Language-Guided World Models
A Model-Based Approach to AI Control

*Alex Zhang[◇], *Khanh Nguyen[♠], Jens Tuyls[◇], Albert Lin[♣], Karthik Narasimhan[◇]
[◇] Princeton University [♠]University of California, Berkeley
[♣]University of Southern California

Project website: language-guided-world-model.github.io

Abstract

This paper introduces the concept of *Language-Guided World Models* (LWMs)—probabilistic models that can simulate environments by reading texts. Agents equipped with these models provide humans with more extensive and efficient control, allowing them to simultaneously alter agent behaviors in *multiple* tasks via natural verbal communication. In this work, we take initial steps in developing robust LWMs that can generalize to compositionally novel language descriptions. We design a challenging world modeling benchmark based on the game of MESSENGER (Hanjie et al., 2021), featuring evaluation settings that require varying degrees of compositional generalization. Our experiments reveal the lack of generalizability of the state-of-the-art Transformer model, as it offers marginal improvements in simulation quality over a no-text baseline. We devise a more robust model by fusing the Transformer with the EMMA attention mechanism (Hanjie et al., 2021). Our model substantially outperforms the Transformer and approaches the performance of a model with an oracle semantic parsing and grounding capability. To demonstrate the practicality of this model in improving AI safety and transparency, we simulate a scenario in which the model enables an agent to present plans to a human before execution, and to revise plans based on their language feedback.

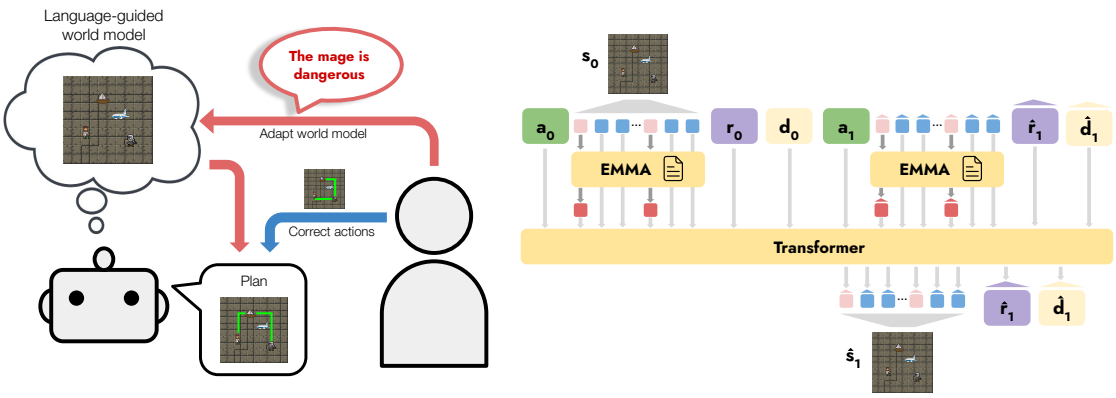
1 Introduction

Model-based agents are artificial agents equipped with probabilistic "world models" that are capable of foreseeing the future state of an environment (Deisenroth and Rasmussen, 2011; Schmidhuber, 2015). World models endow these agents with the ability to plan and learn in imagination (i.e., internal simulation) and have led to exciting results in the field of reinforcement learning (Finn and Levine, 2017; Ha and Schmidhuber, 2018; Chua et al., 2018; Hafner et al., 2023). These models have been studied extensively for the purpose of improving the autonomous performance of artificial agents.

In this paper, we endorse and enhance the model-based approach for a different goal: to strengthen the controllability of artificial agents. Since all policies of a model-based agent are optimized with respect to a common world model, a human can adjust multiple policies simultaneously by making appropriate changes to this model. This mechanism complements the model-free approach that updates policies individually, offering greater efficiency and flexibility in control. For example, by incorporating the fact that the floor is slippery into the world model of a robot, a person can effectively remind it to handle *every* object in a room with greater caution. If the performance of the robot on a task remains unsatisfactory, the person can continue to fine-tune its policy for that specific task. In contrast, without a world model, they have to separately adapt the robot's policies to the slippery-floor condition.

The model-based approach requires world models that can be easily modulated by humans. Traditional world models fall short in this quality because they can only be modified using observational data, which is not a suitable medium for humans to convey intentions (Sumers et al., 2023; Zheng et al., 2023). To overcome the limitations of these models, we develop ***L**anguage-Guided **W**orld **M**odels* (LWMs)—world models that can be effectively steered through human verbal communication. Agents equipped with LWMs inherit all the benefits of model-based agents while being able to incorporate language-based supervision. This capability reduces human teaching effort and mitigates the risk of agents taking harmful actions in an environment to explore its dynamics. LWM-based agents can also self-improve by reading "free" texts

*First two authors contribute equally. Correspondence email: kxnguyen@berkeley.edu.



(a) (b)

Figure 1: Language-guided world models (LWMs) offer human an efficient mechanism to regulate artificial agents. (a) We illustrate a potential application of LWMs to improving AI safety and transparency. These models enable an agent to generate visual plans and invite a human supervisor to validate them. Moreover, the human can adjust the plans by modifying the agent's world model with language feedback, in addition to directly correcting its policy. (b) We design an architecture for LWMs that exhibits strong compositional generalization. We replace the cross-attention mechanism of the standard Transformer with a new attention mechanism inspired by Hanjie et al. (2021) to effectively incorporate language descriptions. We then train a model that auto-regressively generates tokenized observations conditioned on language descriptions and actions.

composed to guide humans (e.g., game manuals), reducing the subsequent effort to fine-tune them through direct interaction.

Building LWMs poses a unique research challenge: grounding language to environmental dynamics. This problem is difficult because the language used to describe environment dynamics can be incredibly rich and complex, encompassing a wide range of concepts such as entity names, appearances, motions, interactions, spatial and temporal relations, and more. Moreover, in natural settings, especially when describing artificial environments (e.g., games), new concepts are often introduced but may not always be clearly defined. Humans deal effectively with this issue because they possess remarkable reasoning capabilities that allow them to infer word meanings from observations. For example, a caption like "*the Ziff, which is chasing the player, is extremely hostile*" and a video depicting this scene likely provide enough clues for a person to determine what "the Ziff" refers to, assuming that they are familiar with the concept of "chasing". Not only understanding word meanings, humans are also capable of applying newly learned words in novel ways, enabling imagination of new dynamics, such as envisioning a "fleeing Ziff" that runs away from the player.

Toward building world models with similar capabilities, we construct a benchmark based on the game of MESSENGER (Hanjie et al., 2021). In this benchmark, a model is given trajectory "videos" of games involving several entities interacting with each other. Each video is accompanied by language descriptions of the attributes of the entities. The model begins with almost zero language understanding and has to identify the entities and learn the grounded meanings of their attributes purely by watching the videos. At test time, it must demonstrate *compositional generalization* by being able to simulate environments featuring entities with attributes different from those it observes during training. For example, it has to portray a "fleeing mage" despite having only seen the mage chase the player in training games. We design three evaluation settings that test for incrementally greater degree of compositional generalization.

