL (Kostrikov et al., 2022), used in our method. IQL ensures that the policy does not choose out-of-distribution actions by using advantage-weighted regression on in-distribution actions for policy extraction. With this, we found the conservative auxiliary loss AM adds to push down Q-values for out-of-distribution actions to be unnecessary and even hurtful to its overall performance, so we omit this additional loss term.

Finally, after consulting the authors of AM, we tried varying maximum trajectory lengths when sampling random goals. We found that allowing random goals to be sampled from anywhere within a trajectory resulted in the best zero-shot evaluation performance for AM, so our numbers are reported with this implementation detail.

### C.3 SPRINT

The implementation details of SPRINT follow from the discussion about implementing IQL at the top of this section. The key differences are in (1) language model skill aggregation and (2) cross-trajectory skill chaining, detailed below.

**LLM Skill Aggregation.** We perform LLM skill aggregation fully offline by iterating through every ALFRED trajectory and aggregating sequences of adjacent primitive skill sub-trajectories. Assuming a trajectory with  $N$  primitive skills, we select all  $\binom{N}{2}$  pairs of start and end skills and aggregate all instructions from start to end with the LLM. With 73k original language-annotated

trajectories, this procedure allows us to generate an additional 110k aggregated trajectories. We then add these trajectories to the original dataset and train on the entire set.

**Cross-trajectory skill chaining.** We perform cross-trajectory skill chaining in-batch. Instead of sampling a second trajectory to perform chaining on, we simply permute the batch indices to generate a set of randomly sampled second trajectories. Then, we perform a second loss function update, in addition to the original update on the sampled trajectories, with equal loss weighting, to apply the skill-chaining update.

SPRINT-specific hyperparameters follow:

Param	Value
Large Language Model for Relabeling	OPT-13B (Zhang et al., 2022)
LLM Token Filtering Top-p <sup>1</sup>	0.9
LLM Token Sampling Temperature	0.8

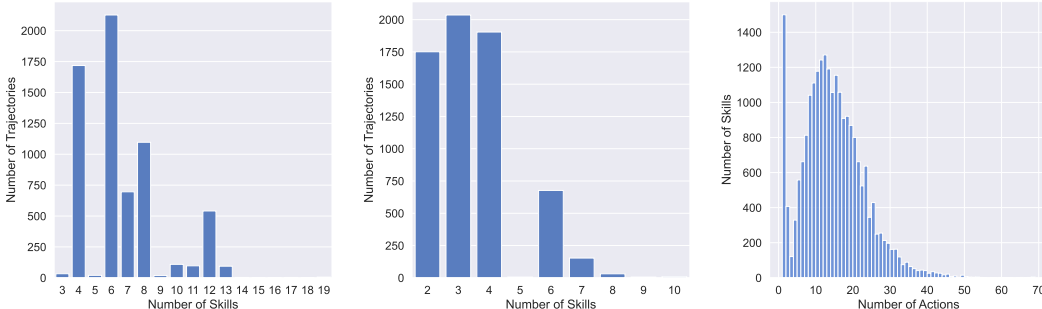
<sup>1</sup>At each token generation step, only the highest probability tokens with total probability mass that add up to the top-p are considered.

## D DATASET AND ENVIRONMENT DETAILS

### D.1 DATASET DETAILS

For training and evaluation we leverage the ALFRED benchmark and dataset (Shridhar et al., 2020). The ALFRED training dataset contains  $\sim 6.6$ k trajectories collected by an optimal planner following a set of 7 high-level tasks with randomly sampled objects (*e.g.*, pick up an object and heat it). Each trajectory has at least three crowd-sourced sets of language instruction annotations. Each trajectory consists of a sequence of 3-19 individually annotated skills (see Figure 9, left). This results in a total of 141k language-annotated skill trajectories.

However, nearly half of the language instructions in the ALFRED dataset are navigation skill instructions like “turn left, then look up and walk to the counter on the right”. To get a more balanced skill annotation dataset, we merge all navigation skills with the skill that immediately follows them, using only the annotation of the next skill. After this processing step, the resulting dataset contains 73k language-annotated primitive skill trajectories. After we merge the navigation skills, the average number of skills in each trajectory is 3.5 skills per trajectory (Figure 9, middle), and the average number of actions in each skill is 14.3 (Figure 9, right).



(a) Skills per trajectory in the original ALFRED dataset. (b) Skills per trajectory in the merged dataset. (c) Actions per skill in the merged dataset.

