Improving Coherence of Language Model Generation with Latent Semantic State

Anonymous ACL submission

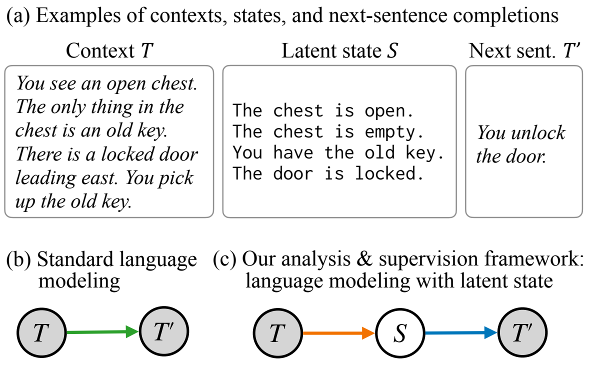
Abstract

Sentences generated by neural language models (LMs) often suffer from coherence errors: they describe events and situations inconsistent with the *state of the world* described by preceding text. We show that coherence errors can arise at multiple stages of LM computation, and describe a procedure for distinguishing errors in *inferring state* from errors in *generating sentences*. In models with correctable errors of the first type, we show that targeted supervision can address them. We introduce two procedures for using explicit representations of world state as auxiliary supervision. These procedures efficiently improve LM coherence, in some cases providing the benefits of 1,000–9,000 training examples with only 500 state annotations.

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 suffer from **failures of semantic coherence**. Samples from LMs refer to entities that have not yet been mentioned, present contradictory facts, or describe impossible events (Marcus and Davis, 2020).

This paper introduces a framework for understanding and mitigating incoherent language generation. First, **we identify two distinct sources of coherence errors in LMs**: failures to *represent the state of the world* described by a discourse, and failures to *identify next sentences* licensed by a given state. In LMs with coherence errors, we describe a procedure for quantifying how much implicit state representations can be improved, and how much improvement would affect the coherence of downstream text. Second, we show that in LMs with improvable state representations, **small amounts of state supervision can improve coherence of generated text**. Given a small set of annotations describing the states of mentioned entities in training documents (Fig. 1a), we describe how to impute


(a) Examples of contexts, states, and next-sentence completions
Context T
Latent state S
Next sent. T'
(b) Standard language modeling
(c) Our analysis & supervision framework: language modeling with latent state
T
T'
T
S
T'

You see an open chest. The only thing in the chest is an old key. There is a locked door leading east. You pick up the old key.

```
The chest is open.
The chest is empty.
You have the old key.
The door is locked.
```

You unlock the door.

Figure 1: Language modeling with latent state. (a) Generating a sentence T' in a context T requires first inferring the state of the world S described by T. (b) Standard neural LMs do not represent world state explicitly, but their representations may encode it implicitly. (c) By formulating language modeling as an explicit latent variable problem, we can identify and correct generation errors in neural LMs.

latent states for remaining examples, then use these states as auxiliary supervision during LM training. In experiments on the TextWorld dataset, applying this procedure with 500 seed annotations gives coherence improvements comparable to 1,000–9,000 additional training sentences. Our results show that errors in LM generation can, in some cases, be linked directly to LMs' internal representations and corrected with simple supervision.

2 Preliminaries

A **language model** (LM) encodes a distribution $p(T' \mid T)$ over texts T' given contexts T (Fig. 1b). Today, LMs are primarily implemented as deep neural networks trained on massive text corpora (Brown et al., 2020). LM generation is prone to several failure modes: samples may be incoherent, untruthful, or unreliable (Zhou et al., 2021; Maynez et al., 2020; Martindale et al., 2019), propagate social biases (Abid et al., 2021), or contain hateful content (Askell et al., 2021).

Incoherent LM output, the focus of this paper, contradicts known facts from prior context, refers to non-existent entities, or describes impossible events or actions. Existing work addresses coherence issues by introducing model architectures that facilitate global planning (Zhai et al., 2019; Fan et al., 2018; Kiddon et al., 2016), state tracking (Zellers et al., 2021; Storks et al., 2021), or use external knowledge sources (Lewis et al., 2020b; Shuster et al., 2021); however, these errors remain a persistent feature of standard LMs.

How might we represent and reason formally about coherence errors? One idealization, drawn from the linguistic theory of dynamic semantics, states that the process of generating a next sentence T in a context T' involves first inferring the *state of the world* S described by T, then identifying utterances that could come next given that S is true (Fig. 1c; Heim, 2012). In NLP, recent work has found that pre-trained LMs implement a version of this inference procedure: encoders build representations of mentioned entities and their properties, and these representations are causally linked to decoder output (Li et al., 2021). We investigate whether *errors* in state representations contribute to errors in generation (§3), and whether improving them can improve the coherence of LM output (§4).

