Entity Tracking in Language Models

Anonymous ACL submission

Abstract

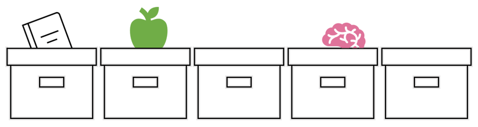
Keeping track of how states and relations of entities change as a text or dialog unfolds is a key prerequisite to discourse understanding. Despite this fact, there have been few systematic investigations into the ability of large language models (LLMs) to track discourse entities. In this work, we present a task to probe to what extent a language model can infer the final state of an entity given an English description of the initial state and a series of state-changing operations. We use this task to first investigate whether Flan-T5, GPT-3 and GPT-3.5 can track the state of entities, and find that only GPT-3.5 models, which have been pretrained on large amount of code, exhibit this ability. We then investigate whether smaller models pretrained primarily on text can learn to track entities, through finetuning T5 on several training/evaluation splits. While performance degrades for more complex splits, we find that even for splits with almost no lexical overlap between training and evaluation, a finetuned model can often perform non-trivial entity tracking. Taken together, these results suggest that language models can learn to track entities but pretraining on large text corpora alone does not make this capacity surface.

1 Introduction

A key prerequisite to long-context understanding and generating coherent text is the ability to accurately represent entities as the discourse unfolds (Karttunen, 1976; Groenendijk and Stokhof, 1991; Heim, 2002; Nieuwland and Van Berkum, 2006; Kamp et al., 2011, *i.a.*). For example, consider the following example in the context of a recipe:

(1) Put the eggs, sugar, flour, and baking powder in a bowl and mix to form a light batter. Make sure that the final batter does not contain any lumps of flour or sugar.

In order to understand this instruction, several distinct abilities are necessary:



Q: Box 1 contains the book. Box 2 contains the apple. Box 4 contains the brain. Move the book into Box 2. Put the bell into Box 4. Move the bell and the brain into Box 5. Box 2 contains ____
A: the apple and the book

Figure 1: A sketch of our entity tracking task.

New discourse entity recognition: recognizing when new discourse entities are introduced. E.g., *a bowl* introduces a new discourse entity but *the final batter* or *any lumps of* ... does not.

Coreference resolution: associating referring expressions with the right discourse entities. E.g., *a light batter* and *the final batter* refer to the same entity.

Discourse entity tracking: tracking the state changes made to each discourse entity. E.g., *the eggs* are put into *the bowl* and *mixed* with the other ingredients.

There exist many datasets that aim to evaluate these abilities (e.g., Walker et al., 2006; Pradhan et al., 2012; Rahman and Ng, 2012; Weston et al., 2015; Chen et al., 2018; Bamman et al., 2020; Uryupina et al., 2020) and many NLP models that aim to solve these tasks (e.g., Haghighi and Klein, 2010; Lee et al., 2011; Hill et al., 2016; Henaff et al., 2017; Ji et al., 2017; Lee et al., 2017; Bosselut et al., 2018; Gupta and Durrett, 2019a,b; Aina et al., 2019; Toshniwal et al., 2020; Wu et al., 2020). In the context of large language models (LLMs), Tenney et al. (2019), Clark et al. (2019), and Sorodoc et al. (2020) found that representations of LSTMs and Transformer-based models such as BERT (Devlin et al., 2019) do capture coreference relations. Loáiciga et al. (2022) and Schuster and Linzen (2022) found that pretrained models are able to detect whether noun phrases introduce discourse entities, albeit not fully systematically.