Figure 9: **Left:** distribution of the number of skills in each trajectory in the original ALFRED dataset. **Middle:** distribution of skills per trajectory in the “merged” dataset with merged navigation skills. **Right:** distribution of number of actions per skill in the “merged” dataset.

## D.2 EVALUATION TASKS

### Data Collection Overview

Thank you for participating in this short summarization task. You will be writing one-sentence summaries of ordered instructions describing household tasks. Please try to ensure that the one-sentence summary you write either implicitly or explicitly describes the entire set of instructions.

For example, given the following sentences:

1. Open the fridge and take a pot of water out of the fridge, then close the fridge door.
2. Boil the pot of water on the stove.

You could write "Boil a pot of water from the fridge."

A few more examples:

1. Cut the lettuce to the left of the sink.
  2. Put the knife down on the stove.
  3. Take a slice of lettuce.
  - Summary: Cut the lettuce on the left of the sink and take a slice.
1. Pick up the glass of water to the right of the sponge, on the left of the shelf.
  2. Pour the glass of water into the plant's soil.
  3. Fill the glass with water from the sink.
  4. Pour the glass of water into the plant.
  5. Put down the glass of water.
  - Summary: Water the plant with two glasses of water.

You will be writing at most 40 summaries for instruction sets ranging from 2 to up to 5 instructions long.

During this process, if you feel like a given set of instructions can't be readily turned into a one-sentence high-level summary, feel free to get a new set of sentences by pressing the "Skip" button. Once you understand these instructions, feel free to continue to the next cell and begin!

### Annotation Task 1 (40 summaries)

```
In [1]: from sam_train_valid_data_collection_utils import interact_program
train_annotated_skills = []
scene_type="train"
interact_program(train_annotated_skills, scene_type)
```

Skills to summarize: Please write a one-sentence task description that describes the following instructions:

1. Pick up the bottle on the toilet basin.
2. Place the bottle behind the bar soap on the counter.

Write your summary here:

Skip

Submit Summary

Figure 10: Data collection jupyter notebook page.

We evaluate agents through zero-shot policy evaluation and finetuning on three sets of evaluation tasks in the ALFRED environment: (1) *EVAL\_100* to measure the ability of pre-trained agents to execute semantically meaningful instructions at varied levels of abstraction, (2) *EVAL\_CHAIN* to measure the ability of agents to chain behaviors across multiple trajectories to solve long tasks, and (3) *EVAL\_UNSEEN* to evaluate generalization performance when finetuning to unseen household floor plans.

**Collecting evaluation task data.** The ALFRED dataset provides high-level language annotations for each of the trajectories in the dataset. We could use these annotations as unseen task-instructions to evaluate our agents. However, we found that the different skills is not equally distributed across trajectories of different skill lengths, *e.g.*, most 2-skill trajectories perform pick-and-place tasks while tasks involving heating skills only appear in length 7+ trajectories. To allow evaluation with a less biased skill distribution, we create the *EVAL\_100* task set by randomly choosing a trajectory from the ALFRED dataset *and then randomly sampling a subsequence of skills* from this trajectory. To obtain a high-level language instruction that summarizes this new subsequence, we crowd-source labels from human annotators. For labeling, each annotator is presented with a remotely hosted Jupyter notebook interface (see Figure 10). Whenever we by chance sample a full ALFRED trajectory for annotation, we directly used the existing high-level annotation from the ALFRED dataset.

We annotate 80 trajectories with human annotators and combine them with 20 randomly sampled single-skill trajectories, resulting in a total of 100 evaluation tasks (see Figure 11 for example instructions).

For *EVAL\_CHAIN*, we randomly sampled 20 full trajectories from the ALFRED dataset that had sequences of 7 or 8 skills (10 of length 7, 10 of length 8) and removed these trajectories from the post-LLM aggregated training dataset. We did not remove any of the LLM-aggregated trajectories made up of subsequences of skills within that trajectory. This allows AM and SPRINT to perform skill chaining to solve these tasks by ensuring that there were valid sequences of skills to chain together to be able to solve these removed tasks. For example, assume a (shortened for clarity) sampled skill sequence is “pick up apple,” then “put apple in microwave”, then “slice the apple.” Then, either Actionable Models or SPRINT can chain together the sub-trajectory associated with “pick up apple” then “put apple in microwave” with the “slice the apple” sub-trajectory to solve this task. These trajectories all had annotations from ALFRED annotators, so we used those annotations directly (see Figure 13 for example instructions).