3 Identifying improvable state representations

In the idealized generative process of Fig. 1, coherence errors can arise in two places: in $p(S \mid T)$ (**state inference errors**) and in $p(T' \mid S)$ (**sentence prediction errors**). Li et al. (2021) found that LMs approximate this generative process by using *context encodings* to represent the state of the world. We thus expect state inference errors to manifest as *encodings from which state representations cannot be extracted*, and prediction errors to manifest as *failure to produce coherent text given ground-truth state representations*. This section presents an experimental framework for measuring whether each error type is present and correctable in an LM.

3.1 Models and Datasets

We study three English language modeling datasets. **TW** is 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* and *game responses*. As state supervision, we

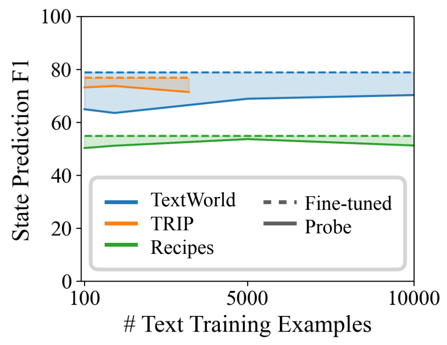


Figure 2: Accuracy of states extracted from LM encoders. Solid lines show accuracy of a state *probe* applied to fixed encodings from LMs fine-tuned for text generation with varying amounts of data. Dashed lines show performance of a state prediction *upper bound* in which the encoder and decoder were jointly fine-tuned to predict state representations. Shaded areas represent the possible improvement in *state prediction* that could be obtained by directly supervising state representations, which is large for TW but smaller in other domains.

use the set of state variables (given as entity-centric facts) that are *verifiable* and *relevant* in the current context.[1] **TRIP** (Storks et al., 2021) features pairs of plausible and implausible short stories which require physical commonsense reasoning to disambiguate. Models are trained to predict plausible next sentences given histories. The state is given by a set of attributes for each entity, which is updated after each sentence. **Recipes** (Kiddon et al., 2016) consists of cooking instructions annotated with state changes and affected ingredients at each step. Models are trained to generate next steps.

For all experiments, we use BART-base (Lewis et al., 2020a) as the language model and fine-tune it on the dataset being evaluated. The BART model represents the context with a transformer encoder, then generates a next sentence using a transformer decoder. Fine-tuning uses the AdamW optimizer with learning rate 1e-5 and patience 5–15.

3.2 Measuring state inference errors

Method We evaluate the accuracy of (1) models trained to predict states given fixed LM encodings, and (2) upper bound models trained to predict states from context text. The *magnitude* of (1) measures how well LMs encode state, while the *difference* between (2) and (1) measures how much state representations could be improved with additional supervision. The upper bound is obtained by jointly fine-tuning the encoder and decoder of an LM; the probe is obtained by fine-tuning a decoder only.

[1]See Appendix B.1 for state representation details.

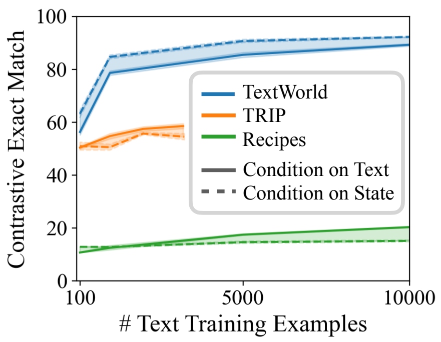


Figure 3: Next sentence prediction, conditioned on text (solid line) vs. state (dashed line), across 100-10k text-to-text or state-to-text examples. In TW, ground-truth states improve next-sentence prediction. In TRIP and OpenPI, models conditioned on state perform *worse*.

Results Results are shown in Fig. 2. In TW, we find a large gap between the upper bound and the probe, indicating that states are not well-encoded by models trained on text alone, but learnable with targeted supervision. In TRIP and Recipes, the gap is much smaller. In all domains, *simply scaling up the amount of text used to fine-tune the LM does not yield improved state representations.*

3.3 Measuring sentence prediction errors

Method We next evaluate whether, in each problem domain, LMs with access to ground-truth state representations can generate more coherent output than LMs that condition on text representations. We train (1) an $S \to T'$ LM to map from explicit state annotations to next sentences, and (2) a base $T \to T'$ LM. The *magnitude* of (1) measures the prevalence of text prediction errors given an ideal state, while the *difference* between (2) and (1) measures how much generation would in principle be improved by better state representations.