Despite its apparent simplicity, our benchmark covers many complications in building robust LWMs. We find that the prominent Transformer model (Vaswani et al., 2017) struggles in the harder evaluation settings. Even with a ground-truth disentangled representation of the observations, the model cannot learn generalizable grounding functions and yields minimal improvements in simulation quality compared to a model that ignores the language descriptions entirely. We augment the model with the EMMA attention (Hanjie et al., 2021), which mimics a two-step reasoning process. Our results confirm the effectiveness of this new architecture, as it robustly

generalizes even in the hardest evaluation setting, outperforming baselines by substantial margins in various evaluation metrics. It is even competitive with a skyline model with an oracle semantic parsing and grounding capability.

Last but not least, we illustrate a promising application of LWMs by simulating a cautious agent that, instead of performing a task right away, uses its LWM to generate an execution plan and asks a human to review it (Figure 1a). This form of pre-execution communication can potentially improve the agent's safety and transparency, following the spirit of the guaranteed safe AI approach proposed by Dalrymple et al. (2024). Moreover, it allows the human to improve the performance of the agent by revising the plan. In this setting, our LWM-based agent has the advantage of being able to assimilate *language feedback* describing the environment dynamics. We demonstrate that the language understanding capabilities of our proposed LWM are sufficient to enact this strategy. In the most challenging evaluation setting, without gathering additional interactions in the environment, the agent equipped with our model achieves an average reward three to four times higher than that of an agent using an observational world model.

We hope that our work will serve as a catalyst for exploring novel approaches to developing robust language-guided world models. More generally, we call for the design of modular agents whose components are parameterized by natural language. As previously argued, a modular design can dramatically boost communication efficiency, because the same component may be involved in the learning of various policies. We hypothesize that this approach can potentially surpass the efficiency of the currently prevalent approach that integrates language into a monolithic policy (e.g, Bisk et al. (2016); Misra et al. (2018); Anderson et al. (2018); Narasimhan et al. (2018); Hanjie et al. (2021); Zhong et al. (2021) and work on large language models like Ouyang et al. (2022)).

2 Background: world models

We consider a Markov Decision Process (MDP) environment E with state space $\mathcal{S}$, action space $\mathcal{A}$, and transition function $M : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S} \times \mathbb{R} \times \{0, 1\})$, where Δ denotes the set of all probability distributions over a set. An agent implementing a policy $\pi(a \mid s) : \mathcal{S} \to \Delta(\mathcal{A})$ interacts with the environment by choosing actions using its policy. Taking an action $a_t \sim \pi(s_t)$ in state s_t transitions the agent to a new state s_{t+1}, and incurs a reward r_{t+1} and a termination signal d_{t+1}, where $s_{t+1}, r_{t+1}, d_{t+1} \sim M(s_t, a_t)$.

A (one-step) *world model* M_θ (Robine et al., 2023; Micheli et al., 2023; Hafner et al., 2023) is an approximation of $M(s_{t+1}, r_{t+1}, d_{t+1} \mid s_t, a_t)$. A *model-based agent* uses data gathered in the environment to construct a world model and leverages it to learn policies for accomplishing tasks.[1] In contrast, a *model-free* agent learns its policies directly from data collected in the environment.

Model-based agents can require less effort to adapt. Because all policies of a model-based agent are derived from a shared world model, any modifications made to this model would affect all of them. This feature can be exploited to reduce human effort in controlling this type of agent. Specifically, suppose we concern m tasks in the environment, necessitating m policies. If there is a change in the environment dynamics, a model-based agent only needs task-agnostic data to replicate this change in its world model. It can then re-optimize its policies with respect to the updated model. Meanwhile, a model-free agent needs to collect task-specific data to re-train all of its m policies. The data collection cost of the model-free approach scales with m, whereas that of the model-based approach is independent of m, since the policy re-optimization step uses only data generated by the world model.

Observational world models. The dominant approach to world modeling learns a function $M_\theta(s_{t+1}, r_{t+1}, d_{t+1} \mid h_t)$ parameterized by a neural network θ and conditioned on a history $h_t = (s_1, r_1, d_1, a_1, \ldots, s_t, r_t, d_t, a_t)$. We refer to this class of models as *observational world models* because they can be adapted with only observational data, through either in-weight learning (updating the model parameters to fit a dataset of observations), or in-context learning (plugging in a history of observations).

Relying on observation-based adaptation leads to two drawbacks. First, controlling these models is difficult because observations are inadequate for conveying complex, abstract human intentions. Second, collecting observations requires taking real actions in the environment, which can be expensive, time-consuming, and risky.

[1] Note that M_θ includes a reward function but can be combined with any other reward function for learning.

3 Language-guided world models (LWMs)

We introduce LWMs, a new class of world models that can interpret language descriptions to simulate environment dynamics. These models address the drawbacks of observational world models. They allow humans to easily adapt their behavior through natural means of communication. Consequently, humans can effectively assist these models, significantly reducing the amount of interactive experiences that they need to collect in environments. In addition, these models can also leverage pre-existing texts written for humans, saving human effort to fine-tune them.

3.1 Formulation

We consider a family of environments $E(\boldsymbol{v})$ whose transition function has the form $M(s_{t+1}, r_{t+1}, d_{t+1} \mid h_t, \boldsymbol{v})$ where $\boldsymbol{v}$ is a parameter vector. Plugging in a specific $\boldsymbol{v}$ gives rise to an environment. We assume that each environment $E(\boldsymbol{v})$ is accompanied by a *language manual* $\boldsymbol{\ell} = (l_1, \cdots, l_N)$ consisting of language descriptions l_i. This manual describes $\boldsymbol{v}$ and the internal operations of M. Our goal is to learn a world model $M_\theta(s_{t+1}, r_{t+1}, d_{t+1} \mid h_t, \boldsymbol{\ell})$ that approximates the true dynamics $M(s_{t+1}, r_{t+1}, d_{t+1} \mid h_t, \boldsymbol{v})$.

The training data for our LWMs is a dataset $\{(\tau^i, \boldsymbol{\ell}^i)\}$ where τ^i is a trajectory generated in an environment $E(\boldsymbol{v}_i)$ with $\boldsymbol{v}_i$ drawn from some distribution P_{train}, and $\boldsymbol{\ell}^i$ is the accompanying manual. Each trajectory $\tau = (s_1, r_1, d_1, a_1, \ldots, s_T, r_T, d_T)$ is a sequence of states, actions, rewards, and termination signals. It can be viewed as a "video" that is annotated with actions and rewards. The trajectories are generated using a behavior policy, which can be a rule-based or learned policy, or a human.

3.2 Modeling entity-based environments

We view an environment as a set of C *entities* interacting with each other within a constrained space. Each entity c has a set of K *attributes*, each of which has value v_k^c. There is a special attribute called the identity of the entity (e.g., the name of a character or object in a video game). Each action triggers an event that changes a subset of attributes of a group of entities. The specific change is determined by the attributes of the entities involved in the event (e.g., an enemy entity attacks a player when colliding with them). In this work, we assume that each description in a manual portrays all attributes of an entity; hence, the number of descriptions N is equal to C.