Finally, for *EVAL\_UNSEEN*, we collected a set of 10 full-length trajectories from the ALFRED “valid-unseen” dataset consisting of validation tasks in unseen floor plans. We collected 2 of each length from 1 through 5 for a total of 10 tasks by sampling random full-length trajectories from this dataset, with the exception of length 1 tasks (we just sample random skills to create length 1 tasks). As these are full trajectories, they already have human annotations from ALFRED, which we directly use as the task description (see Figure 12 for example instructions).

We list additional details about the tasks in each evaluation set in Table 2.

Table 2: Evaluation Task Specifics. Note that the “number of env actions per task” corresponds to the number of environment actions the ALFRED expert planner required to complete that task.

	<i>EVAL_100</i>	<i>EVAL_CHAIN</i>	<i>EVAL_Unseen</i>
Number of Tasks	100	20	10
Task Lengths (# primitive skills)	[1, 2, 3, 4, 5, 6, 7]	[7, 8]	[1, 2, 3, 4, 5]
Min Number of Env Actions per Task	1	34	2
Avg Number of Env Actions per Task	39.1	60.9	46.6
Max Number of Env Actions per Task	113	104	124

Finally, we display 5 randomly sampled tasks, along with their human annotations, from each of our task sets in Figures 11, 12, and 13.

**Online finetuning environment setup.** During online-finetuning we initialize the agent in the same house floor plan as the trajectory the task was extracted from to ensure executability. During finetuning, we give each episode a time horizon of  $2 \times$  the number of environment actions needed by the expert planner to solve the task. We give sparse rewards for each skill solved by the agent during the episode. Therefore for length 1 tasks, the agent can only be rewarded once before the episode ends, while for length 5 tasks, the episode terminates on the fifth reward signal. We give a reward of  $\frac{1}{\text{num total skills}}$  for each skill the agent successfully executes so that the return sums to 1. We found that this helped to finetune all comparison methods more stably, possibly due to the fact that giving larger rewards (e.g., 1 for each skill) results in out-of-distribution critic values (when compared to pre-training) that de-stabilize online reinforcement learning.



<p>Skills to Summarize: 1: Grab the knife on the counter. 2: Place the knife in the sink then turn the faucet on so water fills the sink. Turn the faucet off and pick up the knife again. 3: Place the knife on the table to the left of the wooden bowl.</p> <p>Annotator Summary: Wash the knife from the counter, put in on the table.</p>
<p>Skills to Summarize: 1: Pick up the blue book closest to your and the phone from the bed. 2: Turn on the lamp to take a look at the book in the light.</p> <p>Annotator Summary: Examine the book by the light of a lamp.</p>
<p>Skills to Summarize: 1: Pick up yellow candle on counter. 2: Open cabinet, put candle in cabinet, close cabinet 3: Pick up yellow candle from toilet.</p> <p>Annotator Summary: Move the candle from the sink to the cabinet under the sink, close it and then pick the candle from the top of the toilet in front of you.</p>
<p>Skills to Summarize: 1: Pick the pot on the left side up from the stove. 2: Set the bowl and knife on the table next to the tomato.</p> <p>Annotator Summary: Put the bowl with the knife in it next to the tomato.</p>
<p>Skills to Summarize: 1: Pick up the pen that's in front of you that's under the mug. 2: Put the pencil in the mug that was above it. 3: Pick up the mug with the pencil in it.</p> <p>Annotator Summary: Put the pen into the mug and pick up the mug.</p>

Figure 11: Randomly sampled, human language instruction annotations from the *EVAL\_100* task set.

