Baked-in State Probing

Shubham Toshniwal[1], Sam Wiseman[2], Karen Livescu[3], Kevin Gimpel[3]
[1]Meta AI
[2]Duke University
[3]Toyota Technological Institute at Chicago
shtoshni@meta.com, swiseman@cs.duke.edu, {klivescu, kgimpel}@ttic.edu

Abstract

Neural language models have been analyzed for their linguistic and extra-linguistic knowledge via probing. Of particular interest has been the following question: how much can a language model trained only on *form* learn about *meaning*? Recent work has demonstrated via probing classifiers that in the setting of simple procedural text, where by "meaning" we mean the underlying world state, language models have a non-trivial performance on world state tracking. However, our proposed evaluation based on model predictions shows differing results, suggesting that these models are either not capturing the world state or not using it. How do these results change if the model has access to the world state? We explore this alternate setting with access to the underlying world state only during training and investigate ways of "baking in" the state knowledge along with the primary task of language modeling. Our proposed approaches allow for state probing during inference simply via text prompts, avoiding any probing classifier machinery. In terms of performance, we show that baking in the state knowledge during training leads to significant improvements in state tracking performance and text generation quality.[1]

1 Introduction

There has been recent interest in the extent to which models trained solely on text (*form*) capture aspects of the underlying world state (*meaning*) (Bender and Koller, 2020; Bisk et al., 2020; Wu et al., 2021; Bender et al., 2021; Li et al., 2021; Toshniwal et al., 2022). Li et al. (2021) show via the use of probing classifiers that language models trained for simple semantic domains learn to represent the state of the world described by the text without any explicit state supervision. However, one of the key limitations of probing classifiers is that the extractability of any information, such as world state, doesn't necessarily mean the model is using this information (See Belinkov (2022) for a detailed discussion on limitations of probing classifiers). Our proposed evaluation based on model predictions shows results contrary to Li et al. (2021) wherein we find that the model performance is close to chance in capturing/utilizing the state knowledge (see Table 3).

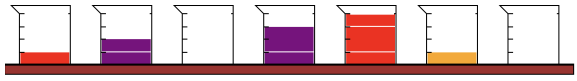
How do these results change if the language model has access to the ground truth world state knowledge? We explore this alternate setting where we assume that: (a) the language model has access to the world state *only during training*, and (b) that the world state (or parts of it) can be translated to text. Access to such oracle annotations during training has also been explored in recent work (Nye et al., 2021; Lampinen et al., 2022). Concretely, we build on the setup of Li et al. (2021) where we explore simple approaches to explicitly adding the state to the language model training sequences. We show that this *baking-in* of the world state knowledge allows for state probing during inference simply via prompting. Our experiments on the Alchemy dataset (Long et al., 2016) show that our proposed approaches, particularly a multi-task learning approach, significantly improve over a baseline language model both in terms of state prediction and language modeling. More broadly speaking, the proposed approaches could present an alternate way for injecting linguistic knowledge during end task training for NLP tasks (Wu et al., 2021).

2 Task Setup

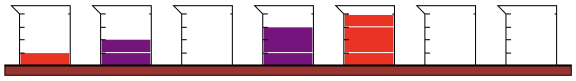
In this section, we first formally describe the setup in which we conduct the language modeling and state probing experiments. Following that, we briefly describe the probing classifier setup used by Li et al. (2021), and finally our proposed lan-

[1]Code and resources at https://github.com/facebookresearch/state_probing_lm

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 5430–5435
December 7-11, 2022



W_0



W_1

$s_1 \rightarrow$ `throw out the orange beaker`

$\mathcal{T}(W_0) \rightarrow$ `the first beaker has 1 red, the second beaker has 2 purple, ⋯, the second to last beaker has 1 orange, the last beaker is empty`	$\mathcal{T}(W_1) \rightarrow$ `the first beaker has 1 red, the second beaker has 2 purple, ⋯, the second to last beaker is empty, the last beaker is empty`
$\mathcal{T}(W_0, \text{fifth beaker}) \rightarrow$ `fifth beaker has 4 red`	$\mathcal{T}(W_1, \text{fifth beaker}) \rightarrow$ `fifth beaker has 4 red`

Table 1: Sample Alchemy instance. W_0 represents the initial world state while W_1 represents the world state at the end of sentence s_1. $\mathcal{T}(.)$ represents the state translator which given the world state and optionally an entity, outputs a natural language description of the world/entity state. The Alchemy world consists of beakers (entities) and the state of a beaker is represented by the volume of the colored liquids residing in it.

guage model variants which assume access to state knowledge during training.