Evaluation We use a contrastive evaluation. For each test context T or S, models are presented with the true next sentence following the context and (one or more) distractor sentences. We measure the fraction of contexts in which the true next sentence is assigned greater probability than the distractor.

In TW, we select six distractors that *cannot be produced* by the simulator in a given context. In TRIP, we select a single distractor taken from the implausible story in each pair. Finally, in Recipes, we use 15 in-batch next sentences as distractors.

Results Results are shown in Fig. 3. In TRIP and Recipes, even ground-truth state representations do not promote more coherent text generation. In TW, this is reversed, and LMs achieve better performance when conditioning on state.

To summarize: coherence errors in language modeling problems reflect a diverse set of underlying model failures. In TRIP, states are correctly inferred most of the time, but difficult to use; generation errors arise in next sentence prediction. In Recipes, states are not correctly inferred and not easily correctable, and also seemingly unhelpful for generation. In TW, states are incorrectly inferred but inferrable in principle, and useful if inferred.

4 Improving LM coherence

§3 predicts that language models for TW (but not other domains) could be improved with better state representations. We next describe a procedure for supervising state representations directly, and show that it leads to improved coherence of LM outputs.

We assume access to a large set of text-only examples $\mathcal{X}$ (consisting of (T, T') pairs), of which only a small subset $\mathcal{X}_A \subset \mathcal{X}$ are annotated with state information (and consist of (T, S, T') triples). The remaining $\mathcal{X}_U = \mathcal{X} \setminus \mathcal{X}_A$ are unaligned.

Method 1: Auxiliary Supervision (AS) The baseline BART LM comprises an encoder $\mathcal{E}$ and a decoder $\mathcal{D}$. $\mathcal{D}(\mathcal{E}(T))$ outputs a distribution over next sentences. $\mathcal{E}$ and $\mathcal{D}$ are trained 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 BART decoder $\mathcal{D}_{S|T}$ (distinct from the text-to-text LM's decoder) which is trained to predict S from encodings of the text history $\mathcal{E}(T)$. We define

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

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

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

We first fine-tune BART-base to convergence on $\mathcal{X}$ using $\mathcal{L}_{T'|T}$, then train on Eq. (3) above.

Method 2: Latent Supervision (LS) Even for unannotated examples $\mathcal{X}_U$, it may be possible to *infer* state annotations at training time. States are in general easier to infer at training time (when both

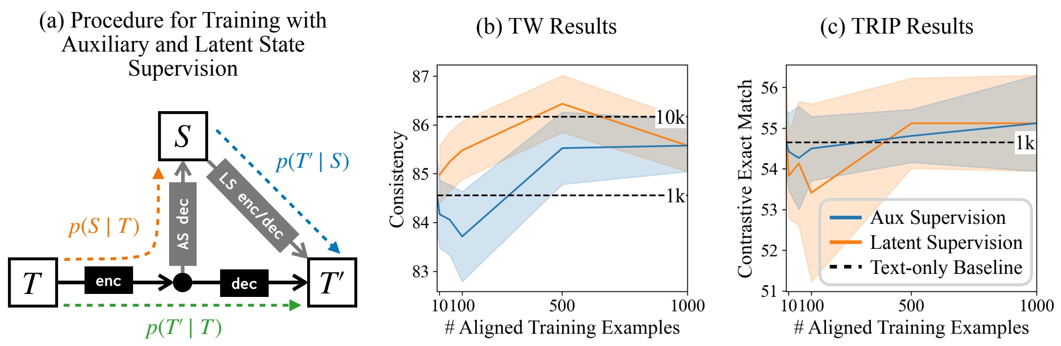


Figure 4: Training with auxiliary and latent state supervision. (a) depicts our training procedure: we train an LM with an auxiliary decoder to predict the state from the context, using an additional encoder/decoder to semi-supervisedly infer latent state. We run experiments in TW (b) and TRIP (c), assuming access to 1000 total examples and varying how many of them are aligned with state (from 10-1000). We benchmark generation coherence against LMs trained on 1k and 10k text-only examples. In TW, state supervision improves generation coherence, sometimes beyond a model trained on 10k text-only examples. As predicted, state supervision in TRIP is unhelpful.