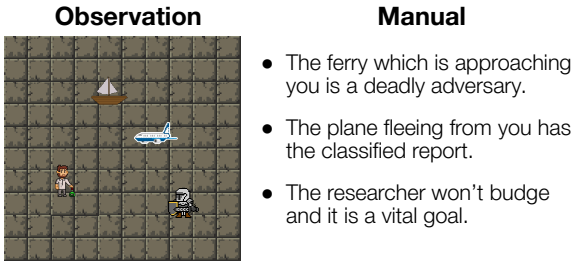


Figure 2: MESSENGER environment with manual.

Testing for compositional generalization. With this formulation, the environment parameters $\boldsymbol{v} = (v_1^1, \cdots, v_K^1, v_1^2, \cdots, v_1^C, \cdots, v_K^C)$ is a vector that contains the attributes of the C entities depicted in a manual. We are concerned with building LWMs that, at test time, can simulate environments whose paramerer vectors are compositionally novel. The term "compositionally novel" means that all components of the vector are individually seen during training, but the vector as a whole is previously unseen. This implies that the manuals at test time are also new.

This problem requires a LWM to be able to learn a representation of the transition function $M(\boldsymbol{v})$ by studying the language of the manuals, and to extract the specific parameters $\boldsymbol{v}$ described by each manual. The function $M(\boldsymbol{v})$ has two important properties. The first is the *independence* among its parameters because they represent orthogonal attributes. The second is the *locality* of the parameters, as each is an attribute associated with only a single entity. These properties make it difficult to recover the function exactly from purely observational data without injecting strong inductive biases into the learning model.

3.3 The MESSENGER-WM benchmark

The game of MESSENGER, developed by (Hanjie et al. (2021); Figure 2) exemplifies the class of environments discussed in the previous section. Despite being a simple grid-world environment, the dynamics possess the independence and locality properties that we want to study. In fact, it is our intention to use this visually simplistic environment to highlight the challenges in building LWMs that are orthogonal to the computer graphics challenge of mapping state representations to realistic-looking outputs.

Environment dynamics. The game takes place in a 10×10 grid world. A player interacts with entities of three *roles*: message, goal, and enemy. We use the stage-two version of the game, in which there are three entities, one of each role, in a game instance. In addition to the role, each entity is assigned an *identity* among twelve possibilities (mage, airplane, orb, etc.) and a *movement pattern* (chasing the agent, fleeing from the agent, immobile). The objective of the player is to acquire the message and deliver it to the goal while avoiding the enemy. Fetching the message is awarded 0.5 points and delivering it to the goal adds another point. If the player collides with the enemy or reaches the goal without carrying the message, the game ends, and the player receives -1 points.

Game manual. A game's manual consists of three descriptions corresponding to the three entities. MESSENGER provides a dataset of 5,316 language descriptions, each of which describes a combination of identity, role, and movement. The descriptions employ various linguistic expressions for each identity, role, or movement pattern (e.g., an airplane can be mentioned as a "plane", "jet", or "airliner"), making it non-trivial to interpret.

Evaluation settings. To test for compositional generalization, we construct three evaluation settings, ordered in increasing degree of difficulty:

- **NewCombo (easy).** Each game features a combination of three identities that were never seen together in a training game. However, the role and movement pattern of each identity are the same as during training.
- **NewAttr (medium).** The three identities were seen together in a training game, but each identity is assigned at least a new attribute (role, or movement pattern, or both).
- **NewAll (hard).** This setting combines the difficulties of the previous two. The identity triplet is novel, and each identity is assigned at least a new attribute.

To generate trajectories, we implement rule-based behavior policies that execute various intentions: act randomly, avoid the enemy, suicide (go to the enemy), obtain the message, and win the game (obtain the message and deliver it to the goal). We generate a total of 100K trajectories for training, each of which is generated by rolling out a uniformly randomly chosen rule-based policy. More details of the data are given in Appendix B. Our evaluation is more comprehensive than the original MESSENGER paper's evaluation, which does not construct different levels of compositional generalization, and is more difficult than the setting of Lin et al. (2024), which does not concern generalization.

To succeed in MESSENGER-WM, a model must be able to understand the non-trivial concepts represented by the attributes. For example, the concept of "chasing" involves planning actions to reduce the distance between two entities. The model must also capture the independence of the attributes, despite observing correlations in the training data (e.g., the "mage" is never immobile during training). Finally, to reflect the locality of the attributes, the model needs to learn a representation that disentangles the entities and to route attributes to the right entities. For example, the movement of one entity should not influence that of another. These are among the difficult, under-explored problems in machine learning, making MESSENGER-WM a respectable research challenge. We will empirically show that the state-of-the-art Transformer architecture struggles to perform well on the benchmark, suggesting that it may be insufficient for tackling more complex world-modeling problems.

4 Modeling approach

State representation. In MESSENGER, a state s is represented by an $H \times W$ grid with C channels (an $H \times W \times C$ tensor), where each channel corresponds to an entity. In each channel c, there is a single non-zero cell $s(h, w, c)$ that represents the identity of the entity. The position of this cell is the location of the entity in the grid. We note that this is an idealized representation that disentangles the entities. Even so, the problem remains challenging, as the model needs to recognize attributes mentioned in the manual and associate them with the right entity token. This requires a special attention mechanism, which we will introduce shortly. Meanwhile, learning entity-disentangled representations for pixel-based environments remain an open problem, which we defer to future work.

World modeling as sequence generation. Our model (illustrated in Figure 1b) is an encoder-decoder Transformer (Vaswani et al., 2017) which encodes a manual $\boldsymbol{\ell}$ and decodes a trajectory τ. We transform the trajectory into a long sequence of tokens and train the model as a sequence generator.

Concretely, our model processes a data point $(\tau, \boldsymbol{\ell})$ as follows. For the manual $\boldsymbol{\ell} = \{l_i\}_{i=1}^{N}$, we

first use a pre-trained BERT model to convert each description l_i into a sequence of hidden vectors. We feed each sequence to a Transformer encoder, which outputs a tensor $\boldsymbol{m}^{\text{enc}}$ of size $N \times L \times D$, where $N = C$ is the number of descriptions, L is the maximum number of words in a description, and D is the hidden size.

For the trajectory, we convert each tuple (a_{t-1}, s_t, r_t, d_t) into a token block B_t. The first action a_0 is set to be a special <s> token. Each state s_t is mapped to $3C$ tokens $(i_t^1, h_t^1, w_t^1, \cdots, i_t^C, h_t^C, w_t^C)$, which represents each of the C entities by its identity i followed by its location (h, w). The real-valued reward r_t is discretized into an integer label, and the termination signal d_t is translated into a binary label. In the end, B_t consists of $3C + 3$ tokens $(a_{t-1}, i_t^1, h_t^1, w_t^1, \cdots, i_t^C, h_t^C, w_t^C, r_t, d_t)$. Finally, we concatenate all T blocks in the trajectory into a sequence of $T \times (3C + 3)$ tokens, embed them into a $T \times (3C+3) \times D$ tensor, and add positional embeddings. We will use bold notation (e.g., $\boldsymbol{a}$, $\boldsymbol{i}$) to refer to the resultant embeddings of the tokens.

