Language Modeling with Latent Situations

Belinda Z. Li   **Maxwell Nye**[*]   **Jacob Andreas**
Massachusetts Institute of Technology
{bzl,mnye,jda}@mit.edu

Abstract

Language models (LMs) often generate incoherent outputs: they refer to events and entity states that are incompatible with the state of the world described in inputs. We introduce SITUATIONSUPERVISION, a family of approaches for improving coherence in LMs by training them to construct and condition on explicit representations of entities and their states. SITUATIONSUPERVISION has two components: an **auxiliary situation modeling** task that trains models to predict entity state representations in context, and a **latent state inference** procedure that imputes these states from partially annotated training data. SITUATIONSUPERVISION can be applied via fine-tuning (by supervising LMs to encode state variables in their hidden representations) and prompting (by inducing LMs to interleave textual descriptions of entity states with output text). In both cases, it requires only a small number of state annotations to produce substantial coherence improvements (up to an 16% reduction in errors), showing that standard LMs can be efficiently adapted to explicitly model language and aspects of its meaning.[1]

1 Introduction

Recent years have seen dramatic improvements in the quality of text generated by neural language models (LMs; Brown et al., 2020; Raffel et al., 2020). Nevertheless, even the best LMs still suffer from **failures of semantic coherence**. Samples from LMs refer to entities that have not yet been mentioned, assert contradictory facts, or describe impossible sequences of events (Marcus and Davis, 2020). This paper introduces SITUATIONSUPERVISION, a family of methods for efficiently mitigating incoherent language generation. SITUATIONSUPERVISION adapts pre-trained LMs to **explicitly**
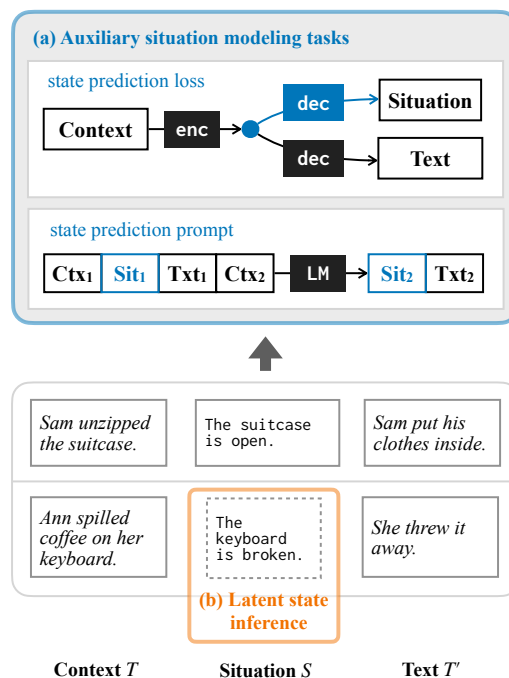
Figure 1: Language modeling with SITUATIONSUPERVISION, which comprises two components: (a) An auxiliary situation modeling task: given contexts annotated with explicit textual representations of the situations they describe, we use them to adapt LMs through either an auxiliary fine-tuning loss or a scratchpad-style prompt. (b) A latent state inference procedure, whereby missing situation annotations are semi-supervisedly inferred, enabling auxiliary situation modeling starting from a small number of seed situation annotations.

model the situations they describe by tracking the properties and relations of entities in generated text. The core of this approach is an auxiliary situation modeling task that trains LMs to predict textual representations of entity state jointly with target text. Unlike prior work in state tracking focused predominantly on *reasoning* (where the end task is to answer questions about the state, or to solve math or coding problems), we focus on using state tracking to improve *language generation*.

For most generation tasks, state information is not readily available: it must be manually annotated

and is costly to collect. Thus, to make auxiliary situation modeling for generation practical, we additionally introduce a semi-supervised procedure for inferring entity states in unannotated text, making it possible to apply SITUATIONSUPERVISION very small number of initial annotations.

Modern LMs can be specialized to new tasks in a variety of ways, including fine-tuning their parameters and modifying their prompts. We develop versions of SITUATIONSUPERVISION suitable for both adaptation methods. For fine-tuned models, we introduce an *auxiliary state prediction loss* that encourages models' hidden representations to encode state variables. For prompted models, we introduce a *scratchpad* approach that instructs models to generate explicit textual descriptions of world states prior to generating output text. Both approaches ultimately yield ordinary LMs, compatible with standard pre-training and decoding procedures.

We evaluate SITUATIONSUPERVISION on two challenging text generation tasks: the TextWorld (TW) task of generating acceptable next actions in a text-adventure game (Côté et al., 2018), and the TRIP task of evaluating commonsense physical plausibility of short (5-sentence) stories (Storks et al., 2021). In experiments on fine-tuned BART LMs (Lewis et al., 2020), applying SITUATION-SUPERVISION with 500 seed annotations reduces coherence errors by 5% on TW and 15% on TRIP. In experiments on prompted GPT-3 models (Brown et al., 2020), 12 seed annotations reduce coherence errors by 9% on TW and 20 seed annotations reduce errors by 16% on TRIP. In both cases, it is far more sample-efficient to annotate entity states in existing training samples than to augment training data with additional text-only samples: in fine-tuned models, SITUATIONSUPERVISION with 500 state annotations performs comparably to training on 9000 more text-only sentences, while in prompted models, devoting a fixed token budget to state annotations rather than additional text samples yields a coherence improvement of up to 10%.

Additional experiments characterize the ingredients of a good situation representation, showing that training LMs to predict *causally relevant* state variables is essential for good performance. Because the latent state inference objective favors representations that improve LM predictions, SITUATIONSUPERVISION discovers these variables automatically, sometimes improving on human-designed state representations. In summary:

1. We show that training LMs to build explicit representations of entity state (via auxiliary losses or scratchpad-based prompting) improves coherence in text generation tasks.

2. We describe new algorithms for *semi-supervised* learning of state representations, enabling auxiliary supervision and scratchpad techniques to be applied with extremely small numbers of annotations.

Our results show that, even when LMs struggle to generate coherent continuations of input text, only a small amount of supervision is needed to train them to explicitly represent the situations that their inputs describe. Once predicted, these representations in turn confer large improvements in LM coherence itself.