The question of whether LLMs *can track the state of discourse entities*, however, has only been indirectly evaluated so far. Toshniwal et al. (2022) showed that GPT-2 (Radford et al., 2019) can learn to predict valid chess moves based on a compact, nonlinguistic description of previous moves. While this suggests that LLMs such as GPT-2 can learn the rules of chess and can learn to track the states in the game, it does not tell us whether LLMs track the entity states in natural language discourses. Li et al. (2021) evaluated whether the state of an entity can be decoded from LLM representations, which most directly aims to evaluate entity tracking abilities. Using a probing classifier, they found that the state of an entity can be decoded from T5 and BERT with high accuracy. However, as we show in a reanalysis of their results (Section 2), their results do not provide definitive evidence for entity tracking. Hence, whether LLMs can track entities during the processing of natural language discourse remains an open question.

Contributions This work attempts to answer this question by developing a task targeted towards evaluating a language model's ability to track state changes of discourse entities (illustrated in Figure 1). We use this novel task to evaluate GPT-3 (Brown et al., 2020), GPT-3.5,[1] and Flan-T5 (Chung et al., 2022) models without any additional finetuning. We find that only models in the GPT 3.5 series, which have been trained on both text and code, are able to perform non-trivial entity tracking. We then show that a smaller language model (T5: Raffel et al. 2020) can learn to perform non-trivial entity tracking, demonstrating the capacity to generalize to state descriptions with little lexical overlap. Our results suggest that language models can learn to track entities but pretraining on text corpora alone does not make this capacity surface.[2]

2 Reanalysis of results from Li et al. (2021)

We start from examining Li et al. (2021), the most relevant work to ours. They adapted two existing datasets, Alchemy (Long et al., 2016) and TextWorld (Côté et al., 2019), to test a model's ability to track state changes of an entity. The input to the model is a text description of the initial world state followed by state-changing instructions. Based on this description, the model is expected to identify the correct final state of each entity. For example, for Alchemy, the model receives formulaic descriptions of 7 beakers filled with different amounts of colored liquids, followed by instructions that manipulate the contents of the beakers such as pouring the liquid from one beaker into another, or draining the liquids from a beaker. Given an input like (2), the model is expected to recognize that the first beaker has 4 brown, the second beaker has 2 red, and the third beaker is empty.

(2) *The first beaker has 1 green, the second beaker has 2 red, the third beaker has 3 red. Pour the last red beaker into beaker 1. Mix.*

Using such descriptions, Li et al. (2021) found that a probing classifier that takes as input the encoding of these descriptions from T5 and BART (Lewis et al., 2020) is able to correctly predict the state of 75–76% of the entities, suggesting some degree of success on entity tracking.

However, this conclusion becomes questionable when the datasets and the results are scrutinized further. Specifically, we conducted a fine-grained analysis of the success cases of the Alchemy experiment. In this experiment, the state of each beaker was probed after each state-changing instruction. Because each instruction targets at most two beakers (e.g., *pour X into Y*) and there are 7 beakers in total, there is a sparse representation of cases probing a beaker that actually underwent a change in the dataset. Indeed, 62.7% of all beaker states probed were identical to the initial state, meaning that a simple baseline that always predicts the initial state already achieves 62.7% accuracy (this is also noted by Li et al.). A second potential for shortcuts was the high rate of empty final states (32.4%).[3] For these cases, the initial state can often be entirely disregarded, due to the presence of an emptying instruction such as *Drain the fourth beaker*—this instruction alone is sufficient to predict the fourth beaker's final state independent of its initial state. Therefore, such examples are also not best suited to fully assess entity tracking. Given the high prevalence of these two trivial scenarios (87.6% in total), only 12.4% of the datapoints can be considered as truly assessing state changes unfolding over a discourse context. If the accuracy

[1] `https://beta.openai.com/docs/model-index-for-researchers/models-referred-to-as-gpt-3-5`

[2] Code and data available at `anonymized`

[3] This percentage also includes cases where a beaker was already empty in the initial world state description. The percentage of cases where a beaker was empty but contained some liquid in the initial description is 24.9%.

is computed on the trivial and non-trivial cases separately, the probing classifier achieves 86.8% accuracy on trivial cases but only 3.1% accuracy on non-trivial cases, showing that most of the reported success derives from the trivial cases.