Entity mapper with multi-modal attention. We implement a variant of EMMA (Hanjie et al. (2021)) that first identifies the description that mentions each entity and extracts from it words corresponding to the attributes of the entity. From the tensor $\boldsymbol{m}_n^{\text{enc}}$ computed by the encoder, we generate a key tensor $\boldsymbol{m}^{\text{key}}$ and a value tensor $\boldsymbol{m}^{\text{val}}$, both of which are of size $N \times L \times D$, where

$$\begin{aligned} \boldsymbol{m}_n^{\text{key}} &= \texttt{Softmax}(\texttt{Linear}_{\text{key}}(\boldsymbol{m}_n^{\text{enc}})^\top)\boldsymbol{m}_n^{\text{enc}} \\ \boldsymbol{m}_n^{\text{val}} &= \texttt{Softmax}(\texttt{Linear}_{\text{val}}(\boldsymbol{m}_n^{\text{enc}})^\top)\boldsymbol{m}_n^{\text{enc}} \end{aligned} \quad (1)$$

for $1 \leq n \leq N$. Here, $\texttt{Linear}_{\text{key}}^{D\to1}$ and $\texttt{Linear}_{\text{val}}^{D\to1}$ are linear layers that transform the input's last dimension from D to 1, and $\texttt{Softmax}(\cdot)$ applies the softmax function to the last dimension. Intuitively, we want each $\boldsymbol{m}_n^{\text{key}}$ to retain words that signal the identity of the entity mentioned in the n-th description (e.g., *ferry*, *plane*, *researcher*), and $\boldsymbol{m}_n^{\text{val}}$ to retrieve words depicting the other attributes (e.g., *approaching*, *deadly*, *fleeing*).

Let $\boldsymbol{i}_t^c$ be the embedding of the identity of entity c. We perform a dot-product attention with $\boldsymbol{i}_t^c$ as the query, $\boldsymbol{m}^{\text{key}}$ as the set of keys, and $\boldsymbol{m}^{\text{val}}$ as the set of values to compute the attribute features of c

$$\boldsymbol{z}_t^c = \texttt{DotAttend}(\boldsymbol{i}_t^c, \boldsymbol{m}^{\text{key}}, \boldsymbol{m}^{\text{val}}) \quad (2)$$

The features are added to the identity tokens $\boldsymbol{i}_t^c$. The final input of the model is as follows:

$$(\boldsymbol{a}_{t-1}, (\boldsymbol{i}_t^c + \boldsymbol{z}_t^c, \boldsymbol{h}_t^c, \boldsymbol{w}_t^c)_{c=1}^C, \boldsymbol{r}_t, \boldsymbol{d}_t) \quad (3)$$

Unlike the standard encoder-decoder Transformer, our architecture does not perform cross-attention between the encoder and the decoder because information from the encoder has already been incorporated into the decoder through EMMA.

Model training. We train the model to minimize cross-entropy loss with respect to the ground-truth (tokenized) trajectories in the training set. The label at each output position is the next token in the ground-truth sequence. In particular, we do not compute the losses at the positions of the action tokens and the first block's tokens, because those tokens will be set during inference.

5 Experiments

5.1 Baselines

We compare our model, which we call EMMA-LWM, with the followings:

(a) **Observational** world model does not leverage textual information. It is identical to EMMA-LWM except that we zero out the manual representation $\boldsymbol{m}^{\text{enc}}$;

(b) **Standard** is the encoder-decoder Transformer model following Vaswani et al. (2017) with multi-headed cross-attention between the decoder and the encoder. Similarly to EMMA-LWM, the model uses BERT to initially encode the manual into hidden vectors. The encoder applies self-attention to the hidden vectors of each description separately, instead of joining all vectors into a sequence and applying self-attention to it;

(c) **GPTHard** is similar to EMMA-LWM but uses ChatGPT instead of EMMA to ground descriptions to entities. More details about this model are in Appendix A;

(d) **OracleParse** is the same as GPTHard, but uses an oracle information extraction function. A description like *"the crucial target is held by the wizard and the wizard is fleeing from you"* is converted into *"mage fleeing goal"* for this model.

We train all models using AdamW (Loshchilov and Hutter, 2017) for 10^5 iterations. For further details, please refer to Appendix C.

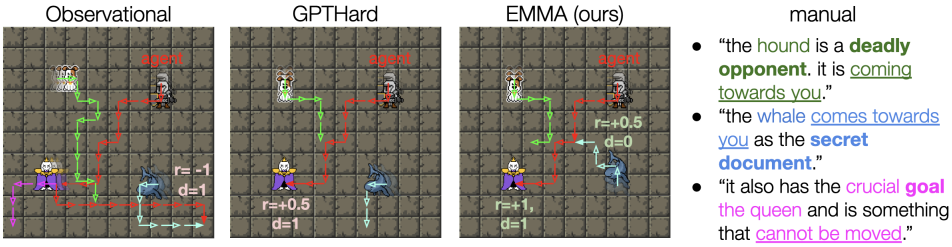


Figure 3: A qualitative example taken from the NewAll split. The `Observational` model mistakenly captures the movement patterns of the immobile queen goal and the chasing whale message. It also misrecognizes the whale as an enemy, predicting a wrong reward r and incorrectly predicting a termination state d after the player collides with this entity. The `GPTHard` model incorrectly identifies the queen as the message and predicts the whale to be fleeing. Meanwhile, our model `EMMA-LWM` accurately captures all of those roles and movements.

Table 1: Cross entropy losses (↓) of different models on test ground-truth trajectories. Note that the minimum loss is non-zero because the MESSENGER environment is stochastic. We run each model with five different random seeds, selecting the final checkpoint for each seed based on the loss in the development NewAll split. We report the mean losses with 95% t-value confidence intervals. The bold number in each column indicates the best non-oracle mean.

World model	NewCombo (easy)	NewAttr (medium)	NewAll (hard)
`Observational`	0.12 ± 0.04	0.18 ± 0.02	0.19 ± 0.01
`Standard`	0.10 ± 0.04	0.15 ± 0.04	0.16 ± 0.03
`GPTHard`	0.10 ± 0.02	0.15 ± 0.01	0.16 ± 0.00
`EMMA-LWM`	**0.08** ± **0.01**	**0.10** ± **0.02**	**0.13** ± **0.01**
`OracleParse`	0.08 ± 0.01	0.09 ± 0.02	0.12 ± 0.06

5.2 Results

Evaluation with ground-truth trajectories. Table 1 shows the cross-entropy losses of all models on ground-truth trajectories sampled from the true environment dynamics (more in Appendix E). In the more difficult NewAttr and NewAll splits, our `EMMA-LWM` model consistently outperforms all baselines, nearing the performance of the `OracleParse` model. As expected, the `Observational` model is easily fooled by spurious correlations between identity and attributes, and among attributes. A specific example is illustrated in Figure 3. There, the `Observational` model incorrectly captures the movement of the whale and the queen. It also mistakenly portrays the whale as an enemy, whereas, in fact, the entity holds the message. In contrast, `EMMA-LWM` is capable of interpreting the previously unseen manual and accurately simulates the dynamics.