2 Background and Preliminaries

A **language model** (LM) encodes a distribution $p(T' \mid T)$ over texts T' given contexts T (Fig. 1). Today, most LMs are implemented as deep neural networks trained on massive text corpora (Brown et al., 2020). Sampling from them produces naturalistic text that often resembles human-generated language. However, LM generation is prone to several failure modes, including generation of text that is incoherent, untruthful, or unreliable (Zhou et al., 2021; Maynez et al., 2020; Martindale et al., 2019). Past work has shown that some of these behaviors stem from models' failure to build good representations, both of entities' default properties (Onoe et al., 2021) and state changes in context (Zellers et al., 2021). Humans' ability to avoid these failure modes, and to generate truthful and coherent text, is generally understood to rest upon explicit mental representations of the *situations* that language communicates. The nature and structure of these representations remains an ongoing topic of research in linguistics and cognitive science, but existing theories broadly agree that language users maintain explicit beliefs about the properties of and relations among entities mentioned in a discourse, updating these beliefs in response to new observations or new information conveyed in language (e.g. Kratzer, 2007; Zwaan and Pecher, 2012).

These representational theories suggest that language models $p(T' \mid T)$ may also benefit from explicit modeling of situation state. Given an input text T, we wish to first represent the **situation** S described by T before predicting a next sentence.

Inspired by models of situation semantics in the linguistics literature (Barwise and Perry, 1981, *inter alia*) we propose to model situations as sets of propositions s_i that are known or inferable about entities in a discourse.[2] Examples, with propositions expressed as sentences in natural language, are shown in Fig. 1(b) and Fig. 2.

Having inferred S from T, we may then condition on it when sampling T' from $p(T' \mid S, T)$. Past work has proposed a number of language generation models that explicitly model the state of the world, primarily by developing specialized prediction architectures that maintain internal state representations (Henaff et al., 2016; Gupta and Durrett, 2019) or interact with outside simulation engines (Liu et al., 2022). While effective, these approaches come at a cost—requiring complex training data (Mishra et al., 2018), limiting models to narrow, pre-defined domains, and generally precluding the large-scale (text-only) pretraining responsible for many of the greatest successes of current LMs. The main question this paper seeks to answer is whether the benefits of explicit world modeling may be obtained entirely within the language modeling paradigm itself, without specialized model architectures or large amounts of specialized supervision.

We do so by adapting pre-trained LMs to better represent situations S. There are two standard frameworks for LM adaptation. In smaller models, which are generally adapted by **fine-tuning** of model parameters, we develop auxiliary loss functions that encourage models' hidden states to contain the information required to generate textual descriptions of state.[3] In larger models, which can also be **prompted** by prepending a task description or set of examples to the input context, we develop prompts that induce models to generate textual state descriptions in LM output itself. Our research builds on a large body of work that uses auxiliary prediction tasks to shape model representations, notably work using "scaffold" decoders to shape model representations of syntax (Swayamdipta et al., 2018; Wilcox et al., 2019), and and "scratchpad" or "chain-of-thought" approaches to perform intermediate computations in models'

---

[2]This approach to modeling contrasts with approaches that implicitly or explicitly represent the complete set of possible worlds consistent with a text.

[3]Concurrent work by Richardson et al. (2022) also introduces a fine-tuning objective aimed at improving state representations, but focuses on state-tracking tasks rather than generation, and only examines a fully supervised setting.

output spaces (Camburu et al., 2018; Nye et al., 2021; Wei et al., 2022). In §3, we show how to adapt both techniques for a new class of text generation problems.

Adapting LMs with auxiliary prediction tasks requires a source of data for auxiliary supervision. This kind of supervision is uniquely difficult to obtain for generation tasks. But the probabilistic framing described above makes it natural to formulate language modeling with explicit situations as a latent variable problem. At training time, we may use context T and targets T' to guide inference of the unknown S from which T' was predicted. Once inferred, these states supervise the representation-building model that predicts S from T alone. As above, a great deal of past work has focused on treating string-valued prompts or plans as latent variables (Sharma et al., 2021; Zelikman et al., 2022; Sun et al., 2022). In §4, we generalize these methods to support multi-step text generation, and show that inferred states can be used to supervise small models as well as prompt large ones.

3 Auxiliary Situation Modeling

We begin by assuming access to a pre-trained LM and two sources of supervision: a dataset $\mathcal{X}_U$ of text examples of the form (T, T'), and a smaller dataset $\mathcal{X}_A$ of examples (T, S, T') annotated with textual situation descriptions S. Our full training data $\mathcal{X}$ is thus $\mathcal{X}_U \cup \mathcal{X}_A$. As depicted in Fig. 2, we take these situation descriptions to consist of declarative sentences about the properties and relations of entities that are relevant to the text being generated. In this section, we describe two auxiliary prediction schemes that use these annotations to improve the LM's ability to model the conditional text distribution $p(T' \mid T)$.

3.1 Situation Modeling for Fine-tuning

Our first approach uses a *auxiliary decoding loss* that encourages context representations to directly encode entity state information. We focus on encoder–decoder models consisting of an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$, with $\mathcal{D}(\mathcal{E}(T))$ producing as output a probability distribution over next sentences T'. In standard training, parameters of $\mathcal{E}$ and $\mathcal{D}$ are chosen to maximize:

$$\mathcal{L}(T'|T) = \log p(T'|T) = \log \mathcal{D}(T' \mid \mathcal{E}(T)) \quad (1)$$