T and T' are available) than at test time (when only T is observed). AS may thus be extended to incorporate *latent* supervision, in which state variables are simultaneously inferred and used as auxiliary supervision for a model that predicts $p(T' \mid T)$. We do so with the objective:

$$\arg\max_{\Theta,\hat{S}} \sum_{\substack{T,T'\\ \in\mathcal{X}}} \mathcal{L}(T'|T) + \sum_{\substack{T,T',S\\ \in\mathcal{X}_A}} \left(\mathcal{L}(S|T) + \mathcal{L}(T'|S,T)\right) + \sum_{\substack{T,T',\hat{S}\\ \in\mathcal{X}_U}} \left(\mathcal{L}(\hat{S}|T) + \mathcal{L}(T'|\hat{S},T)\right), \qquad (4)$$

Eq. (4) extends AS by concurrently training an encoder-decoder $\mathcal{M}_{T'|S,T}$ to model $p(T'|S,T)$. The full set of parameters is $\Theta = \{\theta_{\mathcal{E}}, \theta_{\mathcal{D}}, \theta_{\mathcal{D}_{S|T}}, \theta_{\mathcal{M}_{T'|S,T}}\}$, optimized via alternating coordinate ascent on Θ and $\hat{S}$. We initialize $\theta_{\mathcal{E}}, \theta_{\mathcal{D}}, \theta_{\mathcal{D}_{S|T}}$ using AS, and $\theta_{T'|S}$ by fine-tuning to convergence on $\mathcal{X}_A$. We then iterate between:

1. Set $\hat{S} \approx \arg\max_S p(S \mid T)p(T' \mid S)$ for $\mathcal{X}_U$ by sampling five state candidates from $p(S \mid T)$, then reranking these candidates according to $p(S \mid T)p(T' \mid S)$.

2. Using the new $\hat{S}$, train Θ to maximize Eq. (4). Rather than training to convergence, we perform SGD on Eq. (4) for five epochs.

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

Evaluation We select 1000 training examples ($|\mathcal{X}| = 1000$) and experiment with varying amounts of alignment ($|\mathcal{X}_A| = \{10, 100, 500, 1000\}$). For each size of $\mathcal{X}_A$, we randomly create 8 different training sets (both $\mathcal{X}$ and $\mathcal{X}_A$ can vary among the 8) and train LMs using Eq. (3) and Eq. (4). For each context in the TW test set, we draw five samples from the LM and measure the fraction of these that are semantically coherent by comparing them to possible outputs of the TW simulator. In TRIP, we use the contrastive evaluation from §3.3.

Results We report results in TW (Fig. 4b), and TRIP (Fig. 4c). We report means and standard errors across the 8 training sets. In TW, 10-50 state annotations suffice to outperform the text-only baseline, and with 500 state annotations, LS gives comparable coherence improvements to training on 9,000 more text-only examples. (Note, though, that there is high variability across replicates: we should expect LS with 500 annotations to underperform 1,000 annotations asymptotically.) Additional experiments in Appendix C.1 show that these improvements come at no cost to generation *diversity*. In TRIP (and Recipes, see Appendix C.2), state supervision is unhelpful. This is consistent with the prediction in §3 that state supervision is helpful only when LM state representations are initially incorrect, fixable, and usable.

5 Conclusion

Effective generation of coherent text requires reasoning about the world that text describes. We have introduced a framework for measuring how well LMs perform this reasoning, and described an algorithm for sample-efficiently improving them. Our results point to a potentially broad role for semantic supervision in LM training—in some cases, small amounts can yield large coherence improvements.

6 Impact Statement

This work introduces ways of using state supervision for diagnosing and 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 training with small amounts state supervision can, in certain circumstances, be comparable to to training with a much larger amount of text-only supervision. Thus, state supervision could be one way to reduce the scale of pre-training, which would be both environmentally friendlier, and would reduce the risk of unknowingly pre-training on toxic content or private data.

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. Furthermore, the methods described here are imperfect and work only in certain domains (§4). We encourage future work to examine different ways of incorporating state annotations, and also to think carefully about what kinds of state annotations would be useful when developing new datasets.

References

Abubakar Abid, Maheen Farooqi, and James Zou. 2021. *Persistent Anti-Muslim Bias in Large Language Models*, page 298–306. Association for Computing Machinery, New York, NY, USA.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, and Jared Kaplan. 2021. A general language assistant as a laboratory for alignment.

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.

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.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Irene Heim. 2012. *File change semantics and the familiarity theory of definiteness*. de Gruyter.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. 2016. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339, Austin, Texas. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. 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.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020b. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

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.

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.

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.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation.