The performance of the `Standard` model is sensitive to initialization; in some runs, it performs as well as `EMMA-LWM`, but in others it performs as badly as `Observational`. A plausible explanation is that the model's attention mechanism lacks sufficiently strong inductive biases to consistently find generalizable solutions. Our results agree with previous work on the lack of compositional generalizability of Transformers, which is often remedied by adding various forms of inductive bias (Keysers et al., 2020; Jiang and Bansal, 2021; Chaabouni et al., 2021; Dziri et al., 2023).

Another interesting finding is that the `GPTHard` model does not perform as well as expected. As a reminder, this model relies on ChatGPT to parse identities from descriptions and only needs to learn to extract attributes. Its underperformance compared to `EMMA-LWM` can be attributed to (i) the imperfection of ChatGPT in identifying identities in descriptions (its accuracy is around 90%; see Appendix B) and (ii) the fact that `EMMA-LWM` jointly learns to extract both identity and attribute words, which may be more effective than learning to extract only attribute words.

Evaluation with imaginary trajectories. In this evaluation, for each world model and test trajectory, we reset the model to the initial state of the trajectory and sequentially feed the actions in the trajectory to the model until it predicts the end of the episode. This process generates an imaginary trajectory. We refer to the evaluation trajectory as the real trajectory. We compute precisions of predicting non-zero rewards ($r \neq 0$) and terminations ($d = 1$). To evaluate movement prediction, we compare the distances from the player to an entity in the real and imaginary trajectories. Concretely, let $\delta^{\text{real}}_{c,t}$ and $\delta^{\text{imag}}_{c,t}$ be the Hamming distances from the player to entity c at the t-th time step in a real trajectory τ_{real} and an imaginary trajectory τ_{imag}, respectively. We cal-

Table 2: Results on imaginary trajectory generation. Δ_{dist} measures the similarity between the distances from the player to an entity in a real trajectory and the corresponding imaginary trajectory. The bold number in each column represents the best non-oracle result. `EMMA-LWM` outperforms all baselines in all metrics.

	Δ_{dist} (↓)			Non-zero reward precision (↑)			Termination precision (↑)		
World model	NewCombo (easy)	NewAttr (medium)	NewAll (hard)	NewCombo (easy)	NewAttr (medium)	NewAll (hard)	NewCombo (easy)	NewAttr (medium)	NewAll (hard)
`Observational`	2.04	2.91	3.00	0.39	0.20	0.15	0.51	0.33	0.28
`Standard`	0.82	1.48	1.68	0.68	0.43	0.50	0.75	0.55	0.62
`GPTHard`	0.89	2.74	2.89	0.75	0.34	0.25	0.79	0.45	0.45
`EMMA-LWM`	**0.57**	**1.14**	**1.29**	**0.88**	**0.69**	**0.70**	**0.88**	**0.75**	**0.71**
`OracleParse`	0.49	0.77	0.92	0.93	0.81	0.77	0.89	0.84	0.79

culate the average difference in a specific time step: $\Delta_{\text{dist}} = \frac{1}{|\mathcal{D}_{\text{eval}}|} \sum_{\tau_{\text{real}} \in \mathcal{D}_{\text{eval}}} \frac{1}{T_{\min}} \sum_{t=1}^{T_{\min}} |\delta_{c,t}^{\text{real}} - \delta_{c,t}^{\text{imag}}|$ where $\mathcal{D}_{\text{eval}}$ is an evaluation split, $T_{\min} = \min(|\tau_{\text{real}}|, |\tau_{\text{imag}}|)$, and τ_{imag} is generated from τ_{real} . For example, for a chasing entity, $\delta_{c,t}^{\text{real}}$ decreases as t increases. If a model mistakenly predicts the entity to be immobile, $\delta_{c,t}^{\text{imag}}$ remains a constant as t progresses. In this case, Δ_{dist} is non-negligible, indicating an error. All evaluation metrics are given in Table 2. The ordering of the models is similar to that in the evaluation with ground-truth trajectories. `EMMA-LWM` is still superior to all baselines in all metrics.

5.3 Application: agents that discuss plans with humans

In this section, we showcase the practicality of our LWM by illustrating that it can facilitate *plan discussions* between an agent and a human supervisor. This approach has the potential to improve the transparency, safety, and performance of real-world agents.

We imagine an agent ordered to perform a task in a previously unseen environment (Figure 1a). Letting the agent perform the task immediately would be extremely risky because of its imperfect knowledge of the environment. Implementing a world model enables the agent to imagine a solution trajectory and present it to a human as a *plan* for review. Conveying plans as trajectories helps the human envision the future behavior of the agent in the real world. Furthermore, the human can improve this behavior by providing feedback to enhance the policy that produces the plan.

A human can update the policy by telling the agent which actions it should have taken. This type of feedback can be incorporated using some form of imitation learning. An agent equipped with a LWM additionally enables the human to **update its policy by giving language feedback that aims to modify its world model**. Although an observational world model also allows this form of adaptation, it requires much more effort from the human to generate the feedback. Concretely, the human has to generate observations in the same format as those in the agent's plan (e.g., they have to draw grids in this setting). Furthermore, many abstract concepts may not be efficiently or precisely specified through non-verbal communication.

We simulate this scenario by placing agents with randomly initialized policies in test environments. These agents are forbidden to interact with the environments. However, they are equipped with world models, which allows for imaginary policy update. The world models are the ones we evaluated in the previous section. Importantly, the models were not trained on any data collected in the environments, simulating the fact that these environments are completely new to the agents.

We train all policies with imitation learning, considering two types of feedback: in *online imitation learning* (Ross et al., 2011), the expert suggests the best actions to take in the states present in the plan; in the *filtered behavior cloning* setting, the expert simply overwrites the agent's plan with their own plan. In the latter setting, the agent chooses the plans that achieve the highest returns according to their world models to imitate. We experiment with a near-optimal expert and a suboptimal expert. We provide more details in Appendix D.

The agents endowed with LWMs can also process language feedback aiming to change their world models. This feedback is simulated by the game manuals accompanying the environments. It serves as the input ℓ of the LWMs. We suppose that a human gives this feedback once to an agent, before adapting it via imitation learning.

We present the performance of the agents after adaptation in Table 3. Learning with the `Observational` world model amounts to the case

where the human provides only imitation-learning feedback and cannot adapt the world model via language. Meanwhile, learning with EMMA-LWM represents the case where the human can use language feedback to improve the world model. In all evaluation settings, we observe significant improvements in the average return of policies that adopt our EMMA-LWM. There are still considerable gaps compared to using the OracleParse model, indicating that our model still has room for improvement.

Table 3: Average returns (↑) in real environments of policies trained with imaginary imitation learning using world models. Bold numbers indicate the best non-oracle means in the corresponding settings. An expanded table with all models and details on how the metric was computed are available in Appendix E.