To improve state representations, we add an **auxiliary loss**. This takes the form of an auxiliary
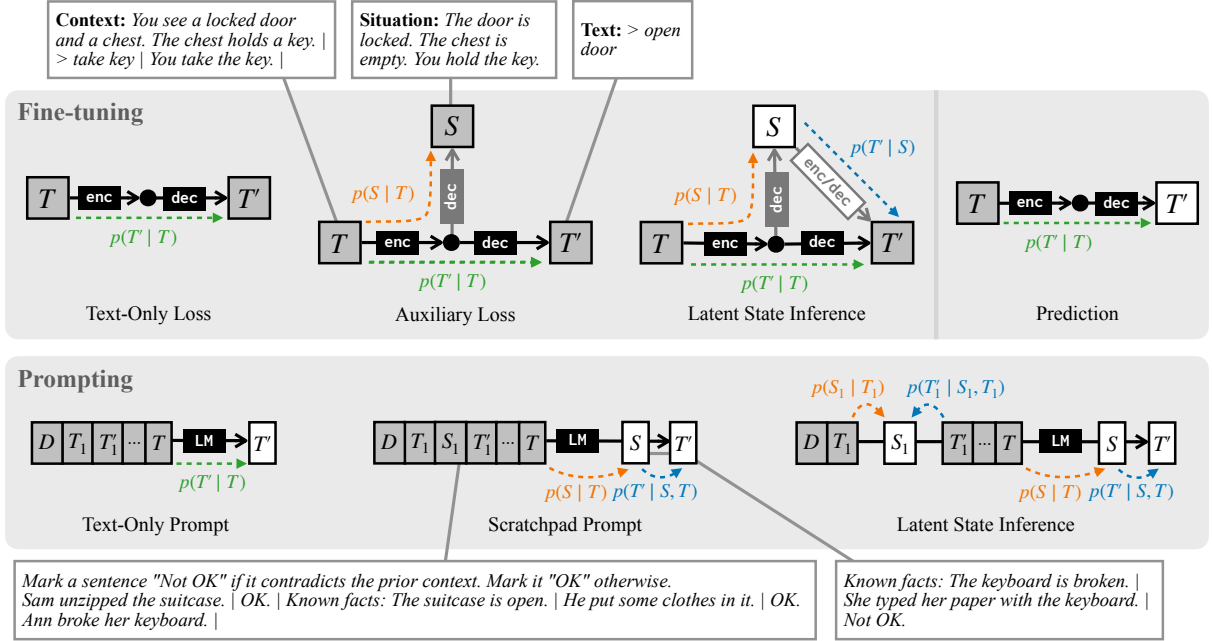
Figure 2: Fine-tuning (top) and prompting (bottom) with SITUATIONSUPERVISION. For each, we show: (Left) Text-only supervision, in which the LM is trained/prompted with text only samples and expected to produce next sentences T' from contexts T. (Middle) Auxiliary situation modeling, in which the LM is given state descriptions S during training or in the prompt and expected to learn to encode it in its parameters or infer it in-context. (Right) Latent state inference, in which the LM must infer missing state descriptions in the training data or prompt demonstrations. Finally, in fine-tuning, we discard the auxiliary decoder during inference, use the shared encoder and text-only decoder to infer the next sentence from the context.

decoder $\mathcal{D}_{S|T}$ (distinct from the original decoder $\mathcal{D}$) which is trained to predict state representations S from the encoded context $\mathcal{E}(T)$. We define:

$$\mathcal{L}(S|T) = \log p(S|T) = \log \mathcal{D}_{S|T}(S|\mathcal{E}(T)) \quad (2)$$

and train parameters of the encoder ($\theta_\mathcal{E}$) and both decoders ($\theta_\mathcal{D}, \theta_{\mathcal{D}_{T,S}}$) to maximize:

$$\arg\max_{\theta_\mathcal{E}, \theta_\mathcal{D}, \theta_{\mathcal{D}_{T,S}}} \sum_{T, T' \in \mathcal{X}} \mathcal{L}(T'|T)$$
$$+ \sum_{T, S \in \mathcal{X}_A} \mathcal{L}(S|T) \quad (3)$$

Intuitively, to minimize this objective, the output of $\mathcal{E}(T)$ must encode information about the latent situation S. Once encoded, this information is accessible to the original LM text decoder $\mathcal{D}$. Eq. (3) is a straightforward application of standard multi-task training approaches for deep networks; however, to the best of our knowledge it has not previously been used for state prediction tasks or shown to improve LMs' factual coherence.

3.2 Situation Prediction for Prompting

The approach described above is general. But in LMs with very large numbers of parameters, it

might be costly to apply (or we may risk over-fitting if the fine-tuning dataset is too small). Thus, the second approach we describe is based on *prompting* models to construct better state representations. We build on the observation in recent work that prompts can induce models to build better task representations by writing these representations to output: generating, then conditioning on, textual encodings of useful intermediate variables.

To induce language models to output textual situation descriptions, we construct prompts with three components: a task description D, a set of task demonstrations ("training set") $\mathcal{X}$, and an input context T_{pred}. The training set can include both unannotated and annotated examples: unannotated examples are sequences T_i, T'_i, while annotated examples are sequences T_i, S_i, T'_i. Formally, we construct a prompt string:

$$\mathcal{P} = [D \cdot \mathcal{P}_A \cdot \mathcal{P}_U \cdot T_{pred}], \quad \text{where:}$$
$$\mathcal{P}_A = [T'_0 \cdot S_1 \cdot T'_1 \cdots S_n \cdot T'_n]_x \quad \forall x \in \mathcal{X}_A$$
$$\mathcal{P}_U = [T'_0 \cdot T'_1 \cdots T'_n]_x \quad \forall x \in \mathcal{X} \quad (4)$$

with $\cdot$ denoting string concatenation. To enable the model to *predict* annotations and text directly, each S is prefixed with an appropriate control token

that informs the model that a situation description string will come next. When predicting (or scoring) a sentence T'_{pred} in context, we first prompt the model to generate a situation representation S_{pred}, then score T'_{pred} conditional on T_{pred}, S_{pred}, and the entire preceding context. The bottom portion of Fig. 2 shows a concrete example from the TRIP domain. As above, this approach to prompting is closely related to existing "scratchpad" and "chain-of-thought" methods used for question answering and formal reasoning tasks; our auxiliary situation modeling approach applies this form of structured prompting to multi-sentence, open-ended text generation problems.

4 Latent State Inference

The methods described in §3 applied SITUA-TIONSUPERVISION only to examples for which a ground-truth state annotation was provided. For these methods to be effective, enough state annotations must be available to provide a useful training signal in the auxiliary loss or to specify the auxiliary prediction task for the prompted model. But such state annotations are in general both *hard to collect* and *hard to design*.

In this section we describe how to obtain them automatically, without the need for large amounts of annotation. Below, we re-formulate the two approaches in §3 as *latent variable* models that can infer and condition on state representations even for unannotated training documents. Intuitively, this inference problem is easier at training time than prediction time: knowing what text followed a context constrains the possible situations the context could describe. Most work on semi-supervised inference of auxiliary prediction targets has focused on automatic optimization of prompts and reasoning chains (Zelikman et al., 2022; Sun et al., 2022). To the best of our knowledge, inferred latent variables have not been used to train auxiliary decoders or to design intermediate state representation for multi-step text generation. The techniques described below are quite general, and might be productively employed beyond the generation applications we describe here.

4.1 Latent State Inference for Fine-Tuning

Intuitively, a good situation representation is one that is both predictable from context, and useful for predicting subsequent text. To guide inference of entity states for auxiliary prediction, we intro-

duce another encoder-decoder into the model of §3.1: one which attempts to predict T' from S. This model now has two pathways for predicting T': one that uses encoder representations to predict it directly from T, and another which generates textual situation descriptions S from decoder representations, then uses these to predict T'. We train this model's parameters and infer situation description that maximize probability of next sentences under both pathways, using information from both T and T' to infer situations S, then using these to supervise the encoder.