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.

Niket Tandon, Keisuke Sakaguchi, Bhavana Dalvi, Dheeraj Rajagopal, Peter Clark, Michal Guerquin, Kyle Richardson, and Eduard Hovy. 2020. A dataset for tracking entities in open domain procedural text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6408–6417, Online. Association for Computational Linguistics.

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.

Fangzhou Zhai, Vera Demberg, Pavel Shkadzko, Wei Shi, and Asad Sayeed. 2019. A hybrid model for globally coherent story generation. In *Proceedings of the Second Workshop on Storytelling*, pages 34–45, Florence, Italy. 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.

A OpenPI Dataset: State Inference Accuracy & NSP on Accurate States

In addition to the datasets in §3.1, we also ran experiments on **OpenPI** (Tandon et al., 2020), which consists of instruction sequences annotated with the changes to entity states that result from executing each instruction.

As shown in Fig. 5(a), in OpenPI, similar to in TRIP and in Recipes, training with state supervision does not improve state inference accuracy. For domains where state supervision does not help, state inference is either too difficult, even in the presence of explicit supervision (OpenPI), or too easy and already learned from text-only training (TRIP).

In Fig. 5(b), it appears that conditioning the LM on states result in more accurate next sentence predictions than conditioning the LM on text in OpenPI. However, this is simply because the state contains the full set of entities, even unknown ones that have not yet appeared in the prior context. This is supported by the fact that conditioning the LM on the full set of entities alone results in comparable gains as conditioning on state. This is a quirk of the contrastive evaluation—the LM simply needs to check whether the entities mentioned in the next sentence matches one of the entities in the state. Thus, we cannot conclude that the state, in and of itself, is any more informative of the next sentence than the text context. (It is natural to ask whether this is also the case for TW. However, in TW, we are conditioning on the *current belief state* rather than the full state, which contains only *known facts about known entities*. See more in §B.)

B Constructing the State

B.1 State supervision in each domain

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. Sample states from each domain can be found in Table 1.

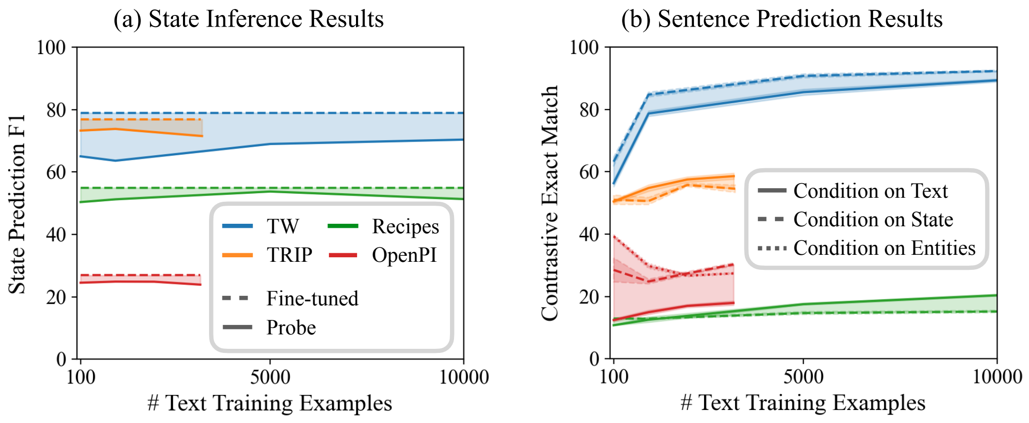


Figure 5: (a) State inference accuracies and (b) sentence prediction accuracies for all 4 domains (TW, TRIP, Recipes, OpenPI). We run the experiments described in §3.2 and §3.3 on the OpenPI domain. Results in OpenPI mirror results in TRIP and Recipes: training explicitly on state does not significantly reduce state inference errors, and states are difficult to use for next sentence prediction (conditioning NSP on state is comparable to conditioning on just the full set of entities alone).

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 **belief 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 **current belief state**. (We explore whether this choice to use different subsets of the state actually matters in §B.2).

We compute both these sets heuristically: the belief state consists of all facts about any currently or previously accessible entities that the agent has encountered. For the *current* belief state, we discard all facts about previously accessible entities and keep only facts about currently accessible entities. Specifically, the current belief set 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 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, each sentence of each 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 $(\textit{entity}, \textit{attribute}, \textit{value})$ triples. Each attribute is annotated with a value indicating the direction of change for that attribute. For example, $(\textit{shirt}, \textit{cleanliness}, \textit{true} \rightarrow \textit{false})$ indicates *the shirt became dirty*.