In summary, our reanalysis suggests that the results of Li et al. (2021) do not provide conclusive evidence for non-trivial state tracking abilities in language models.[4] However, it remains unclear whether this is due to issues with the setup or a true lack of entity tracking capacity. To this end, we propose a new behavioral evaluation.

3 Task Design and Dataset

3.1 Desiderata

The ability to track entities should be largely *independent of specific linguistic forms*. For a model that can properly track entities, it should not matter whether one talks about beakers or recipes or which specific syntactic constructions are used. This makes it an interesting ability to evaluate in the context of assessing whether and how meaning is represented, since at least classic language models are only trained on forms (Bender and Koller, 2020). At the same, this independence of form and entity states poses a challenge in evaluation design, since one needs to ensure that the training data does not allow the model to predict the state of entities from individual lexical items (such as the word *drain* in the Alchemy dataset, as discussed in Section 2). Furthermore, language models pretrained on large text corpora may have learned common states of entities; for instance, that eggs often end up in a bowl. For these reasons, any task that evaluates entity tracking abilities should conform to the following four desiderata:

1. The probed states of entities should not follow similar distributional patterns to those that are likely to be present in the pretraining data (see also Linzen, 2020).

2. Individual words or phrases should not predict by themselves the state of an entity without considering the remaining discourse.

3. If any data is used for demonstration, finetuning or training of probing classifiers, the training and evaluation data should have little lexical overlap.

4. If any data is used for demonstration, finetuning or training, it should not be possible to solve this task by filling in slots of a fixed template.

[4] Li et al. (2021) also presented two other sets of experiments. See Appendix A for details on how the other experiments exhibit similar issues.

These properties cannot be guaranteed with naturalistic datasets such as recipes (Kiddon et al., 2015), science texts (Dalvi et al., 2019), or the Alchemy and TextWorld datasets, which have been previously used to evaluate entity tracking abilities. We therefore programmatically generated a dataset for which these properties hold.

3.2 Dataset

We take inspiration from the evaluation setup of Li et al. (2021) in designing our data. Our dataset consists of text descriptions of a particular state of the world followed by a sequence of changes. The worlds contain boxes that can be filled with objects. The objects can be placed inside the box, taken out of the box, or moved from one box to another. We define a world $\mathcal{W}$ as $\mathcal{W} = (O, n, m, e)$ where O is a set of objects, n is the number of boxes, m is the maximum number of objects one box can contain, and e is the expected number of objects in each box in the initial world states. For our dataset, we used $n = 7$, $m = 3$, $e = 2$, and used a set of nouns denoting items that can plausibly fit inside a box (e.g., *book*, *rock*, *brain*; $|O| = 100$), selected from a list of words with frequency greater than 27 in the British National Corpus (BNC; Leech et al. 2001).

A dataset consists of multiple distinct *scenario*. A scenario consists of an initial state and a set of operations applied to this initial state. We fixed the number of operations (NumOps) in each scenario to 12. We randomly sampled 2200 scenarios, where the initial state and the 12 operations were both randomly sampled. The sampling process is designed such that only valid operations given the current world state can be sampled. The initial state and the operations were converted into naturalistic descriptions using predefined templates.

We selected the task of moving objects across boxes because this is a domain where lexical contents of the entities do not offer cues to predict the outcome of state changes (Desideratum 1). To satisfy Desideratum 2, we did not include an operation that empties a box that allows the previous history of operations to be disregarded. For Desideratum 3, we considered experiments where the state and operation descriptions differ entirely between demonstration/finetuning and evaluation examples (See

Table 2). Finally, we computed a "signature" of every initial state that indicates the number of objects contained in each box.[5] Using this signature, we then made sure that there were no two examples with identical initial descriptions modulo the object identities where one of them appeared in the training split and the other one in the evaluation split. This prevents that models could solve this task by filling in slots (Desideratum 4). Additionally, compared to the Alchemy setup, our setup also has a benefit of requiring fewer additional reasoning abilities. The beaker scenario requires the model to be able to count and do simple arithmetic (e.g., inferring that adding one unit of liquid to a beaker with two units of liquid results in a beaker with three units of liquid). Moreover, some of the operations in Alchemy require knowledge about how colors are combined (e.g., inferring that mixing red and green liquids results in a brown liquid). The boxes domain removes these requirements.