Formally, we optimize the complete likelihood:

$$
\begin{aligned}
\arg\max_{\Theta,\hat{S}} \sum_{\substack{T,T' \\ \in \mathcal{X}}} & \mathcal{L}(T'|T) \\
+ \sum_{\substack{T,T',S \\ \in \mathcal{X}_A}} & \left(\mathcal{L}(S|T) + \mathcal{L}(T'|S,T) \right) \\
+ \sum_{\substack{T,T',\hat{S} \\ \in \mathcal{X}_U}} & \left(\mathcal{L}(\hat{S}|T) + \mathcal{L}(T'|\hat{S},T) \right) . \quad (5)
\end{aligned}
$$

Eq. (5) extends auxiliary fine-tuning by concurrently training an encoder-decoder $\mathcal{M}_{T'|S,T}$ to model $p(T' \mid S,T)$. We initialize $\theta_{\mathcal{E}}, \theta_{\mathcal{D}}, \theta_{\mathcal{D}_{S|T}}$ using Eq. (3), and $\theta_{T'|S}$ by fine-tuning to convergence on $\mathcal{X}_A$. We then perform coordinate ascent ("hard EM") by alternating between:

1. E-step: Set $\hat{S} \approx \arg\max_S p(S \mid T)p(T' \mid S)$ for $\mathcal{X}_U$ by sampling from $p(S \mid T)$, then reranking according to $p(S \mid T)p(T' \mid S)$.

2. M-step: Using the new $\hat{S}$, train Θ to maximize Eq. (5). Rather than training to convergence, we perform SGD on Eq. (5) for five epochs.

As in auxiliary fine-tuning, $\mathcal{E}$ is shared the $p(T' \mid T)$ and $p(S \mid T)$. Information about inferred descriptions shapes text generation via the auxiliary decoding objective.

4.2 Latent State Inference for Prompting

Work on few-shot prompting consistently finds benefits from adding extra examples to prompts (Brown et al., 2020). As in §4.1, we produce extra examples for a seed prompt by finding situation descriptions S that are predictable from T and improve prediction of T' on unannotated examples. We may do so using a very similar procedure to the one in §4.1: now we choose prompts (but not

model parameters) to maximize:

$$\arg\max_{\hat{S}} \sum_{T,T' \in \mathcal{X}_U} p(\hat{S} \mid T)\, p(T' \mid T, S) \quad (6)$$

then add these newly annotated examples to the prompt (which we may do during both training and evaluation). Algorithmically, we iterate incrementally over unannotated examples $\mathcal{X}_A$:

1. E-step: set $\hat{S} \approx \arg\max_S p(S \mid T)\, p(T' \mid S)$ for each context-sentence pair (T, T') in $\mathcal{X}_U$ by prompting the LM with $[D \cdot \mathcal{P}_A \cdot T]$, then reranking the candidate states according to

$$p(S \mid [D \cdot \mathcal{P}_A \cdot T])\, p(T' \mid [D \cdot \mathcal{P}_A \cdot T \cdot S]) . \quad (7)$$

2. M-step: add $[T \cdot \hat{S} \cdot T']$ to $\mathcal{P}_A$ in Eq. (4).

Once all examples in $\mathcal{X}_U$ have been annotated and added to $\mathcal{P}_A$, we prompt with auxiliary supervision for each context in the evaluation set using $\mathcal{P} = [D \cdot \mathcal{P}_A \cdot T_{pred}]$.

5 Experimental Setup

Datasets We evaluate SITUATIONSUPERVISION on English language modeling datasets. **TW** is a set of 1368 transcripts (992 train / 376 evaluation) derived from TextWorld (Côté et al., 2018) We generate a set of textual game transcripts where players navigate through a house, unlocking doors and containers to hunt for a target object. The LM is trained on these transcripts to generate next *actions*. As state supervision, we use the set of state variables (given as entity-centric facts) that are *known* and *relevant* in the current context (see §7.1 for more details). **TRIP** (Storks et al., 2021) is a set of 1643 plausible and implausible five-sentence stories (1169 train / 474 evaluation) which require physical commonsense reasoning to disambiguate. Models are trained to generate judgments of whether or not a given sentence is acceptable in a context. The state is given by a set of attributes for each entity, which is updated after each sentence.[4]

Each passage $x \in \mathcal{X}$ comprises a sequence of chunks $T'_0, T'_1, \cdots, T'_n$. In TW, each chunk consists of a textual action description followed by a game response. In TRIP, each chunk is a single sentence from the story followed by a plausibility judgment. We test coherence of generating each T' from its context T. For the annotated passages in

$\mathcal{X}_A$, *each context* T_i is annotated with corresponding state information S_i. Thus, passages in $\mathcal{X}_U$ can be broken down into (T, T') pairs, while passages in $\mathcal{X}_A$ can be broken down into (T, S, T') triples.

Models For fine-tuning experiments, we use BART-base (Lewis et al., 2020) as the language model and fine-tune it using the AdamW optimizer with learning rate 1e-5, stopping once validation accuracy has stopped improving for 10 epochs. For prompting experiments, we use the GPT3 `da-vinci-002` model (Brown et al., 2020).[5]

Metrics To evaluate models on TW, we sample next actions from the LM and compute the fraction of these that are semantically **coherent** using the TW simulator.[6][7] For TRIP, we evaluate every story pair by training models to predict the string OK or Not OK after each sentence depending on whether it is semantically acceptable within a given context. The TRIP dataset contains human semantic acceptability judgments for each sentence of each stories; we evaluate the **accuracy** with which models predict these acceptability judgments (labeling a story as unacceptable if any sentence is predicted to be unacceptable).

For TW, we report *sentence-wise* metrics: we measure the fraction of next sentences which are generated to be coherent within the context. In TRIP, we report *passage-wise* metrics: we measure the percent of complete passages for which *every sentence of the passage* is labelled accurately.

As baselines in each domain, we compare to ordinary fine-tuning and prompting. As far as we are aware, no prior work in these domains focus on evaluating generation coherence or accuracy.[8]

---

[4] See Appendix A for state representation details.

[5] Further details can be found in Appendix C.

[6] The simulator also returns *game responses* after each action (e.g. *You entered the kitchen* in response to > *go west*). Game response coherence results can be found in Appendix B.1. Because coherence evaluation is less well-defined for game responses, we do not report results in the main paper.

[7] In Appendix B.2, we also evaluate generation diversity amongst these actions using recall against the full set of ground-truth possible next actions.

[8] For TRIP, most prior work uses the evaluation procedure from Storks et al. (2021), which are focused not on evaluating acceptabilities of incrementally-generated stories, but instead on post-hoc commonsense reasoning. For TW, prior work uses a much richer supervisory signal (environment reward) to select *optimal* actions towards a goal, rather than modeling the full set of plausible next actions (Ammanabrolu et al., 2021).