<p>Skills to Summarize: 1: Pick up the lettuce on the counter. 2: Chill the lettuce in the fridge. 3: Put the chilled lettuce on the counter, in front of the bread.</p> <p>Annotator Summary: Put chilled lettuce on the counter.</p>
<p>Skills to Summarize: 1: Pick up an egg from off of the kitchen counter. 2: Open the fridge, put the egg in to chill for a few seconds and then take it back out. 3: Place the cold egg in the sink.</p> <p>Annotator Summary: Chill an egg and put it in the sink.</p>
<p>Skills to Summarize: 1: Pick up the butter knife off of the right side of the kitchen island. 2: Put the knife handle down in the frying pan that is on the front left burner of the stove. 3: Pick up the frying pan with the knife in it off of the stove. 4: Put the frying pan with the knife in it into the sink basin to the right of the potato.</p> <p>Annotator Summary: Put a frying pan with a knife in it into the sink.</p>
<p>Skills to Summarize: 1: Take the pencil from the desk. 2: Put the pencil on the desk.</p> <p>Annotator Summary: Take the pencil from the desk, put it on the other side of the desk.</p>
<p>Skills to Summarize: 1: Pick up the left pillow on the chair. 2: Put the pillow on the sofa right of the newspaper. 3: Pick up the pillow on the chair. 4: Put the pillow on the sofa left of the newspaper.</p> <p>Annotator Summary: Place two pillows on a sofa.</p>

Figure 12: Randomly sampled, human language instruction annotations from the *EVAL\_UNSEEN* task set.

<p>Skills to Summarize: 1: Pick up the knife in front of the lettuce. 2: Slice the apple in the sink with the knife. 3: Place the knife into the sink. 4: Pick up the sliced apple from the sink. 5: Place the apple slice into the pot on the stove. 6: Pick up the pot from the stove. 7: Pick up the pot from the stove.</p> <p>Annotator Summary: Slice an apple for the pot on the stove and put the pot on the counter to the right of the door.</p>
<p>Skills to Summarize: 1: Take the apple from the counter in front of you. 2: Place the apple in the sink in front of you. 3: Take the knife by the sink in front of you. 4: Cut the apple in the sink in front of you. 5: Place the knife in the sink in front of you. 6: Take an apple slice from the sink in front of you. 7: Heat the apple in the microwave, take it out and close the microwave. 8: Place the apple slice in the sink in front of you.</p> <p>Annotator Summary: Place a warm apple slice in the sink.</p>
<p>Skills to Summarize: 1: Pick up the loaf of bread. 2: Put the bread on the counter above the spatula. 3: Pick up the knife that's above and to the right of the loaf of bread. 4: Cut the top half of the loaf of bread into slices. 5: Put the knife on the edge of the counter in front of you horizontally. 6: Pick up a slice of bread from the middle of the loaf. 7: Cook the bread in the microwave then take it out and close the microwave door. 8: Throw the cooked slice of bread away.</p> <p>Annotator Summary: Put a microwaved slice of bread in the oven.</p>
<p>Skills to Summarize: 1: Pick the knife up from off of the table. 2: Open the microwave, slice the potato, and close the microwave. 3: Open the microwave, place the knife inside of it, and close the microwave. 4: Open the microwave, pick up the potato slice inside, close the microwave. 5: Place the potato slice in the pan on the stove. 6: Pick up the pan from the stove. 7: Open the refrigerator, place the pan inside, and close the refrigerator.</p> <p>Annotator Summary: Move the pan from the stove top to inside the black refrigerator.</p>
<p>Skills to Summarize: 1: Pick up the red tomato on the counter to the right of the stove. 2: Put the tomato onto the island below the butter knife. 3: Pick up the butter knife off of the kitchen island. 4: Slice up the tomato on the kitchen island. 5: Place the butter knife onto the island to the right of the sliced tomato. 6: Pick up a tomato slice off of the kitchen island. 7: Open the fridge and put the tomato slice on the bottom shelf, then close the door, after a couple seconds open the fridge and remove the tomato slice then close the door. 8: Open the microwave door and place the tomato slice inside the microwave in front of the egg.</p> <p>Annotator Summary: Put a chilled tomato slice into the microwave.</p>

Figure 13: Randomly sampled, human language instruction annotations from the *EVAL\_CHAIN* task set.

## E EXTENDED EXPERIMENTS, RESULTS, AND ANALYSIS

Table 3: *EVAL\_100* and *EVAL\_Chain* eval dataset per-length and overall skill completion rates. See Section 5 for experiment setup.