We directly use the provided annotations in TRIP, without filtering for known/unknown facts as in TW. We do this simply out of ease—TW is synthetically generated, allowing us to hard-code rules for discovering the known subset, while TRIP is real and more complex. However, note in TRIP that only the relevant entities and attributes are annotated, rather than the full state, so the provided annotations are already a specially-chosen subset of the full state.

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

Recipes In Recipes, each instruction is annotated with the list of ingredients that have undergone state changes (e.g. *sugar, dough, apples*), and all new states induced by the events in the instruction, given as a list of $(\textit{attribute}, \textit{value})$ pairs (e.g. (*temperature, hot*); (*shape, sliced*)).

Since the provided annotations did not specify which state changes were associated with which entity, we heuristically convert these into entity-

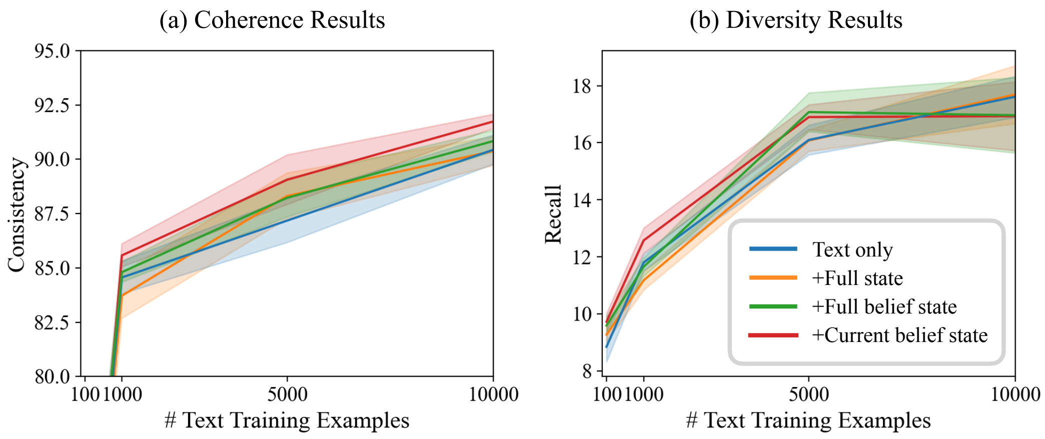


Figure 6: Coherence and diversity improvements from training with different subsets of the state as auxiliary supervision in TextWorld. We see that the choice of state matters, and that using just the causally significant portions of the state (red line) outperforms using the full state (orange line).

centric facts by simply assuming all state changes have been applied to all ingredients. Note this can often be a faulty assumption—for example, in Table 1, the `deformed shape` attribute only applies to the chocolate rather than the water.

Finally, we convert these (*entity*, *attribute*, *value*) triples into natural language descriptions as "*the* {*attribute*} *of* {*entity*} *is* {*value*}". For example, (*apples*, *shape*, *sliced*) becomes "*the shape of apples is sliced*".

OpenPI In OpenPI, each instruction is annotated state changes in the form of (*entity*, *attribute*, *previous value*, *new value*). We convert these to natural language as "*the* {*attribute*} *of* {*entity*} *is* {*new value*}". For example, (*eraser*, *location*, *at store*, *at home*) becomes "*the location of eraser is at home*".

B.2 What facts to use as auxiliary supervision in TW?

As noted in §B.1, in TW, we used a subset of the full state that (1) includes only facts that the agent knows about, and (2) is causally sufficient for predicting all plausible next sentences. In this section, we explore whether the choice to use this subset actually makes a difference.

Specifically, we train with auxiliary supervision (described in method 1 of §4) using the three different choices of state (described in §B.1): the full state, the belief state, and the current belief state. Results are shown in Fig. 6. We find that the training with the full state is often not significantly better than simply training on text only, and occasionally slightly worse. Training on the subset of belief facts outperforms training with the full state, and training with the smaller subset of current belief facts is even better.