	$\|\mathcal{X}\|$	$\|\mathcal{X}_A\|$	Method	
				Coherence
TW	1k	0	Fine-tuning	$79.4\%_{\pm 2.4\%}$
	1k	500	SITSUP	$80.5\%_{\pm 1.8\%}$
	1k	500	SITSUP + Latent	$83.4\%_{\pm 1.4\%}$
	1k	1k	SITSUP	$81.5\%_{\pm 1.5\%}$
	10k	0	Fine-tuning	$83.6\%_{\pm 2.5\%}$
				Accuracy
TRIP	1k	0	Fine-tuning	$36.5\%_{\pm 3.5\%}$
	1k	500	SITSUP	$43.6\%_{\pm 1.0\%}$
	1k	500	SITSUP + Latent	$43.6\%_{\pm 1.0\%}*$
	1k	1k	SITSUP	$43.0\%_{\pm 1.7\%}$

Table 1: BART fine-tuning results on TW and TRIP, where $\|\mathcal{X}\|$ is the number of training examples, and $\|\mathcal{X}_A\|$ is the total amount of state supervision. We report results for standard LM training, SITUATIONSUPERVISION with only auxiliary situation modeling, and SITUATIONSUPERVISION with both auxiliary modeling and the latent state inference. The table shows and standard errors over 8 random seeds. Training with any state supervision helps over training with no state supervision. With a comparable amount of state supervision, latent inference sometimes gives further improvements. *Latent inference unable to improve beyond base auxiliary situation modeling checkpoint

6 Experiments

6.1 Fine-Tuning

Our experiments use 1000 training examples, varying the fraction of these examples for which we provide state annotations ($\|\mathcal{X}_A\| = \{0, 500, 1000\}$). For each choice of $\|\mathcal{X}_A\|$, we repeat experiments across 8 random seeds, training on a different set of 1000 examples for each seed. We compare models trained using ordinary language modeling techniques, Eq. (3), and Eq. (5). We evaluate using metrics described in §5.

Results Evaluation results are shown in Table 1. In TW, using auxiliary supervision and latent state inference, SITUATIONSUPERVISION with 500 state annotations improves generation coherence by $\sim 4\%$ over a text-only baseline, giving comparable improvements to training on 9,000 more text-only examples. Results in Appendix B.2 show that these improvements come at no cost to generation *diversity*. In fact, the latent procedure with 500 seed states is able to outperform full auxiliary supervision — possibly because latent state inference is able *automatically* discover usable state representations, which are more useful for prediction than human-authored ones. In TRIP, SITUATIONSUPER-

	$\|\mathcal{X}\|$	$\|\mathcal{X}_A\|$	Method	
				Coherence
TW	25	0	Text prompting	67.4%
	25	12	SITSUP	68.5%
	25	12	SITSUP + Latent	75.6%
	25	25	SITSUP	73.9%
				Accuracy
TRIP	80	0	Text prompting	59.5%
	80	20	SITSUP	58.2%
	80	20	SITSUP + Latent	67.1%
	80	80	SITSUP	70.7%

Table 2: GPT3 prompting results on TW and TRIP, using text-only querying, SITUATIONSUPERVISION with only the auxiliary situation modeling component, and SITUATIONSUPERVISION with both the auxiliary situation modeling and the latent situation prediction components. Prompting with any state supervision helps over prompting with no state supervision. With a comparable amount of ground-truth state supervision, latent inference significantly improves over only auxiliary situation modeling.

VISION with 500 seed states improves accuracy by $\sim 7\%$ over a text-only baseline. Note in this case that the latent inference procedure was unable to improve beyond auxiliary training. However, even adding in the remaining 500 ground-truth state annotations does not improve the LM, indicating that perhaps the 500 seed states were sufficient for the LM to learn everything it can from state supervision.

6.2 Prompting

In TW, we used 25 sentences (3 stories) in $\mathcal{P}$. In TRIP, we used 80 sentences (16 stories) in $\mathcal{P}$. When evaluating latent supervision, we held out state annotations on 13 sentences (2 stories) in TW, and 60 sentences (12 stories) in TRIP. We run each prompting experiment once.

Results Results are reported in Table 2. Using SITUATIONSUPERVISION with auxiliary situation modeling where all passages are fully annotated with state (rows 4,8) dramatically improves performance compared to a text-only baseline (rows 1,5) in both domains. In TW, we see a $\sim 6.5\%$ improvement in generation coherence,[9] while in TRIP, we see a $\sim 11\%$ improvement in accuracy of coherence judgments.

Next, we examine the setting where certain state annotations are missing from the prompt, compar-

---

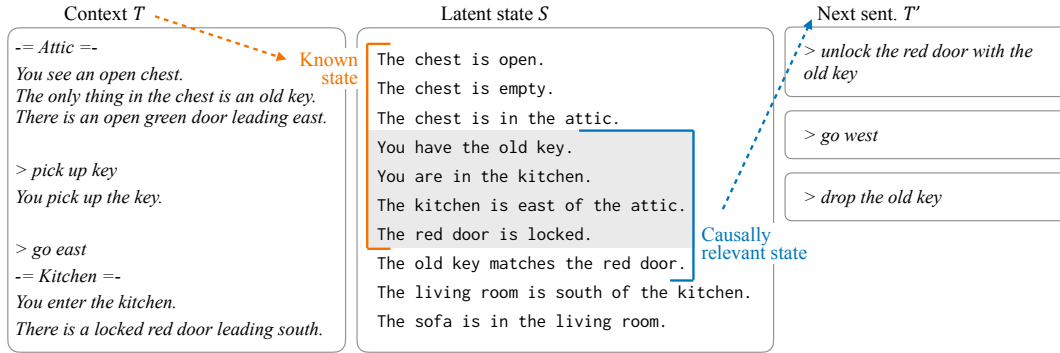[9]Results in Appendix B.2 shows that SITUATIONSUPERVISION also improves generation *diversity*.

Figure 3: Different choices of situation representation. We find that ideal representations consist of the intersection between *known state* and *causally relevant state* (highlighted in gray). The *known state* consists of all facts deducible from the prior context T (e.g. in TW, only facts about rooms or objects that the player has seen). The *causally relevant state* consists of all facts causally relevant to any plausible next sentence T' (e.g. in TW, only facts about the currently accessible items, here the old key but not the chest).

ing SITUATIONSUPERVISION with latent situation prediction (rows 3, 7) against SITUATIONSUPERVISION with only auxiliary situation modeling (rows 2, 6). We find that incorporating generated latent states into the prompt helps performance on both TW and TRIP, by 7.1% and 8.9% respectively.