		AM	L-BC	SPRINT
<i>EVAL_100</i>	Number of Completed Subtasks Overall	$0.46 \pm 0.05$	$0.41 \pm 0.02$	<b><math>1.27 \pm 0.13</math></b>
	Length 1 Progress	$0.23 \pm 0.01$	<b><math>0.77 \pm 0.04</math></b>	$0.53 \pm 0.07$
	Length 2 Progress	$0.22 \pm 0.03$	$0.24 \pm 0.03$	<b><math>0.49 \pm 0.03</math></b>
	Length 3 Progress	$0.17 \pm 0.03$	$0.16 \pm 0.04$	<b><math>0.48 \pm 0.03</math></b>
	Length 4 Progress	$0.10 \pm 0.01$	$0.06 \pm 0.01$	<b><math>0.35 \pm 0.06</math></b>
	Length 5 Progress	$0.16 \pm 0.08$	$0.02 \pm 0.01$	<b><math>0.46 \pm 0.06</math></b>
	Length 6 Progress	$0.16 \pm 0.01$	$0.02 \pm 0.03$	<b><math>0.36 \pm 0.12</math></b>
	Length 7 Progress	$0.00 \pm 0.00$	$0.00 \pm 0.01$	<b><math>0.01 \pm 0.02</math></b>
<i>EVAL_CHAIN</i>	Number of Completed Subtasks Overall	$0.67 \pm 0.09$	$0.04 \pm 0.05$	<b><math>2.59 \pm 0.66</math></b>
	Length 7 Progress	$0.04 \pm 0.01$	$0.00 \pm 0.00$	<b><math>0.25 \pm 0.10</math></b>
	Length 8 Progress	$0.13 \pm 0.02$	$0.01 \pm 0.01$	<b><math>0.43 \pm 0.08</math></b>

Here, we present additional results complementary to the experiments in the main paper in Section 5. We present and analyze LLM annotation examples in Section E.1, zero-shot evaluations with step-by-step task instructions in Section E.3, and an extended ablation analysis in Section E.4.

### E.1 LLM SUMMARY EXAMPLES

We randomly sample 12 OPT-13B task summaries produced while performing skill aggregation (explained in Section 4.2) using the prompt in Figure 8 and display them in Figure 14. Because we sample the tokens sequentially during text generation, some of the summaries mostly describe one or a few of the skills but in a non-sensical manner (e.g., LLM(“Pick up the plastic bowl on the left. Turn on the lamp.”) = “Look at the plastic bowl on the left”). The summaries also likely can be improved by including more examples in the prompt. However, after analyzing many more examples, we see that OPT-13B generally provides fitting high-level summaries for most sequences by skipping over implied sub-tasks (although it sometimes also skips over important sub-tasks, likely due to the prompt). For example, it summarizes the task “Pick up the plaid pillow that is on the left end of the couch. Place the pillow on the ottoman” as “Place a plaid pillow on the ottoman,” in which picking up the plaid pillow is implied. It also often demonstrates object relation understanding, e.g., by stating “Place a plaid pillow” when only the first sub-task in the sequence mentions that the pillow is plaid.

Skills to Summarize: 1: Pick up the plaid pillow that is on the left end of the couch. 2: Place the pillow on the ottoman.

LLM-generated Summary: Place a plaid pillow on the ottoman.

Skills to Summarize: 1: Pick up the piece of potato furthest to the right. 2: Open the microwave, place the potato inside, close the door, microwave for two seconds, then remove the potato. 3: Place the cooked potato piece to the right of the salt shaker.

LLM-generated Summary: Place a cooked potato piece on the table.

Skills to Summarize: 1: Put the spoon in the pot on the left. 2: Pick up the pot with the spoon.

LLM-generated Summary: Put the pot with the spoon on the left.

Skills to Summarize: 1: Open the microwave above the stove and slice the apple inside with the knife. 2: Place the knife standing up on the blue dish. 3: Remove a slice of apple from the microwave. 4: Place the apple slice on the blue plate.

LLM-generated Summary: Put an apple slice on the blue plate.

Skills to Summarize: 1: Pick up the butter knife that is next to the fork. 2: Open the refrigerator. 3: Open the refrigerator and place the knife on the bottom shelf in the refrigerator then close the door.

LLM-generated Summary: Open the refrigerator and place the butter knife on the bottom shelf in the refrigerator then close the door.

Skills to Summarize: 1: Open the fridge and put the knife inside. 2: Take a piece of the apple out of the microwave.

LLM-generated Summary: Take a slice of apple out of the microwave.

Skills to Summarize: 1: Pick up the remote that is on the shelf. 2: Put the remote on the couch.

LLM-generated Summary: Put the remote on the couch.

Skills to Summarize: 1: Pick up the dirty green towel to the right of the bar of soap on the vanity. 2: Put the towel into the sink, rinse it for a few seconds and pick it back up. 3: Open the far left bottom drawer to the left of the sink, put the rinsed green towel inside and close the drawer.