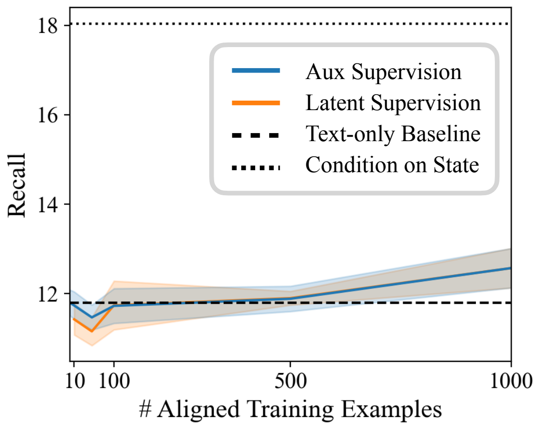


Figure 7: Impact of training with auxiliary and latent state supervision on LM generation diversity, in the TW domain. We assume access to 1000 total examples, varying the number of them that are aligned with state. We see that training with state supervision does not negatively affect diversity. However, we are still lag far behind a model that sees state at test-time (dotted line).

This highlights the importance of *choosing an appropriate state representation* when using state as supervision. The experiments here suggest choosing strategically in accordance with the causal model depicted in Fig. 1(c): we want to reinforce only facts that 1. can be deduced from context (belief state), and 2. will be used when generating the next sentence (current state).

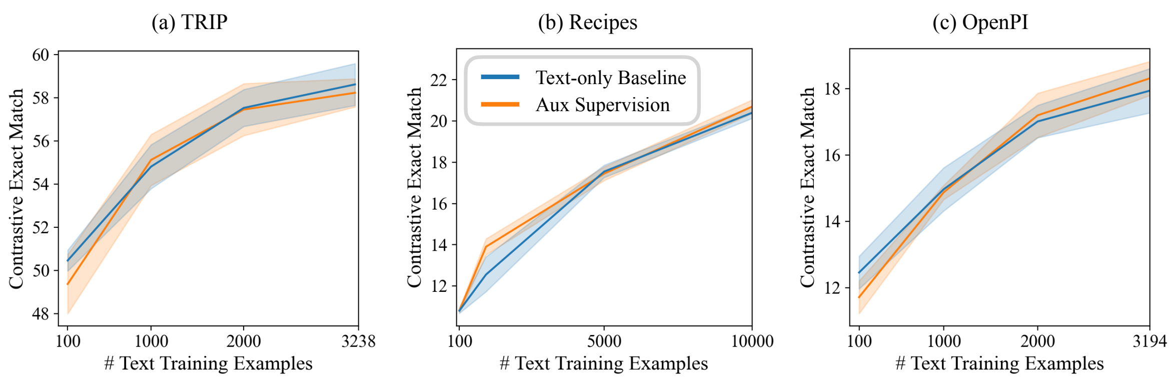


Figure 8: Training with auxiliary state supervision in TRIP, Recipes, and OpenPI. We vary the total number of examples available, and assume all examples have state annotations. We can see that in all 3 domains, training with auxiliary supervision is unhelpful.

C Additional Results for Auxiliary and Latent State Supervision

C.1 Effect of State Supervision on Generation Diversity in TW

To measure the diversity of LM outputs, we use *recall* 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 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's set of valid sentences remains a good heuristic for diversity.

We examine how training with auxiliary or latent state supervision affects generation diversity. We use the same models trained in §4 (on 1000 TW examples paired with varying amounts of state supervision) and evaluate their generation diversity. Recall from §4 that both auxiliary and latent state supervision improve coherence. As shown in Fig. 7, neither of them reduce diversity to achieve coherence gains. However, they do not improve diversity either. In Fig. 7, we train a LM on 1000 examples to explicitly predict the next sentence from state. This is plotted as a dotted line. We can see that a LM that is able to condition on state explicitly produces much more diverse generations, improving recall by over 6 points. It remains an open question as to how to transfer this large diversity improvement to the setting where state annotations are only available at training time.

C.2 Auxiliary State Supervision in Recipes and OpenPI

Results for training with auxiliary state supervision across various data sizes, in the setting where examples are fully aligned with states, is shown in Fig. 8 for Recipes, OpenPI, and TRIP. The analogous TW results can be seen by comparing the blue and red lines in Fig. 6.

In both Recipes and OpenPI, we find similar trends as TRIP: training with auxiliary state *does not* improve generation coherence. This comes at no surprise given what we found from the state inference accuracy and sentence prediction accuracy experiments in §3 and §A. State supervision is only useful in domains where state representations are fixable and usable.

D Sample LM Generations in TW

Sample generations from the language model, before & after training with state supervision, can be found in Table 2.

E Infrastructure and Reproduciility

We ran all 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 do not take 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.