3.3 Task

We define the entity tracking task as follows. Given a natural language description of the initial state of the world followed by 0–12 descriptions of operations that have been performed, the content of each box at the end of the description must be correctly identified. To this end, we created an example for each box after each operation. This corresponds to $n \times (\text{NumOps} + 1)$ questions per scenario (i.e., 91 questions in our dataset). Each question is formulated in the style of a cloze test, a format that language models are typically trained on. That is, the input string describes the initial state followed by a sequence of operations, then followed by *Box n contains __*. The expected output is the correct set of objects in Box n based on the prefix description. See Appendix B for an example.

4 Experiment 1: In-context Demonstration

In the first set of experiments, we used a setup in which the models are provided a small number of in-context demonstrations of the entity tracking task. This provides a way to probe the model without providing substantial supervision from which a task could be learned, as well as guiding the model to output the final state in a consistent format that can then be automatically assessed.

[5]For example, the signature of an initial state in which the first box contains two objects and the rest contains 1 object each would be 2111111.

Model	Size	Code?	Additional Training
GPT-3 `davinci`	175B	✗	-
GPT-3 `davinci-instruct-beta`	175B	✗	Human demonstrations (finetuning)
GPT-3 `text-davinci-001`	175B	✗	Human demonstrations + highly rated model outputs (finetuning)
GPT-3.5 `code-davinci-002`	175B	✓	-
GPT-3.5 `text-davinci-002`	175B	✓	Human demonstrations + highly rated model outputs (finetuning)
GPT-3.5 `text-davinci-003`	175B	✓	RL on human feedback
Flan-T5 base	250M	✗	1.6K tasks + instructions (finetuning)
Flan-T5 XL	3B	✗	1.6K tasks + instructions (finetuning)

Table 1: Summary of the models used for the in-context demonstration experiments.

4.1 Models

We used models that have been shown to support task solving from in-context demonstrations provided as part of the prompt. Specifically, we use GPT-3 175B (`davinci`: Brown et al. 2020), the most recent GPT-3.5 (`text-davinci-003`[6]), and Flan-T5 (base and XL: Chung et al. 2022).

The little information that OpenAI revealed about their models[7] suggests that `davinci` is an autoregressive language model primarily trained on text corpora. `text-davinci-003` was trained on the language modeling objective on a mix of text and code, and additionally trained with human feedback using reinforcement learning. Flan-T5 is based on T5, a sequence-to-sequence model trained on a denoising objective, that has been further instruction-finetuned on a battery of tasks. This has been shown to promote better responses to instructions with and without in-context demonstrations (Chung et al., 2022). See Table 1 for a summary of the models. We compared these models against a baseline computed by randomly outputting 0–3 objects from the set of objects that appeared in the same clauses as the box in question. Note that this baseline is stronger than a fully random baseline that selects outputs from all mentioned objects.

We evaluated the GPT models through the OpenAI API and the Flan-T5 models using the HuggingFace library (Wolf et al., 2020). See Appendix C for details about the implementation.