LLM-generated Summary: Put the towel in the drawer.

Skills to Summarize: 1: Pick up the plastic bowl on the left. 2: Turn on the lamp.

LLM-generated Summary: Look at the plastic bowl on the left.

Skills to Summarize: 1: Slice up the fruit that is in the sink. 2: Place the knife in the second row of the fridge. 3: Grab a slice of the fruit. 4: Place the fruit slice on the back right pan. 5: Grab the pan that has the fruit in it.

LLM-generated Summary: Put a slice of fruit on a pan in the fridge

Skills to Summarize: 1: Pick up the pot on the front right burner. 2: Chill the pot in the fridge then get it back out and close the door.

LLM-generated Summary: Put a pot on the stove, put it in the fridge, get it out, close the door.

Skills to Summarize: 1: Pick up the knife on the counter. 2: Cut the potato in the sink into slices. 3: Put the knife in the sink with the potato. 4: Pick up a slice of potato from in the sink. 5: Warm the potato slice in the microwave and then remove it.

LLM-generated Summary: Warm the potato slice in the microwave.

Figure 14: LLM Summary examples. We randomly sample 12 language combinations generated by the OPT-13B text generation model (Zhang et al., 2022) and display them without cherry-picking specific examples. The LLM often creates a high level description by skipping implied steps (like humans do), *e.g.*, “Put the remote on the couch” implies the need to pick up the remote.

## E.2 COMPARING SUMMARIES FROM DIFFERENT LLMs

In this section, we justify the use of OPT-13b as our summarization model by comparing its summaries of three different skill sequences to OPT-350m (350 million parameters) and OPT-1.3b (1.3 billion parameters). The summaries and sequences are displayed in Figure 15. Although these are 3 randomly sampled examples, they highlight differences between the models that hold true across many examples that we examined.

As the language model size increases, it understands more of the relationships between sub-tasks in each sequence, therefore generating better summaries. An example of this is the second example in Figure 15, which highlights a sequence in which both OPT-1.3b and OPT-13b understand the high-level task of putting a warmed plate in the cabinet, as opposed to OPT-350m which essentially repeats the second sub-task in the sequence. Yet sometimes, only OPT-13b produces a reasonable summary. In the last example, OPT-13b produces a summary that implies that an agent must wash the potato (“Put a *rinsed* potato on the counter...”); the other two models’ summaries only mention the last step of putting the potato down.

<p>Skills to summarize: 1. Pick the clock up from the desk. 2. Turn the lamp on.  <b>OPT-350m Summary:</b> Look at the clock.  <b>OPT-1.3b Summary:</b> Turn on the lamp.  <b>OPT-13b Summary:</b> Look at the clock.</p>
<p>Skills to summarize: 1. Take the plate from the cabinet to the left of the stove. 2. Warm the plate in the microwave, above the stove. 3. Put the plate in the cabinet.  <b>OPT-350m Summary:</b> Place the plate in the microwave.  <b>OPT-1.3b Summary:</b> Put a warmed plate in the cabinet.  <b>OPT-13b Summary:</b> Put a warmed plate in the cabinet.</p>
<p>Skills to summarize: 1. Open the microwave, take out the potato and shut the door. 2. Put the potato in the sink, turn the water on and off again. Take the potato out of the sink. 3. Place the potato on the counter, to the right of the sink.  <b>OPT-350m Summary:</b> Put the potato on the counter, to the left of the sink.  <b>OPT-1.3b Summary:</b> Place the potato on the counter, to the right of the sink.  <b>OPT-13b Summary:</b> Put a rinsed potato on the counter to the right of the sink.</p>

Figure 15: Comparison of OPT-350m, 1.3b, and 13b summaries on 3 randomly sampled sequences from our dataset. In general, as the model size increases, the summary becomes better. However, there are some sequences all three on which all 3 models do not generate good summaries for, such as the first sequence in these examples.

## E.3 STEP BY STEP ZERO-SHOT EVALUATIONS

Humans may sometimes prefer to give more detailed instructions, *e.g.*, if they do not trust a household robot to successfully execute a high-level instruction. In that case, they are likely to give detailed, step-by-step instructions instead. In this section, we demonstrate the effectiveness of our pre-training strategy in enabling execution of longer-horizon tasks when specified by such step-by-step instructions. To generate these step-by-step instructions, we combine all subtask instructions into one paragraph to condition agents on instead of the high-level human annotation. For example, a task with 2 subtasks, “Pick up the knife” and “Slice the potato,” is given the simple task annotation “Pick up the knife. Slice the potato.”