7 Analysis

7.1 Choice of state is important

In this section, we explore the consequences of including/excluding various components of the state.

TW We begin by conducting experiments in TW. Because it is procedurally generated, the TW environment is able to provide detailed ground-truth state annotations for every entity that might be mentioned in text. All experiments described in §6 use situation representations that include only a subset of entities and properties: namely (1) only those that are already *known* (i.e. those have been asserted in the context), and (2) only those that are *causally relevant* (i.e. those that, if changed, would induce a different distribution over next sentences). See Fig. 3 for examples.

We train with auxiliary supervision using the three different choices of state: the full state, the known state (facts that satisfy condition (1)), and the relevant known state (facts that satisfy both conditions (1) and (2)). Results are shown in Table 3. We find that the training with the full state is not significantly better than simply training on text only, and perhaps slightly worse. Training on the subset of known facts outperforms training with the full state, and training on the intersection of known state and causally relevant state is even better.

State Type	Coherence
None	$79.4\%_{\pm 2.4\%}$
Full state	$78.0\%_{\pm 1.7\%}$
Full Known state	$79.7\%_{\pm 1.5\%}$
Relevant Known state	$81.5\%_{\pm 1.5\%}$

Table 3: Using different subsets of the state as auxiliary supervision for TW fine-tuning yields varying amounts of coherence improvements. We report averages and standard errors over 4 random seeds. Design of situation representations is consequential: using only the known and causally relevant portions of the state (relevant known state) substantially outperforms using the full state.

TRIP Using the principles deduced from the previous experiments in TW (the optimal state should be both *known* from prior context and *causally relevant* to the next sentence), we optimize the design of TRIP state annotations.[10] We used these state annotations for all experiments described above. In this section, we demonstrate that this outperforms using the original annotations provided in the dataset. Specifically, we sample 12 training examples to include in the prompt,[11] and compare text-only prompting against SITUATIONSUPERVISION with the original states (Orig) and SITUATIONSUPERVISION with handcrafted states (Ours). Results are reported in Table 4. By using our handcrafted states, we were able to achieve a much higher accuracy than using the original states.

---

[10]See Appendix A for details.

[11]In previous sections we used 16 samples. However, because the original states were much longer than our states, we were only able to fit 12 candidates in context using the original state annotations.

12563

	$\|\mathcal{X}\|$	$\|\mathcal{X}_A\|$	State Type	Accuracy
	12	0	-	59.3%
TRIP	12	12	Orig	62.8%
	12	12	Ours	68.1%

Table 4: Using different types of state annotations for TRIP when prompting GPT3 yields various amounts of performance improvements. We compare SITUATION-SUPERVISION using TRIP's original state annotations (Orig) against SITUATIONSUPERVISION using our own handcrafted state annotations (Ours). Note that using the original state annotation is only able to improve 3.6% over a text-only baseline, while using our state annotations improves 8.8%.

	TW	TRIP
SITUATIONSUPERVISION	75.6%	67.1%
without state reranking	72.4%	65.8%

Table 5: Ablating state reranking with $p(T' \mid S)$ when inferring the optimal latent state for prompting. In both TW and TRIP, SITUATIONSUPERVISION works better when we sample multiple states from $p(S \mid T)$ and rerank according to $p(T' \mid S)$, than when we simply take the greedy optimal state from $p(S \mid T)$.

7.2 Explicit state inference outperforms greedy next state prediction

A simplification of our latent state inference procedure for prompting simply asks GPT3 to greedily generate the most likely state according to prior context (i.e., $\arg\max_S p(S \mid T)$), without considering $p(T' \mid S)$ (as in chain-of-thought approaches; Wei et al., 2022). We compare our currently latent state procedure against this greedy state generation baseline in Table 5. We find that it indeed helps to consider $p(T' \mid S)$ when generating states, improving next sentence coherence by 3.2% in TW and next sentence accuracy by 1.3% in TRIP.

7.3 For a fixed context window budget, including more state annotations outperforms including more text samples

Because the limiting factor in many current few-shot prompting methods is context window size rather than annotation effort, we study whether it is more token-efficient to include additional state annotations or additional text examples in the prompt. We compute the number of tokens in prompts annotated with state ($\mathcal{P}_A$), then formulate a text-only prompt ($\mathcal{P}_T$) by stripping the state annotations from $\mathcal{P}_A$, then appending randomly-selected text-only samples from the remaining training data until the

	# toks	$\|\mathcal{X}\|$	$\|\mathcal{X}_A\|$	Metric
				Coherence
TW	3199	54	0	56.7%*
TW	3199	25	25	65.0%*
				Accuracy
TRIP	3053	229	0	60.5%
TRIP	3054	80	80	70.7%

Table 6: When prompting with limited context-window size, supplementing existing prompt demonstrations with states is more token-efficient than providing more text-only training examples.
*Coherences of *greedy* next generations are reported in this experiment for TW.

number of tokens in the new prompt is equal (or nearly equal) to the number of tokens in $\mathcal{P}_A$.

We prompt the LM using either text-only prompting conditioned on $\mathcal{P}_T$, or auxiliary prompting conditioned on $\mathcal{P}_A$. The results are shown in Table 6. (Due to limitations in annotation budget, for TW in this experiment, we report coherence of the greedily-generated next actions rather than sampling 5 actions.) We see that under a fixed context token budget, in both domains, it is more helpful to supplement existing examples with their state annotations rather than insert additional text-only examples into the context window.

8   Conclusion

Effective generation of coherent text requires reasoning about the world that text describes. In this work, we use entity states as auxiliary supervision to improve LMs ability to perform this reasoning under both fine-tuning and prompting. We find that when either annotation budget (for fine-tuning) or context window size (for prompting) are limited, it is more sample- and token-efficient to increase the amount of state supervision rather than text-only supervision. However, since state annotations are harder to collect, we introduce latent supervision algorithms for *sample-efficiently* improving LM generation coherence, and demonstrate improvements in two domains. Our results point to a potentially broad role for semantic supervision in LM training and prompting—even small amounts can yield large coherence improvements. This work more generally suggests that semantic state reasoning is still challenging for even modern large language models, and but can be improved without fundamental changes to the architecture of existing LMs.

9 Limitations

The main limitation of SITUATIONSUPERVISION is that situation annotations can often be expensive to curate and difficult to design (though we outline some general principles for their design in §7). Furthermore, we conducted experiments on only two datasets in this paper. Future work could explore a wider genre of texts, more domains, and more languages.