4.2 Prompting and Demonstrations

Our prompts consist of: (a) a general description of the task, (b) two examples of the task to demonstrate the expected format, (c) an initial state description followed by a series of operations, and

[6]https://beta.openai.com/docs/models/gpt-3
[7]https://beta.openai.com/docs/model-index-for-researchers

(d) an incomplete sentence *Box N contains* ___ to be completed by the model (see Appendix D for full prompts). In order to reduce the inference cost, we used demonstrations that output the state of all boxes at once. However, in early experiments, Flan-T5 frequently only output the state of the first box even when the in-context demonstrations contained descriptions of all box states. For this reason, for Flan-T5, we probed each box individually.[8]

Demonstration/Test Mismatch for Form-meaning Disentanglement (AltForms) As discussed in Sections 3.1 and 3.2, we additionally experimented with a setup where the demonstration and test examples were mismatched in the form of the description of the world state and operations. Under this setup, the models were evaluated on descriptions where the names of the objects, the description of the initial state, and the phrasing of the core operations are all different from those in the demonstration examples (see Table 2). Except for the determiner "the" and the preposition "into", the two sets share no words (although subwords may be shared depending on tokenization).

4.3 Evaluation

We estimated the entity tracking capacity of the models by computing the accuracy of predicting the contents of each box after each operation. Given that we rely on arbitrary-length cloze completion to predict the contents, we had to score unconstrained generations. While we only considered instances as correct where the generated output mentions all objects (and no additional objects) in a given box, our evaluation setup did give some leeway to the exact form of the response. That is, we allowed the objects to appear in any order, the object mentions may be separated by commas or *and*, and we only considered the nouns in the object mentions, so it did not matter whether the model outputs a complete noun phrase or only the bare noun.

The task becomes intrinsically harder as more operations are applied to a specific box, since the initial state description needs to be combined sequentially with all subsequent operations. Further, similarly to our observation in Section 2, every operation changes the state of at most two boxes. This implies that the number of datapoints corresponding to fewer operations is much greater than the number of datapoints for more operations. For these reasons, we report accuracy as a function of the number of operations affecting a particular box rather than reporting aggregate accuracy, and show all results with 95% confidence intervals.

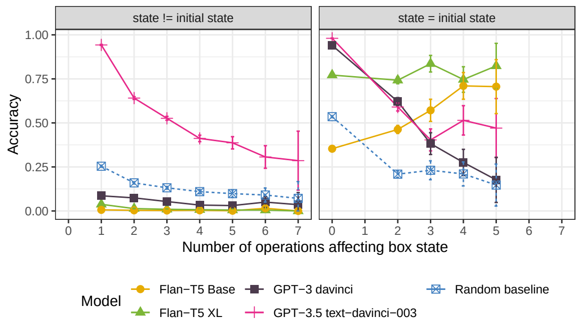


Figure 2: Accuracy on state prediction after n operations that affect a specific box. Left: predictions for boxes whose content differs from the initial state, Right: predictions for boxes whose content is the same as in the initial state. Error bars show 95% CIs.

4.4 Results

Figure 2 shows the prediction accuracy for different number of operations that acted upon a box (e.g., 3: three operations changed the content of the box after the initial state). The left panel shows the instances where the probed state differed from the initial state; the right panel shows the instances where the probed state were the same as the initial state. As the left panel shows, only GPT-3.5 `text-davinci-003` consistently outperformed the (strong) random baseline. While, not surprisingly, the accuracy of this model also decreases as the number of operations increases, the model still correctly predicted all contents of a box after 7 operations in more than 25% of the cases. The Flan-T5 models, on the other hand, primarily output the initial state description and seem to ignore the operations, as indicated by the consistently high accuracy on predicting the state when the probed state equals the initial state (right panel), as well as the consistently low accuracy on cases where the final state deviates from the initial state (left panel). GPT-3 `davinci` also successfully repeated the initial state, but as seen in the decreasing curve in the right panel, seemed to be distracted by a larger number of intervening operations.

Form-meaning Disentanglement We addition-

[8]To verify that this difference in the task format does not underestimate the accuracy of the GPT models, we also conducted an experiment in which we prompted GPT to only output the state of one box at a time. We found that, contrary to the Flan-T5 models, the accuracy of GPT was *lower* when we prompted it to output individual boxes than when it output the contents of all boxes, so we can rule out that this difference in the task format is underestimating GPT's performance.