2.1 Task Description

Let $(s)_{i=1}^{n}$ represent the segmentation of a text discourse into a sequence of sentences. Suppose the world described by this text consists of entities $(e)_{j=1}^{m}$. Let $(W)_{i=0}^{n}$ represent the sequence of world states where W_0 represents the initial world state and W_i represents the world state at the end of sentences $s_{1:i}$. We assume that the world state is simply the aggregation of the state of all the entities in the world. Formally speaking, let $\mathcal{S}(e_j, s_{1:i})$ represent the state of the entity e_j at the end of sentences $s_{1:i}$, then $W_i = [\mathcal{S}(e_1, s_{1:i}) \cdot \mathcal{S}(e_2, s_{1:i}) \cdots \mathcal{S}(e_m, s_{1:i})]$. Finally, we assume access to a state translator $\mathcal{T}$ which takes as input the entity or world state and outputs a corresponding natural language description. Figure 1 illustrates the instantiation of the concepts discussed in the context of Alchemy. We next describe the language modeling and state probing task avoiding any model-specific details.

Language Modeling We finetune the sequence-to-sequence (seq2seq) language model BART (Lewis et al., 2020) in all our experiments. The input to the seq2seq language model is $\mathcal{T}(W_0) \cdot s_{1:i}$ i.e. the concatenation of the initial state translated to text and the initial i sentences of the discourse. The language model is (primarily) trained to predict the next sentence but the exact output sequence differs across models (see Table 2). After training the language model, we probe the model for its state tracking capabilities which we define next.

State Probing tasks require predicting the world state W_i given the initial state W_0 and the first i sentences of the discourse $s_{1:i}$. Since the world state is simply an aggregation of entity states, the world state probing task can be decomposed to predicting the states of all the entities involved in the discourse.

Given these generic description of the language modeling and state probing task, we next describe the model variants. The input sequence to the seq2seq language model remains the same across all the model variants. Hence, we limit the model details to: (a) the output sequence with which the language model is trained, and (b) how state probing is done post language model training.

2.2 Baseline Model

For our baseline, we use the setup of Li et al. (2021). The baseline model is the canonical language model trained to predict the next sentence of the discourse s_{i+1}.

State Probing Li et al. (2021) predict the state of each entity individually with a linear probing classifier (number of world states grows exponentially in the number of entities for Alchemy). Predicting the state of each entity requires extracting their representation from the discourse representation for which Li et al. (2021) employ heuristics.

2.3 Baking-in State Knowledge

In this section, we assume access to the ground truth world state W_i, or equivalently $\mathcal{T}(W_i)$, during language model training. We propose two approaches to explicitly adding the state sequence $\mathcal{T}(W_i)$ to language model training sequences. We first describe the language model training details

Model	Training Prob.	Training Output Sequence	Inference Output Sequence	Probing via Prompt
Baseline	1	s_{i+1}	s_{i+1}	✗
Multitask Learning	p $(1-p)$	[START] $\mathcal{T}(W_i)$ [END] s_{i+1}	s_{i+1}	✓
RAS	p $(1-p)$	[START] $\mathcal{T}(W_i)$ [END] s_{i+1} s_{i+1}	s_{i+1}	✓

Table 2: Summary of Output Sequences during Training and Inference for all the language model variants. W_i denotes the world state at the end of first i sentences $s_{1:i}$ of the discourse. $\mathcal{T}(W_i)$ denotes the translation of world state to natural language text. [START] and [END] denote the special tokens used to mark the state description boundary.

for the proposed models, and then describe the shared *baked-in probing* methodology where state probing can be done simply via prompting the language model.

2.3.1 Multitask Learning

In multitask learning, we train the language model to predict the next sentence s_{i+1} with probability $(1-p)$ or the current world state [START] $\mathcal{T}(W_i)$ [END] with probability p where the probability p is a hyperparameter, and the delimiter tokens [START] and [END] denote that the text sequence is about world state. In practice, we use the probabilities of the tasks to scale the prediction losses i.e.

$$\mathcal{L}_{mult} = p\mathcal{L}_{state} + (1-p)\mathcal{L}_{next}$$

where $\mathcal{L}_{state}$ is the loss for predicting the state sequence, and $\mathcal{L}_{next}$ is the loss for predicting the next sentence.