10 Impact Statement

This work introduces ways of using state supervision for improving the coherence of language model generations. This can be used to reduce the incidence of false or misleading generations from language models. Furthermore, we found that we can bootstrap starting from small amounts of seed state supervision to achieve large coherence gains, meaning the method can be used with relative ease without the need for extensive annotation. However, the methods described in this paper can also be used maliciously to improve the coherence of automatically-generated misinformation, hate speech, or other harmful content.

References

Prithviraj Ammanabrolu, Jack Urbanek, Margaret Li, Arthur Szlam, Tim Rocktäschel, and Jason Weston. 2021. How to motivate your dragon: Teaching goal-driven agents to speak and act in fantasy worlds. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 807–833, Online. Association for Computational Linguistics.

Jon Barwise and John Perry. 1981. Situations and attitudes. *The Journal of Philosophy*, 78(11):668–691.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. 2018. e-snli: Natural language inference with natural language explanations. *Advances in Neural Information Processing Systems*, 31.

Marc-Alexandre Côté, Ákos Kádár, Xingdi (Eric) Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2018. Textworld: A learning environment for text-based games. In *Computer Games Workshop at ICML/IJCAI 2018*, pages 1–29.

Aditya Gupta and Greg Durrett. 2019. Tracking discrete and continuous entity state for process understanding. *arXiv preprint arXiv:1904.03518*.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2016. Tracking the world state with recurrent entity networks. *arXiv preprint arXiv:1612.03969*.

Angelika Kratzer. 2007. Situations in natural language semantics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Belinda Z. Li, Maxwell Nye, and Jacob Andreas. 2021. Implicit representations of meaning in neural language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1813–1827, Online. Association for Computational Linguistics.

Ruibo Liu, Jason Wei, Shixiang Shane Gu, Te-Yen Wu, Soroush Vosoughi, Claire Cui, Denny Zhou, and Andrew M Dai. 2022. Mind's eye: Grounded language model reasoning through simulation. *arXiv preprint arXiv:2210.05359*.

Gary Marcus and Ernest Davis. 2020. Gpt-3, bloviator: Openai's language generator has no idea what it's talking about. [Online; posted 22-August-2020].

Marianna Martindale, Marine Carpuat, Kevin Duh, and Paul McNamee. 2019. Identifying fluently inadequate output in neural and statistical machine translation. In *Proceedings of Machine Translation Summit XVII: Research Track*, pages 233–243, Dublin, Ireland. European Association for Machine Translation.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

Bhavana Dalvi Mishra, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension. *arXiv preprint arXiv:1805.06975*.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. 2021. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*.

Yasumasa Onoe, Michael JQ Zhang, Eunsol Choi, and Greg Durrett. 2021. Creak: A dataset for commonsense reasoning over entity knowledge. *arXiv preprint arXiv:2109.01653*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Curran Associates Inc., Red Hook, NY, USA.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Kyle Richardson, Ronen Tamari, Oren Sultan, Reut Tsarfaty, Dafna Shahaf, and Ashish Sabharwal. 2022. Breakpoint transformers for modeling and tracking intermediate beliefs.

Pratyusha Sharma, Antonio Torralba, and Jacob Andreas. 2021. Skill induction and planning with latent language. *arXiv preprint arXiv:2110.01517*.

Shane Storks, Qiaozi Gao, Yichi Zhang, and Joyce Chai. 2021. Tiered reasoning for intuitive physics: Toward verifiable commonsense language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4902–4918, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. 2022. Black-box tuning for language-model-as-a-service. *arXiv preprint arXiv:2201.03514*.

Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A Smith. 2018. Syntactic scaffolds for semantic structures. *arXiv preprint arXiv:1808.10485*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Ethan Wilcox, Peng Qian, Richard Futrell, Miguel Ballesteros, and Roger Levy. 2019. Structural supervision improves learning of non-local grammatical dependencies. *arXiv preprint arXiv:1903.00943*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Eric Zelikman, Yuhuai Wu, and Noah D Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *arXiv preprint arXiv:2203.14465*.

Rowan Zellers, Ari Holtzman, Matthew Peters, Roozbeh Mottaghi, Aniruddha Kembhavi, Ali Farhadi, and Yejin Choi. 2021. PIGLeT: Language grounding through neuro-symbolic interaction in a 3D world. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2040–2050, Online. Association for Computational Linguistics.

Chunting Zhou, Graham Neubig, Jiatao Gu, Mona Diab, Francisco Guzmán, Luke Zettlemoyer, and Marjan Ghazvininejad. 2021. Detecting hallucinated content in conditional neural sequence generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1393–1404, Online. Association for Computational Linguistics.

Rolf A Zwaan and Diane Pecher. 2012. Revisiting mental simulation in language comprehension: Six replication attempts. *PloS one*, 7(12):e51382.

A Constructing the State

In each domain, the state is a collection of facts (attributes and/or relations) about each entity. It is updated each time there is a new action, instruction, or sentence. We convert the state to natural language to take advantage of existing linguistic understanding in pre-trained models. Future work can examine the effect of using non-natural-language forms of state.

Below, we discuss the details of this conversion from the available state annotations in each domains.

TW In TW, the simulator gives us the **full state**, or the full set of facts describing the state of the world after executing each agent action. Facts are either entity properties (e.g. `locked(door)`), or relations between two entities (e.g. `is-in(key, chest)`). However, since the agent has not explored the full state at the start of each game, at each step, we compute a subset of the facts that the agent *knows about*. We call this the **known state**. We further restrict this subset to only facts that are *causally relevant* to any possible next action that the agent can take, such that all possible next actions can be inferred from just this set. We call this the **relevant known state**.

We compute both these sets heuristically: the known state consists of all facts about any currently or previously accessible entities that the agent has encountered. For the *relevant* known state, we discard facts about previously accessible entities and keep only facts about currently accessible entities. Specifically, the relevant known state consists of facts about: 1. player location, 2. all currently accessible items (i.e. in the current room or in the inventory), 3. which doorways are accessible from the current room and/or which rooms neighbor the current room.

We convert collections of facts to natural language following the same procedure as Li et al. (2021). Specifically, propositions $p(o)$ are converted to "*the {o} is {p}*", while relations $r(o_1, o_2)$ are converted to "*the {o_1} is {r} {o_2}*".