Setting	World model	NewCombo (easy)	NewAttr (medium)	NewAll (hard)
Online IL *(near-optimal)*	Observational	0.75 ± 0.16	-0.41 ± 0.21	-0.21 ± 0.21
	EMMA-LWM (ours)	**1.01** ± 0.12	**0.96** ± 0.17	**0.62** ± 0.21
	OracleParse	1.04 ± 0.13	0.85 ± 0.20	0.91 ± 0.18
Filtered BC *(near-optimal)*	Observational	0.77 ± 0.14	-0.42 ± 0.15	-0.30 ± 0.16
	EMMA-LWM (ours)	**1.18** ± 0.10	**0.75** ± 0.20	**0.44** ± 0.18
	OracleParse	1.17 ± 0.11	0.84 ± 0.19	0.80 ± 0.18
Filtered BC *(suboptimal)*	Observational	0.71 ± 0.15	-0.35 ± 0.18	-0.33 ± 0.17
	EMMA-LWM (ours)	**0.98** ± 0.13	**0.29** ± 0.25	**0.13** ± 0.19
	OracleParse	1.09 ± 0.13	0.50 ± 0.24	0.49 ± 0.18

6 Related work

World models. World models have a rich history dating back to the 1980s (Werbos, 1987). The base architecture has evolved from feed-forward neural networks (Werbos, 1987), to recurrent neural networks (Schmidhuber, 1990a,b, 1991), and most recently, Transformers (Robine et al., 2023; Micheli et al., 2023). In RL settings, world models are the key component of model-based approaches, which train policies in simulation to reduce the amount of interactions with real environments. Model-based RL has been successful in a variety of robotic tasks (Finn and Levine, 2017) and video games (Hafner et al., 2019, 2020, 2023). However, the incorporation of language information into world models has been underexplored. Cowen-Rivers and Naradowsky (2020) propose language-conditioned world models but focus on emergent language rather than human language. Poudel et al. (2023) incorporate features language into the representations of the model. These approaches, however, do not use language to control a world model.

Language-based adaptation. Language information has been incorporated into various aspects of learning. In instruction following (Bisk et al., 2016; Misra et al., 2018; Anderson et al., 2018; Nguyen and Daumé III, 2019), agents are given descriptions of the desired behaviors and learn to interpret them to perform tasks. Language-based learning (Nguyen et al., 2021; Scheurer et al., 2023) employs language-based feedback to train models. Another line of work uses language descriptions of environment dynamics to improve policy learning (Narasimhan et al., 2018; Branavan, 2012; Hanjie et al., 2021; Wu et al., 2023a; Nottingham et al., 2022; Zhong et al., 2020). Rather than using texts to directly improve a policy, our work leverages them to enhance a model of an environment. Recently, several papers propose agents that can read text manuals to play games (Wu et al., 2023a,b). Our work differs from these papers in that we aim to build models that capture exactly the transition function of an environment.

Compositional generalization for language-guided world models. Lin et al. (2024) model a variety of text-augmented environments but do not demonstrate the generalizability of their approach in MESSENGER. Recent work (Zhao et al., 2022; Du et al., 2024; Zhou et al., 2024; Zhang et al., 2024) has developed LWMs with compositional generalizability. While these papers operate on more visually realistic domains than ours, the language they study is simpler, focusing on concepts that correspond to straightforward mappings from input to output such as colors and objects. In contrast, the concepts in MESSENGER are more intricate, regarding interactions among multiple entities.

7 Conclusion

We introduce *Language-Guided World Models*, which can be adapted through natural language. We outline numerous advantages of these models over traditional observational world models. Our model is still lacking in performance and the grid-world environments we experiment with severely underrepresent the real world. Nevertheless, we hope that this work helps envision the potential of LWMs in enhancing the controllability of artificial agents and inspires future efforts to address the compositional generalization challenge.

Acknowledgements

We thank Ameet Deshpande, Vishvak Murahari, and Howard Chen from the Princeton NLP group for valuable feedback, comments, and discussions. We thank Kurtland Chua for helpful feedback. This material is based upon work supported by the National Science Foundation under Grant No. 2239363. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683.

Yonatan Bisk, Deniz Yuret, and Daniel Marcu. 2016. Natural language communication with robots. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 751–761.

SRK Branavan. 2012. Learning to win by reading manuals in a monte-carlo framework. *Journal of Artificial Intelligence Research*, 43:661–704.

Rahma Chaabouni, Roberto Dessì, and Eugene Kharitonov. 2021. Can transformers jump around right in natural language? assessing performance transfer from scan. In *BlackboxNLP workshop (EMNLP)*.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. 2018. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31.

Alexander I Cowen-Rivers and Jason Naradowsky. 2020. Emergent communication with world models. *arXiv e-prints*, pages arXiv–2002.

David Dalrymple, Joar Skalse, Yoshua Bengio, Stuart Russell, Max Tegmark, Sanjit Seshia, Steve Omohundro, Christian Szegedy, Ben Goldhaber, Nora Ammann, et al. 2024. Towards guaranteed safe ai: A framework for ensuring robust and reliable ai systems. *arXiv preprint arXiv:2405.06624*.

Marc Deisenroth and Carl E Rasmussen. 2011. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472.

Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. 2024. Learning universal policies via text-guided video generation. *Advances in Neural Information Processing Systems*, 36.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jian, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D Hwang, et al. 2023. Faith and fate: Limits of transformers on compositionality. In *Proceedings of Advances in Neural Information Processing Systems*.

Chelsea Finn and Sergey Levine. 2017. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE.

David Ha and Jürgen Schmidhuber. 2018. World models. *arXiv preprint arXiv:1803.10122*.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. 2019. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. 2020. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. 2023. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*.

Austin W Hanjie, Victor Y Zhong, and Karthik Narasimhan. 2021. Grounding language to entities and dynamics for generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 4051–4062. PMLR.

Yichen Jiang and Mohit Bansal. 2021. Inducing transformer's compositional generalization ability via auxiliary sequence prediction tasks. In *Proceedings of Empirical Methods in Natural Language Processing*.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, et al. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *Proceedings of the International Conference on Learning Representations*.

Jessy Lin, Yuqing Du, Olivia Watkins, Danijar Hafner, Pieter Abbeel, Dan Klein, and Anca Dragan. 2024. Learning to model the world with language. In *Proceedings of the International Conference of Machine Learning*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Vincent Micheli, Eloi Alonso, and François Fleuret. 2023. Transformers are sample-efficient world models. In *Proceedings of the International Conference on Learning Representations*.

Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. 2018. Mapping instructions to actions in 3d environments with visual goal prediction. *arXiv preprint arXiv:1809.00786*.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. 2018. Grounding language for transfer in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 63:849–874.

Khanh Nguyen and Hal Daumé III. 2019. Help, anna! visual navigation with natural multimodal assistance via retrospective curiosity-encouraging imitation learning. *arXiv preprint arXiv:1909.01871*.

Khanh X Nguyen, Dipendra Misra, Robert Schapire, Miroslav Dudík, and Patrick Shafto. 2021. Interactive learning from activity description. In *International Conference on Machine Learning*, pages 8096–8108. PMLR.

Kolby Nottingham, Alekhya Pyla, Sameer Singh, and Roy Fox. 2022. Learning to query internet text for informing reinforcement learning agents. *arXiv preprint arXiv:2205.13079*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Rudra PK Poudel, Harit Pandya, Chao Zhang, and Roberto Cipolla. 2023. Langwm: Language grounded world model. *arXiv preprint arXiv:2311.17593*.

Jan Robine, Marc Höftmann, Tobias Uelwer, and Stefan Harmeling. 2023. Transformer-based world models are happy with 100k interactions. In *Proceedings of the International Conference on Learning Representations*.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings.