2.3.2 Randomly Added State (RAS)

In the multitask learning approach, the current state prediction and next sentence prediction are treated as separate prediction tasks. In RAS, we combine the two tasks stochastically. Specifically, for some chosen probability p, the RAS(p) language model is trained to predict the concatenated sequence [START] $\mathcal{T}(W_i)$ [END] s_{i+1} with probability p, and with probability $(1-p)$ it is trained on the canonical task of predicting the next sentence s_{i+1}. The RAS model allows for the flexibility of Multitask Learning model in the sense that both the state sequence (see discussion on State Probing later in this Section) and next sentence can be independently predicted. At the same time during training it allows the model to learn the relationship between the two tasks.

Note that the RAS($p = 1$) model is always trained to predict the next sentence with the state sequence. Hence, during inference we first predict the state sequence, and then predict the next sentence.

2.3.3 State Probing

In the above proposed models, the language model is either explicitly trained to predict the world state or the world state is a prefix of the predicted sequence. Moreover, the state sequence $\mathcal{T}(W_i)$ is delimited by the special tokens [START] and [END]. Thus, prompting the decoder of the trained seq2seq language model with [START] token conditions the model to generate the state.[2]

For Alchemy, we can enumerate all the possible states for any entity (beaker) and score them with the language model, the predicted entity state being the one with the highest probability.[3] We predict the world state in Alchemy by individually predicting the state for each entity (beaker). [4]

2.4 Next Sentence Probability

For the baseline model, the probability of next sentence $P(s_{i+1})$ is trivially calculated using the autoregressive decoder of the seq2seq language model (for ease of notation we hide the conditioning on W_0 and $s_{1:i}$). For the proposed model variants, we additionally mask out the probability assigned to the [START] and [END] tokens at all timesteps i.e. renormalize the distribution after zeroing out the probability assigned to the [START]

[2]This bears similarity with the use of control tokens in text generation (Keskar et al., 2019; See et al., 2019).

[3]Any beaker in our Alchemy setup has one of 210 states.

[4]During training, we shuffle the order in which the beaker states are presented in $\mathcal{T}(W_i)$. We do this to avoid bias towards any particular order in which the language model is used to predicting the beaker states.

Model	Perplexity	World State	Entity State	Valid Next Sentence
Baseline (Li et al., 2021)	2.98	7.6	75.0	48
Multitask Learning	**2.91**	**57.8**	**92.2**	70
RAS	**2.91**	49.3	90.1	**74**

Table 3: Comparison of performance of Language Model variants on the proposed evaluations. The state tracking results for the Baseline model are from Li et al. (2021).

and `[END]` tokens as the special tokens are only used while predicting the state.

For the RAS ($p = 1$) model, calculating the exact $P(s_{i+1})$ is non-trivial because the model conditions the next sentence prediction on the state prediction. We report an approximation of $P(s_{i+1})$ (for details see Appendix A.1.1).

3 Experimental Details

3.1 Hyperparameter Details

For all our experiments, we use the BART-base language model. For the Multitask and RAS model, we tune the probability p of the auxiliary task over $\{0.1, 0.2, 0.3, \cdots, 0.8, 0.9, 1.0\}$. Training details in Appendix A.1.2.

3.2 Data and Evaluation Details

We borrow the Alchemy setup from Li et al. (2021) (see Appendix A.1.3 for details). For the validation split, we report: (a) *Perplexity*, and (b) *World/Entity State Accuracy*: The world state accuracy measures the instances for which the model predicts the correct state for all the entities while the entity state accuracy measures the accuracy over predicting the correct entity state.

Valid Next Sentence Evaluation Additionally, we create an artificial evaluation set of 100 Alchemy language modeling instances where for each input we also have the exact set of valid outputs i.e. next sentences which can be "executed" given the world state. For each input, the model selects the most probable next sentence among all the candidate next sentences. We then measure the *Valid Next Sentence Accuracy* which is how often does a model select a valid next sentence as the top choice. This evaluation is meant to test the models behaviorally in terms of whether the models can use their world state knowledge or not while predicting the next sentence. A model doing poorly on this evaluation is either failing at state tracking or unable to use its state knowledge while predicting the next sentence. For context, a naive baseline of selecting the next sentence among the 100 candidate next sentences using a uniform distribution gets an accuracy of 45.56 ± 4.65.

4 Results

Table 3 presents the results for the baseline model and our proposed language model variants. Across all evaluations we see a clear benefit of having access to state knowledge during training. In particular, the Multitask Learning model improves over the baseline model on the world state prediction task by about 50 points absolute (and also avoids training a separate probing classifier). For the language modeling task, we see a drop in perplexity for both the proposed models in comparison to the baseline model suggesting that the state knowledge also aids the language modeling task. This is even more emphatically reflected in the Valid Next Sentence evaluation where the proposed variants improve over the baseline language model by about 40% relative. Interestingly, the baseline language model gets an accuracy of 48 on this task which is within one standard deviation of the random baseline's score. This suggests that even if the baseline language model trained on just the text discourse (form) has implicitly learned the state (meaning), it most likely has not learned to use the state knowledge while predicting the next sentence. Among our proposed variants the Multitask Learning variant excels in predicting the state while the RAS model is the best at utilizing the state knowledge in next sentence prediction.