TRIP In TRIP, we write out seed states for 16 stories, consisting of facts known to hold true after each sentence of the story — then use GPT3 to automatically infer states for the remaining stories in the training data. We aim to construct the state in TRIP to capture the spirit of the **relevant known state** in TW (which we know from §7.1 to

be the optimal state supervision), whereby we only include facts both known from the prior context and potentially causally relevant to the next sentence. However, though capturing known facts is straightforward, because TRIP is a real dataset consisting of open-ended text, the set of plausible next generations is open-ended, meaning that the full set of causally relevant known facts cannot be always be anticipated ahead of time. Instead, we use the ground-truth acceptable completion as a minimal guarantee – we aim to include facts informative for generating at least the single ground-truth next sentence in the acceptable story (which isn't always straightforwardly derived from the known facts). One example is as follows:

- T = *Tom packed his gloves in his suitcase. Tom checked his suitcase in at the airport.*

- S = `Tom's gloves are in the suitcase. The suitcase is checked in at the airport. Tom does not have his suitcase. Tom does not have his gloves.`

- T' = *Tom boarded the plane without his gloves.*

Note that while *Tom does not have his gloves* is technically inferrable from `Tom's gloves are in the suitcase. The suitcase is checked in at the airport`, including this fact explicitly in S reinforces the causal link between the next sentence T' and S.

For the analysis in §7.1, we compare against a stringified version of the originally-provided states. In the original dataset, each sentence of a story is annotated with the state changes applied to each of the (up to 15) attributes of that entity. The state annotations take the form of (*entity, attribute, value*) triples. Each entity attribute is associated with a value indicating the direction of change for that attribute. For example, (*shirt, cleanliness, true → false*) indicates *the shirt became dirty*.

Because there are a finite set of (15) attributes and (8) values, we enumerate rules for converting all (*attribute, value*) pairs to natural language predicates VP. We then convert (*entity, attribute, value*) triples into "*the {entity}* VP".

$|\mathcal{X}|$	$|\mathcal{X}_A|$	Method	Coherence
1k	0	Fine-tuning	$40.0\%_{\pm 0.7\%}$
1k	500	SITSUP	$40.0\%_{\pm 0.8\%}$
1k	500	SITSUP + Latent	$40.0\%_{\pm 0.4\%}$
1k	1k	SITSUP	$42.9\%_{\pm 1.0\%}$

Table 7: TW game response generation coherence. We evaluate BART fine-tuning with and without components of SITUATIONSUPERVISION.

B Further TW Evaluations

B.1 Game Response Coherence for BART Fine-Tuning

The text of TW consists of alternating *actions* and *game responses*. For example:

```
> open door
You open the door
> go west
-= Kitchen =-
You arrive at a kitchen. You see a counter.
On the counter is an old key. [...]
```

In this example, lines starting with > are actions and all other lines are game responses.

In §6, we only evaluated coherence of generating *actions* in TW. Here, we evaluate coherence of generating *game responses* as well. Due to quota restrictions, we evaluate game responses only for fine-tuning approaches and not prompting approaches.

Table 7 reports coherence results for game responses alone, and game responses and actions combined. Unlike the set of acceptable actions, the TW simulator does not provide us with a set of acceptable game responses. Instead, we can only compare the ground-truth game response from the simulator. This can result in over-penalization: when pieces of the underlying state are still unknown, the LM will be falsely penalized, despite generating a game response coherent with the prior context. Thus the numbers reported in Table 7 are simply a lower bound.

B.2 Generation Diversity

To measure the diversity of LM outputs, we use *recall*[12] between the set of LM generations and the full set of ground-truth valid sentences. This latter set is provided to us by the TextWorld simulator. Note that this set is not entirely complete, as there will be generations that are consistent with

---

[12]Because we sample at most 5 unique generations from the LM, there is a hard ceiling on maximum achievable "recall" in our case.

Model	$|\mathcal{X}|$	$|\mathcal{X}_A|$	Method	Recall
BART	1k	0	Fine-tuning	$11.8\%_{\pm 0.3\%}$
	1k	500	SITSUP	$11.8\%_{\pm 0.3\%}$
	1k	500	SITSUP + Latent	$11.9\%_{\pm 0.2\%}$
	1k	1k	SITSUP	$12.6\%_{\pm 0.4\%}$
GPT3	25	0	Text prompting	33.3%
	25	12	SITSUP	40.1%
	25	12	SITSUP + Latent	42.1%
	25	25	SITSUP	40.9%

Table 8: TW generation diversity for fine-tuning and prompting with and without components of SITUATION-SUPERVISION. We see that using SITUATIONSUPERVISION with fine-tuning does not harm the diversity of generated samples, while using SITUATIONSUPERVISION with prompting actually increases diversity.

the *known facts* from the prior context but contradict an *unknown fact*, and is consequently not accepted by the simulator. However, recall against the simulator-provided set of valid sentences remains a good heuristic for diversity.

We examine how training with SITUATION-SUPERVISION affects generation diversity. We use the same models and training/prompting setups as in §6 and evaluate the diversity among the generated samples. Results are shown in Table 8. We showed in §6 that SITUATIONSUPERVISION improves TW generation coherence in both the fine-tuning and prompting cases. As shown in Table 8, SITUATIONSUPERVISION does not sacrifices diversity to achieve those coherence gains. In fact, prompting with SITUATIONSUPERVISION *improves* diversity when compared against a text-only model, and doing latent inference appears to additionally improve diversity beyond simply auxiliary situation modeling.

C Infrastructure and Reproducibility

We ran all fine-tuning experiments on a single 32GB NVIDIA Tesla V100 GPU. We use a BART-base model which has 6 Transformer layers each in its encoder and decoder, and 139M total parameters. Training time varies depending on domain and data size, but generally is not longer than a few hours. As a reference point: on 1000 TW examples, training takes ~1 hour for text-only training, ~1-2 hours for training with auxiliary state supervision, and ~1-3 hours for training with latent state supervision. For prompting results, we use OpenAI's GPT3 text-davinci-002 model. For sampling next actions in TW, we use a generation temperature of 0.7. When judging acceptability of each

sentence in TRIP, we directly compare $p(\text{Not OK})$ against $p(\text{OK})$. When sampling states for latent state inference, to encourage diversity, we use a generation temperature of 0.9.

We used PyTorch (Paszke et al., 2019) and Huggingface Transformers (Wolf et al., 2020) for implementing and training BART-base models. We use OpenAI's API[13] for querying GPT3.

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 9

☑ A2. Did you discuss any potential risks of your work?
Section 10

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract, Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 5

☑ B1. Did you cite the creators of artifacts you used?
Section 5

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Could not find license

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
All data was intended for research.

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
None of the data used should contain identifiable or offensive information.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 5

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 5

C ☑ Did you run computational experiments?

Section 6,7

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Appendix D

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 5, Appendix D

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 6

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Appendix D

D ☒ **Did you use human annotators (e.g., crowdworkers) or research with human participants?**

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.