Jérémy Scheurer, Jon Ander Campos, Tomasz Korbak, Jun Shern Chan, Angelica Chen, Kyunghyun Cho, and Ethan Perez. 2023. Training language models with language feedback at scale. *arXiv preprint arXiv:2303.16755*.

Jürgen Schmidhuber. 1990a. *Making the world differentiable: on using self supervised fully recurrent neural networks for dynamic reinforcement learning and planning in non-stationary environments*, volume 126. Inst. für Informatik.

Jürgen Schmidhuber. 1990b. An on-line algorithm for dynamic reinforcement learning and planning in reactive environments. In *1990 IJCNN international joint conference on neural networks*, pages 253–258. IEEE.

Jürgen Schmidhuber. 1991. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227.

Jürgen Schmidhuber. 2015. On learning to think: Algorithmic information theory for novel combinations of reinforcement learning controllers and recurrent neural world models. *arXiv preprint arXiv:1511.09249*.

Theodore R Sumers, Mark K Ho, Robert D Hawkins, and Thomas L Griffiths. 2023. Show or tell? exploring when (and why) teaching with language outperforms demonstration. *Cognition*, 232:105326.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Paul J Werbos. 1987. Learning how the world works: Specifications for predictive networks in robots and brains. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics, NY*.

Yue Wu, Yewen Fan, Paul Pu Liang, Amos Azaria, Yuanzhi Li, and Tom Mitchell. 2023a. Read and reap the rewards: Learning to play atari with the help of instruction manuals. In *Workshop on Reincarnating Reinforcement Learning at ICLR 2023*.

Yue Wu, So Yeon Min, Shrimai Prabhumoye, Yonatan Bisk, Ruslan Salakhutdinov, Amos Azaria, Tom Mitchell, and Yuanzhi Li. 2023b. Spring: Studying papers and reasoning to play games. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Hongxin Zhang, Zeyuan Wang, Qiushi Lyu, Zheyuan Zhang, Sunli Chen, Tianmin Shu, Yilun Du, and Chuang Gan. 2024. Combo: Compositional world models for embodied multi-agent cooperation. *arXiv preprint arXiv:2404.10775*.

Linfeng Zhao, Lingzhi Kong, Robin Walters, and Lawson LS Wong. 2022. Toward compositional generalization in object-oriented world modeling. In *International Conference on Machine Learning*, pages 26841–26864. PMLR.

Ruijie Zheng, Khanh Nguyen, Hal Daumé III, Furong Huang, and Karthik Narasimhan. 2023. Progressively efficient learning. *arXiv preprint arXiv:2310.13004*.

Victor Zhong, Austin W. Hanjie, Sida I. Wang, Karthik Narasimhan, and Luke Zettlemoyer. 2021. Silg: The multi-environment symbolic interactive language grounding benchmark. In *Neural Information Processing Systems (NeurIPS)*.

Victor Zhong, Tim Rocktäschel, and Edward Grefenstette. 2020. Rtfm: Generalising to new environment dynamics via reading. In *International Conference on Learning Representations*.

Siyuan Zhou, Yilun Du, Jiaben Chen, Yandong Li, Dit-Yan Yeung, and Chuang Gan. 2024. Robodreamer: Learning compositional world models for robot imagination. *arXiv preprint arXiv:2404.12377*.

A GPTHard model

This approach leverages the language-understanding capabilities of ChatGPT. Through few-shot prompting, we instruct this model to determine the identity of the entity mentioned in each manual description. In this approach, we generate only the set of values $\boldsymbol{m}^{\text{val}}$ as in Eq 1. Instead of learning soft attention, we directly route the values to the identity embeddings. Concretely, the feature vector added to $\boldsymbol{i}_t^c$ in Eq 3 is $\boldsymbol{z}_t^c = \boldsymbol{m}_{j_c}^{\text{val}}$ where j_c is the index of the description that mentions entity c according to ChatGPT.

We compose the following prompt for parsing descriptions. We use the "May 3, 2023" release of ChatGPT. We feed to the model one description at a time instead of a whole manual of three descriptions. We ask it to also extract the role and movement pattern, but use only the parsed identity in the `GPTHard` model. The "ChatGPT identity-parsing" column in Table 4 shows the fraction of games in each split in which ChatGPT correctly identifies all three identities in a game. Note that the `OracleParse` model uses the ground-truth parses rather than these parses.

```
You are playing a role-playing video
    game where you will need to read
    textual descriptions to figure out
    the attributes of a character.

This is a list of characters and their
    corresponding IDs:
airplane: 2
mage: 3
dog: 4
bird: 5
fish: 6
scientist: 7
thief: 8
ship: 9
ball: 10
robot: 11
queen: 12
sword: 13

This is a list of movement types and
    their corresponding IDs:
chasing: 0
fleeing: 1
stationary: 2

This is a list of role types and their
    corresponding IDs:
dangerous enemy: 0
secret message: 1
essential objective: 2

Now, read a description and tell me
    which character is being mentioned
    and what are its movement type and
    role type. Your answer should follow
     this format:
```
```
Answer: Character ID, movement type ID,
    role type ID
```

```
Here are a few examples:

Description: the plane that's flying
    near where you are is the critical
    objective.
Answer: 2, 0, 2

Description: the escaping humanoid is an
     important goal.
Answer: 11, 1, 2

Description: the mage is inching near
    you is a lethal opponent.
Answer: 3, 0, 0

Description: the classified document is
    the hound coming your way.
Answer: 4, 0, 1

Description: the important goal is the
    orb which is creeping close to you.
Answer: 10, 0, 2

Now provide the answer for the following
     description. Follow the format of
    the previous answers:

Description: [PLACEHOLDER]
```

B Dataset

Statistics of our dataset are provided in Table 4. The maximum trajectory length is 32. We implement five rule-based behavior policies: `survive` (avoid the enemy and goal), `win the game`, `suicide` (go to the enemy), `obtain the message`, and `act randomly`. The `survive` policy acts randomly when the distances to the enemy and the goal are greater than or equal to 6. Otherwise, it takes the action that makes its distance to those entities at least 3. If that is impossible, it chooses the action that maximizes the minimum distance to one of the two entities. The `win the game` policy is not optimal: it simply aims to obtain the message and then run to the goal, without having a strategy to avoid the enemy. We run a breadth-first search to find the next best action to get to an entity.

For the training split, we generate 66 trajectories per game. The behavior policy for each trajectory is chosen uniformly randomly among the five rule-based policies. For each evaluation split, we generate 5 trajectories per game, using every rule-based policy to generate trajectories.

Split		Unique games	Unique descriptions	Trajectories	ChatGPT identity-parsing accuracy (%)
Train		1,536	986	101,376	92
Dev	NewCombo	896	598	4,480	89
	NewAttr	204	319	1,020	88
	NewAll	856	1,028	4,280	86
Test	NewCombo	896	587	4,480	90
	NewAttr	204	306	1,020	93
	NewAll	856	1,016	4,280	88

Table 4: MESSENGER data statistics. The last column shows the fraction of games in each split in which ChatGPT correctly identifies all three identities in a game.