Dataset	Sample T	Sample S	Sample T'

TW	*-= Garage =-*
	[...] You see a locker. You can make out a table. On the table you can make out an American limited edition passkey. There is a closed American limited edition hatch leading west. There is an unblocked exit to the north.
	> inventory
	You are carrying nothing.
	> go north
	-= Cookhouse =-
	[...] You make out a case. [...] You see a plate. [...] But oh no! there's nothing on this piece of junk. You hear a noise behind you and spin around, but you can't see anything other than a rack. But oh no! there's nothing on this piece of garbage. You can see a saucepan. The saucepan [...] has nothing on it. [...] You need an unguarded exit? You should try going south.
	> examine case
	The case looks strong, and impossible to break. You can't see inside it because the lid's in your way.
		`The plate is in cookhouse`
		`The saucepan is in cookhouse`
		`The case is in cookhouse`
		`The cookhouse is mapped north of garage`
		`The rack is in cookhouse`
		`The case is closed`
		`The garage is mapped south of cookhouse`
		`The player is in cookhouse`
			> open case

TRIP	*Tom picked up the paper from the copier.*
	Tom picked up the scissors.
		`Tom is conscious`
		`The scissors is picked up`
		`The scissors are existent`
		`The scissors are functional`
		`The scissors are moveable`
			Tom used the scissors to cut the paper.

Recipes	*Ingredients: chocolate, water, egg white, vanilla, sugar, flour, baking powder, salt, sugar.*
	In large heatproof bowl set over hot water, melt chocolate with water, stirring until smooth.
		`the shape of chocolate is deformed`
		`the cookedness of chocolate is cooked`
		`the location of chocolate is bowl`
		`the temperature of chocolate is hot`
		`the shape of water is deformed`
		`the cookedness of water is cooked`
		`the location of water is bowl`
		`the temperature of water is hot`
			Remove from heat

OpenPI	*How to spool knit a winter hat.*
	Knit eighty-four inches of flat web.
	Begin five inches from the end of the web, turn and sew into a rectangular form five inches wide and eight inches long.
		`shape of the webbing is rectangular`
		`width of the webbing is wider`
		`connection of the ends is unjoined`
		`state of ends is loose`
		`state of the ends is unattached`
		`shape of the yarn is rectangular`
		`state of the top is open`
			Join the five-inch ends, and draw in the top with the needle and a piece of the material from which the cap was made.

Table 1: Sample contexts, states, and next sentences from each domain.

Context	1k Text-only Examples	1k Text Examples + 500 State Annotations (Latent supervision)
-= Garage =- [...] You see a locker. You can make out a table. On the table you can make out an American limited edition passkey. There is a closed American limited edition hatch leading west. There is an unblocked exit to the north. > inventory You are carrying nothing. [...] > go north -= Cookhouse =- [...] You make out a case. There's something strange about this thing being here, but you don't have time to worry about that now. You see a plate. Now why would someone leave that there? The plate is usual. But oh no! there's nothing on this piece of junk. You hear a noise behind you and spin around, but you can't see anything other than a rack. But oh no! there's nothing on this piece of garbage. You can see a saucepan. The saucepan is typical. However, the saucepan, like an empty saucepan, has nothing on it. Hm. Oh well You need an unguarded exit? You should try going south. > examine case The case looks strong, and impossible to break. You can't see inside it because the lid's in your way. [...]	> examine plate > examine plate > examine plate > examine plate > open plate	> examine plate > examine plate > examine plate > examine plate > open American limited edition hatch
-= Bar =- [...] You see a box. The box is empty, what a horrible day! You can see a basket. Something scurries by right in the corner of your eye. Probably nothing. You can make out a suitcase. You can see a recliner. But the thing is empty, unfortunately. Were you looking for a mantle? Because look over there, it's a mantle. Looks like someone's already been here and taken everything off it, though. If you haven't noticed it already, there seems to be something there by the wall, it's a bed stand. But the thing is empty. There is a closed type Z door leading west. You don't like doors? Why not try going east, that entranceway is unguarded. > inventory You are carrying nothing. > open basket You open the basket. [...] > go east -= Salon =- [...] As if things weren't amazing enough already, you can even see a portmanteau. Look over there! a couch. Now why would someone leave that there? The couch is normal. On the couch you can see a type Z latchkey. There is an exit to the west. Don't worry, it is unblocked. [...] > examine portmanteau The portmanteau looks strong, and impossible to destroy. You can't see inside it because the lid's in your way. [...] > open portmanteau You open the portmanteau.	> go west > go west > examine couch > examine bed stand > examine bed stand	> go west > go west > examine couch > examine suitcase > take type Z latchkey from couch

Table 2: LM generations in TW, before and after training with state supervision. We take 5 samples from each LM for each context. Green indicates coherent generations. Red indicates incoherent generations.