We compare SPRINT and IL on both *EVAL\_100* and *EVAL\_CHAIN* tasks with the combined, step-by-step task annotations in Table 4. L-BC and SPRINT both perform similarly on the step-by-step *EVAL\_100* tasks compared to with the original annotations, while both perform better on

*EVAL\_CHAIN* tasks than with the original annotations. This is likely due to how *EVAL\_CHAIN* tasks are on average much longer (7 or 8 tasks). Therefore, instructions spelling out exactly what to do allow agents to more easily understand what and in which order subtasks should be performed.

Table 4: Step-by-step evaluation number of completed sub-tasks (number of completed sub-tasks).

	L-BC	SPRINT
<i>EVAL_100</i> Completed Subtasks	$0.46 \pm 0.01$	<b><math>1.24 \pm 0.19</math></b>
<i>EVAL_CHAIN</i> Completed Subtasks	$0.22 \pm 0.03$	<b><math>3.11 \pm 0.08</math></b>

#### E.4 EXTENDED ABLATION STUDY

In this section, we examine a series of additional questions regarding specific design choices of our method, in a manner complementary to Section 5.1.3. We also visualize all ablations’ zero-shot policy evaluation performance in on both *EVAL\_100* and *EVAL\_CHAIN* task sets in Table 5.

**How much does the LLM contribute to skill aggregation?** To answer this question, we compare using SPRINT with LLM aggregation on adjacent sub-trajectory sequences but no chaining (**SPRINT w/o chain**) to SPRINT with skill aggregation, but where the skills are relabeled by naïvely concatenating the sentences together (**SPRINT w/o chain, w/ concat-agg**). Across both task sets, SPRINT w/o chain, w/ concat-agg is outperformed by SPRINT w/o chain, especially in *EVAL\_CHAIN*. This signifies that using the LLM helps with understanding very-long horizon, high-level semantic instructions as the LLM generates relevant task summaries for consecutive skills in a trajectory.

**What if we relabel with the LLM during chaining?** We examine also using the LLM to label skills during cross-trajectory skill aggregation (**SPRINT w/ LLM-chain**), rather than concatenating the skill annotations together (**SPRINT**). Overall, SPRINT w/ LLM-chain performs slightly worse in average return and success rates. When analyzing the summaries generated by the LLM, we found that randomly paired instructions can rarely be summarized meaningfully, thereby resulting in noisy and sometimes meaningless instructions. Therefore, we implemented SPRINT w/ LLM-chain by only utilizing the top 1% (top 10 candidates with our batch size of 1024) of in-batch chaining candidates (ranked by the LLM’s prediction of what the next skill should be). Even so, the performance is not better than just simply concatenating the instructions, as even these top 1% candidates still have a high chance of not being sensible sentences to summarize.

Table 5: *EVAL\_100* Ablation Returns and Success Rates

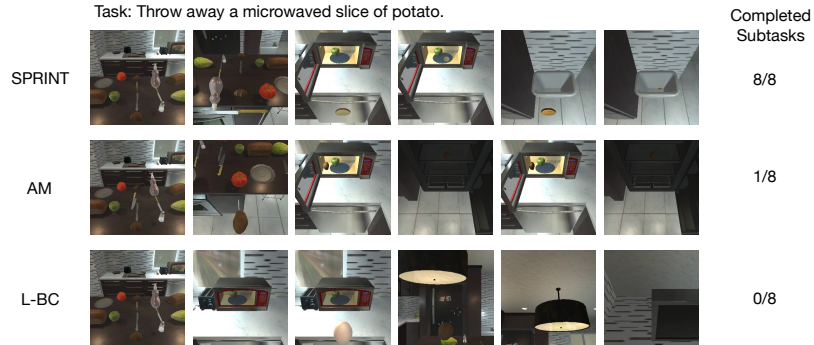
	<i>EVAL_100</i> Average Return	<i>EVAL_CHAIN</i> Average Return
SPRINT (ours)	<b><math>1.27 \pm 0.13</math></b>	<b><math>2.59 \pm 0.66</math></b>
SPRINT w/ LLM-chain	$1.15 \pm 0.01$	$2.46 \pm 0.21$
SPRINT w/o chain	$0.91 \pm 0.03$	$2.04 \pm 0.04$
SPRINT w/o chain, w/ concat-agg	$0.77 \pm 0.06$	$0.67 \pm 0.20$
SPRINT w/o LLM-agg	$0.38 \pm 0.05$	$0.10 \pm 0.04$