Hyperparameter	Value
Hidden size	256
Number of encoder layers	4
Number of decoder layers	4
Number of decoder token blocks	33
Dropout rate	0.1
Batch size	32
Number of training batches	100K
Evaluation every	500 batches
Optimizer	AdamW
Learning rate	1e-4
Max. gradient norm	10

Table 5: Training hyperparameters.

C Training details

Our implementation of Transformer is largely based on the IRIS codebase (Micheli et al., 2023).[2] We implement cross-attention for the `Standard` baseline, and EMMA for our model.

Initialization. We find that the default PyTorch initialization scheme does not suffice for our model to generalize compositionally. We adopt the following initialization scheme from the IRIS codebase:

```
def init_weights(module):
    if isinstance(module, (nn.Linear, nn.Embedding)):
        module.weight.data.normal_(mean=0.0, std=0.02)
        if isinstance(module, nn.Linear) and module.bias is not None:
            module.bias.data.zero_()
    elif isinstance(module, nn.LayerNorm):
        module.bias.data.zero_()
        module.weight.data.fill_(1.0)
```

which is evoked by calling `self.apply(init_weights)` in the model's constructor. We initialize all models with this scheme, but only `EMMA-LWM` and `OracleParse` perform well consistently on various random seeds.

Compute resources. Experiments were primarily run on a cluster of NVIDIA RTX2080 GPUs, and each experiment was run on a single device. To generate Table 1, we trained each world model for 24 GPU hours, 5 seeds each. To generate Table 3 and 6, we trained each of the 5 world models on each of the 90 games (3 difficulties for 30 game configurations) using the 3 different downstream policy training strategies, with each game being 12 GPU hours.

D Imitation learning experiments

The learning policy follows the EMMA-based policy architecture of (Hanjie et al., 2021), which at each time step processes a stack of 3 most recent observations with a convolution-then-MLP encoder. We train the policy with 2,000 batches using the same optimizer hyperparameters as those of the world models.

For the online IL setting, we use the `win the game` rule-based policy (Appendix B) as the expert. For the filtered BC setting, we train an EMMA policy to overfit the test environment. We then use a fully converged checkpoint of the policy as the near-optimal expert, and a not fully converged checkpoint as the suboptimal expert. The former is trained for 10,000 iterations and the latter is trained for 2,000 iterations.

The test environments are randomly chosen from the test splits. We select 10 environments per split. We evaluate each policy for 48 episodes in the real environment. These episodes cover all 24 initial configurations of a stage-two MESSENGER game.

E Extended results

Figure 4 studies the performance of the models when conditioned on prefixes of the ground-truth

[2] https://github.com/eloialonso/iris

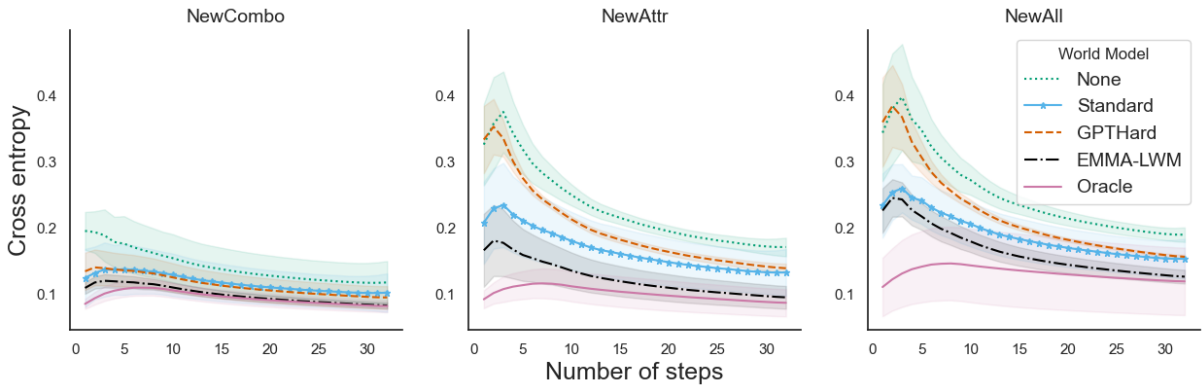


Figure 4: The cross entropy losses of the models when conditioned on ground-truth trajectory prefixes up to a certain length. We plot the means with 95% t-value confidence intervals. The losses generally decrease as the prefix length increases. `EMMA-LWM` outperforms baselines given any prefix length.

trajectories. The losses of all models decrease as the prefix length increases, but the baselines cannot close the gaps with `EMMA-LWM`. Across all splits, `EMMA-LWM` conditioned on a one-step history outperforms `Observational` conditioned on one third of a ground-truth trajectory, demonstrating that our model has effectively leveraged the textual information.

Table 6 presents the results of all the models in the simulation of plan discussion (§5.3).

Table 6: Average returns (↑) in real environments of policies trained with imaginary imitation learning using world models. For each world model type, we use the best checkpoint of a run chosen randomly among the five runs mentioned in Table 1. Experiments are conducted in 90 environments randomly chosen from the test splits (30 from each split). For each environment and learned policy, we compute the average return over 48 runs. For each split, we report the means of the average returns in the 30 environments with 95% t-value confidence intervals. Bold numbers indicate the best non-oracle means in the corresponding settings. `EMMA-LWM` outperforms all baselines in all settings.

Setting	World model	NewCombo (easy)	NewAttr (medium)	NewAll (hard)
Online IL *(near-optimal expert)*	`Observational`	0.75 ± 0.16	-0.41 ± 0.21	-0.21 ± 0.21
	`Standard`	0.93 ± 0.13	0.04 ± 0.26	0.30 ± 0.22
	`GPTHard`	0.82 ± 0.15	-0.20 ± 0.20	-0.06 ± 0.21
	`EMMA-LWM` (ours)	**1.01** ± **0.12**	**0.96** ± **0.17**	**0.62** ± **0.21**
	`OracleParse`	1.04 ± 0.13	0.85 ± 0.20	0.91 ± 0.18
Filtered BC *(near-optimal expert)*	`Observational`	0.77 ± 0.14	-0.42 ± 0.15	-0.30 ± 0.16
	`Standard`	1.05 ± 0.14	0.20 ± 0.27	0.17 ± 0.20
	`GPTHard`	0.79 ± 0.15	-0.10 ± 0.20	-0.07 ± 0.20
	`EMMA-LWM` (ours)	**1.18** ± **0.10**	**0.75** ± **0.20**	**0.44** ± **0.18**
	`OracleParse`	1.17 ± 0.11	0.84 ± 0.19	0.80 ± 0.18
Filtered BC *(suboptimal expert)*	`Observational`	0.71 ± 0.15	-0.35 ± 0.18	-0.33 ± 0.17
	`Standard`	0.68 ± 0.15	-0.15 ± 0.21	-0.10 ± 0.17
	`GPTHard`	0.75 ± 0.22	0.05 ± 0.25	0.06 ± 0.17
	`EMMA-LWM` (ours)	**0.98** ± **0.13**	**0.29** ± **0.25**	**0.13** ± **0.19**
	`OracleParse`	1.09 ± 0.13	0.50 ± 0.24	0.49 ± 0.18