5 Conclusion

We show via our proposed evaluation of Valid Next Sentence prediction that a baseline language model is at par with chance performance, suggesting that the model struggles to capture the state knowledge. Based on this evidence, we explore language model variants which assume access to

the world state during training. The proposed language model variants can be easily probed for world state via text prompt, and significantly improve over a canonical language model for both state tracking and text generation. Our results suggest that injecting semantic knowledge could be beneficial more broadly as well, and our proposed approaches can be useful for this end.

6 Limitations

Our work is limited to a simple setup where we assume: (a) access to the ground truth state knowledge, and (b) that this state knowledge can be deterministically translated to text. Both these assumptions don't necessarily hold true in most real world settings. Our baked-in probing approach requires knowing the probing task before training the model which again may not be true in general settings.

References

Yonatan Belinkov. 2022. Probing Classifiers: Promises, Shortcomings, and Advances. *Computational Linguistics*, 48(1):207–219.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*.

Emily M. Bender and Alexander Koller. 2020. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online. Association for Computational Linguistics.

Yonatan Bisk, Ari Holtzman, Jesse Thomason, Jacob Andreas, Yoshua Bengio, Joyce Chai, Mirella Lapata, Angeliki Lazaridou, Jonathan May, Aleksandr Nisnevich, Nicolas Pinto, and Joseph Turian. 2020. Experience grounds language. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8718–8735, Online. Association for Computational Linguistics.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A conditional transformer language model for controllable generation. *CoRR*, abs/1909.05858.

Andrew K. Lampinen, Ishita Dasgupta, Stephanie C. Y. Chan, Kory Matthewson, Michael Henry Tessler, Antonia Creswell, James L. McClelland, Jane X. Wang, and Felix Hill. 2022. Can language models learn from explanations in context?

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.

Reginald Long, Panupong Pasupat, and Percy Liang. 2016. Simpler context-dependent logical forms via model projections. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1456–1465, Berlin, Germany. Association for Computational Linguistics.

Maxwell I. Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. Show Your Work: Scratchpads for Intermediate Computation with Language Models. *CoRR*, abs/2112.00114.

Abigail See, Stephen Roller, Douwe Kiela, and Jason Weston. 2019. What makes a good conversation? how controllable attributes affect human judgments. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1702–1723, Minneapolis, Minnesota. Association for Computational Linguistics.

Shubham Toshniwal, Sam Wiseman, Karen Livescu, and Kevin Gimpel. 2022. Chess as a testbed for language model state tracking. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI 2022)*.

Zhaofeng Wu, Hao Peng, and Noah A. Smith. 2021. Infusing Finetuning with Semantic Dependencies. *Transactions of the Association for Computational Linguistics*, 9:226–242.

A Appendix

A.1 Miscellaneous

A.1.1 Perplexity Calculation for RAS($p = 1$) Model

For the RAS($p = 1$) model, calculating the exact $P(s_{i+1})$ is challenging because the model conditions the next sentence prediction on the state prediction. To explain this challenge, suppose the world state space is $\mathcal{Z}$. So under the RAS($p = 1$) model, we would be required marginalize over the entire world state space i.e.

$$P(s_{i+1}) = \sum_{z \in \mathcal{Z}} P(s_{i+1}|z)P(z)$$

Since this marginalization is impractical for Alchemy, we instead report:

$$P(s_{i+1}) = \max_{z \in \mathcal{Z}} P(s_{i+1}|z)$$

Thus, the perplexity results for the RAS($p = 1$) model are not directly comparable to other variants.

A.1.2 Additional Training Details

The model is trained for a maximum of 100 epochs with a batch size of 24. Validation set perplexity is computed at the end of every epoch and training stops after no improvement for 5 consecutive epochs. We use the Adam optimizer with an initial learning rate of 10^{-5}. The learning rate is warmed up linearly for the first 10 epochs followed by a linear decay.

A.1.3 Additional Data Details

The dataset has 3657 train and 245 validation instances which correspond to 14,628 and 980 language modeling instances respectively – 5 sentences per original instance, thus, 4 next sentence prediction task per original instance.

A.1.4 Infrastructure Details

All our models are trained on 32GB V100 GPUs. Training for any model finishes within couple of hours.