#### E.5 QUALITATIVE COMPARISON RESULTS

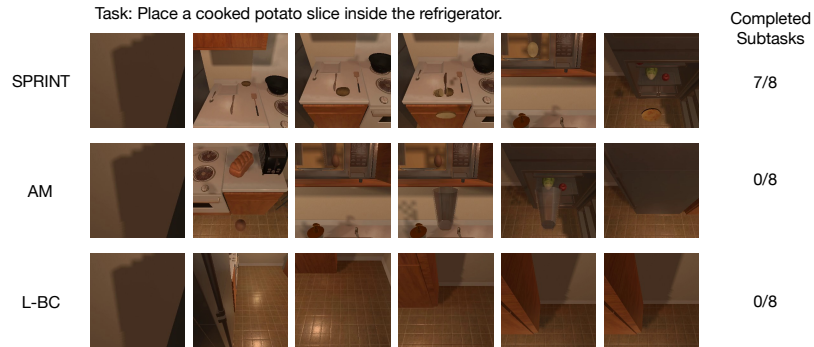
**Zero-shot evaluation.** We compare SPRINT, AM, and L-BC zero-shot evaluation results on long *EVAL\_CHAIN* tasks in Figure 16. In general, SPRINT is able to make substantially more progress on *EVAL\_CHAIN* tasks as it leverages the large language model to generate longer-horizon, semantically meaningful pre-training tasks and performs cross-trajectory chaining to learn to chain its existing dataset tasks. In the visualized examples, SPRINT is able to understand and successfully execute many of the sub-tasks implied but not directly stated by the natural language task instruction. L-BC makes very little progress on these tasks, not even understanding what the first sub-task to complete should be as the task annotation is out of distribution from what it saw while training.

Finally, AM is able to make some progress on some of these tasks due to its long-horizon goal pre-training objective. However, this is less effective than our language-conditioned pre-training in such zero-shot evaluations.

**Finetuning.** We finetune SPRINT, AM, and L-BC on *EVAL\_UNSEEN* tasks, in household floor-plans that were never seen while training, and visualize qualitative policy rollout examples *after finetuning* in Figure 17. In general, SPRINT is able to finetune to longer-horizon tasks while AM and L-BC both struggle with making progress on longer-horizon tasks despite receiving rewards for every completed sub-task. SPRINT’s ability to complete more sub-tasks on many of the longer-horizon tasks is demonstrated in Figure 17a, while a case in which both SPRINT and AM make partial progress throughout finetuning is demonstrated in Figure 17b. We believe that AM has more trouble finetuning on these tasks than SPRINT because the task specification for AM (goal images) is out of distribution; pre-training on *semantic* tasks with SPRINT allows agents to more easily learn longer-horizon behaviors as the task specifications may still be in-distribution of the pre-training tasks that LLM skill-aggregation and skill chaining produce.



(a) SPRINT successfully solves this task, while AM fails to slice the potato and repetitively iterates between putting the potato in the fridge and microwave. L-BC fails to even pick up the potato, as the task annotation does not directly describe picking up a potato.



(b) SPRINT nearly solves this task, while AM picks up an egg instead of a potato. L-BC picks up random objects not related to the annotation.



(c) SPRINT completes the entire task. AM picks up the tomato but fails to put it down onto the counter and slice it. L-BC aimlessly wanders and picks up random objects.

Figure 16: Visualizations of zero-shot policy rollouts on three tasks in the *EVAL\_CHAIN* task set.





(a) SPRINT picks up and places one of the pillows on the sofa, and picks up the second but does not manage to place the second on the sofa, thus completing 3/4 subtasks. AM and L-BC both learn to pick up a pillow but never learned to place it in the correct spot.



(b) SPRINT and AM both learn to pick up a pencil from the desk, although neither manage to put the pencil down in the correct place “on the other side of the desk.” Meanwhile, L-BC never picks up the pencil.

Figure 17: Visualizations of policy rollouts on two tasks in the *EVAL\_UNSEEN* task set, after fine-tuning each method. These floor plans were originally unseen to all agents until finetuning.