Notes: Language Models Meet World Models: Embodied Experiences Enhance Language Models

Kaiwen Bian

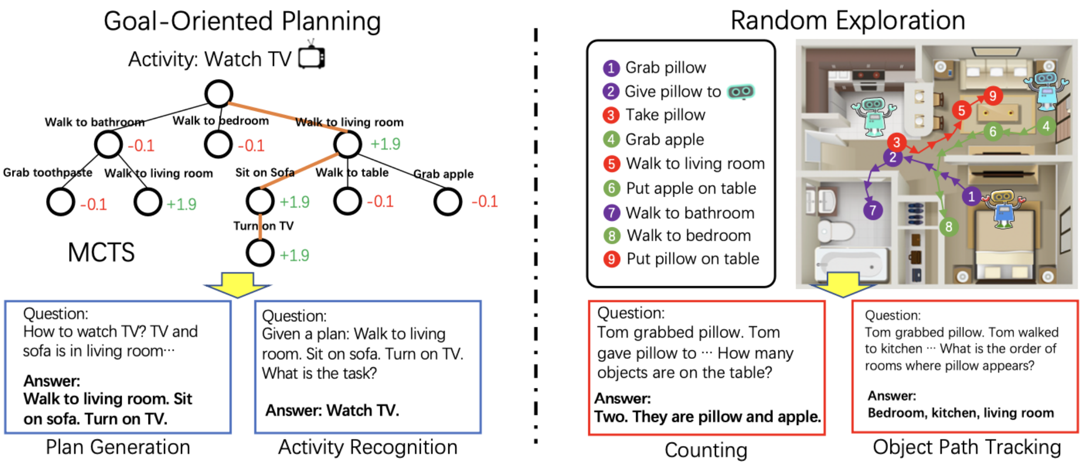
October 16, 2024

Aims and Setup

Aim to inject diverse fundamental embodied knowledge and skills into **pretrained LMs**, while retaining the models' generality. We introduce a novel training paradigm for LMs – **fine- tuning with Embodied Experiences from World Models (E2WM)**. Here, world models are embodied simulators that emulate physical interactions in real-world environments (e.g., VirtualHome).

Aim to consider a diverse range of fundamental knowledge and skills for embodied tasks, including **tracking objects, planning to complete given goals, recognizing other agents' behaviors**, etc. To this end, the paper introduce two ways to collect embodied experiences from world models that give rise to the desired knowledge and skills:

1. **Goal-oriented planning**: aims to gather experiences associated with planning and goal-oriented agent behaviors.
2. **Random exploration**: focuses on accumulating experiences that involve object and world state tracking.



Crucially, to finetune LMs on the collected embodied experiences while retaining their original general knowledge and capabilities, the classical **Elastic Weight Consolidation (EWC)** is introduced into the training paradigm. By regularizing the finetuning loss, EWC aims to preserve the important LM parameters from pretraining. EWC is substantially **more effective than the popular KL regularization**.

Also uses **efficient low-rank updates** by harmonizing the recent Low-Rank Adaptation (LoRA) with the EWC regularizer. This results in the new EWC-LoRA update rule that greatly reduces training costs and makes our E2WM paradigm accessible to cheap hardware (GPUs).

Approach

A world model simulator (e.g., VirtualHome) is used to generate embodied experiences for fine-tuning language models (LMs). These experiences provide information about object permanence, goal planning, and tracking, which are challenging to derive from text alone.

Monte Carlo Tree Search (MCTS)

In goal-oriented planning, the Monte Carlo Tree Search (MCTS) algorithm is used to explore the possible actions an agent can take to achieve a predefined goal. The reward structure is defined as follows:

- At each time step, if a goal predicate is fulfilled, the agent receives a reward of $+2$.
- To avoid repeating the same actions, achieved predicates are removed from the goal set.
- A penalty of -0.1 is given for irrelevant actions to discourage unnecessary steps.

Elastic Weight Consolidation (EWC)

To preserve the general knowledge of the pretrained LM while fine-tuning on embodied tasks, the authors use Elastic Weight Consolidation (EWC) for regularization. The loss function with EWC is:

$$L(\theta) = L_V(\theta) + \lambda \sum_i F_{i,i}(\theta_i - \theta_i^*)^2$$

where:

- $L_V(\theta)$ is the loss for the fine-tuning task,
- $F_{i,i}$ is the Fisher information for parameter θ_i,
- θ_i^* is the pretrained model parameter,
- λ is a regularization coefficient.

The Fisher information matrix F is computed as:

$$F_{i,i} = \frac{1}{N} \sum_{j=1}^{N} \left(\frac{\partial L_U^{(j)}}{\partial \theta_i^*} \right)^2$$

where L_U is the loss for the original pretraining task and N is the number of data samples.

Low-Rank Adaptation (LoRA)

To reduce the memory and computational cost of fine-tuning, the authors combine EWC with Low-Rank Adaptation (LoRA). The weight matrix W is decomposed as:

$$W = W^* + BA$$

where W^* is the frozen pretrained weight matrix, and $B \in \mathbb{R}^{r \times k}, A \in \mathbb{R}^{k \times d}$ are two low-rank matrices, with $k \ll \min(r, d)$.

The loss function with EWC-LoRA becomes:

$$L(\theta) = L_V(\theta) + \lambda \sum_i F_{i,i} h_i^2$$

where $h_i = \theta_i - \theta_i^*$ is the difference between the adapted and original weights, now represented in low-rank form through the matrices A and B.

Task-Specific Learning

The collected experiences are used to fine-tune the language model on tasks such as plan generation, activity recognition, counting, and object tracking. The overall loss for multi-task learning is defined as:

$$L_V = \sum_{v \in V} \alpha_v \sum_{m=1}^{M} \log P(y_m | y_{<m}, x)$$

where:

- V is the set of tasks,
- α_v is the weight for task v,
- x is the input (e.g., the initial condition in plan generation),
- $y = \{y_1, \ldots, y_M\}$ is the stepwise action sequence or the output label.