Operation	Base	AltForm
`Move`	*Move the car from Box 1 to Box 3.*	*Pick up the furby in Container A and place it into Container C.*
`Remove`	*Remove the car from Box 1.*	*Take the furby out of Container A.*
`Put`	*Put the car into Box 1.*	*Place the furby inside Container A.*

Table 2: Different phrasings of the state-changing operations under the AltForms evaluation setup.

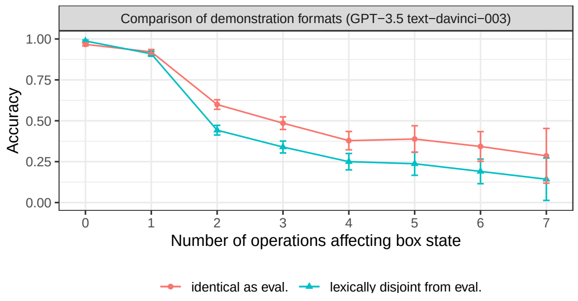


Figure 3: Entity tracking accuracy of `text-davinci-003` with low lexical overlap between demonstration and test examples (AltForms).

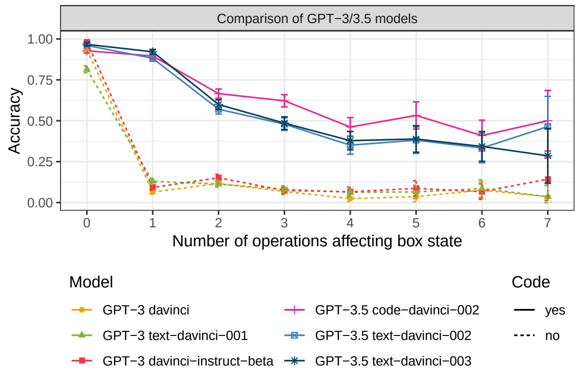


Figure 4: Accuracy on state prediction for different GPT-3 models. Solid lines denote models trained on code and text, and dotted lines denote models mainly trained on text.

ally evaluated `text-davinci-003`, the only model that exhibited a non-trivial entity tracking capacity in the first set of results, under the AltForms setup where the demonstration examples have low lexical overlap with the test examples. Figure 3 shows the prediction accuracy of `text-davinci-003` on a representative subsample[9] of our data. The blue line represents the performance when the descriptions in the demonstration and the test examples are disjoint as described in 4.2. As the comparison to the original results (red line) shows, the lexically disjoint demonstrations did lead to a small drop in performance when there were more than two operations that acted upon a box. Nevertheless, `text-davinci-003` was able to predict the correct state of entities in many cases, further adding support for its non-trivial entity tracking capacity.

4.5 Discussion

Our results show that among the models we evaluated, only GPT-3.5 `text-davinci-003` exhibit non-trivial entity tracking behavior. While its performance does decrease as the number of operations increases, the model still produced many accurate predictions even after six or seven relevant operations. Furthermore, through the experiment where the description of the initial state and operations differed in form in the demonstration examples, we also ruled out the possibility that these two examples are teaching the model this task or that the model is relying on superficial slot-filling heuristics. Therefore, we conclude that `text-davinci-003` does have some capacity to track discourse entities.

On the other hand, entity tracking behavior did not surface in the GPT-3 `davinci` (likely of similar size as GPT-3.5 `text-davinci-003`), a model pretrained primarily on text corpora on the next word prediction objective. This was also true for denoising models that have been finetuned on many tasks combined with instructions and demonstrations: the Flan-T5 models also showed near-zero accuracy on non-trivial examples.

These results show that there exists a language model that can perform entity tracking to some degree, but this capacity does not necessarily surface in all sufficiently large models trained on large corpora. Then, which factors are responsible for this difference? Given that `davinci` and `text-davinci-003` differ along at least two dimensions (`text-davinci-003` is based on a model that was trained on code and it was trained with additional human feedback (Ouyang et al., 2022); see Table 1), our initial results do not shed light on what exactly contributes to this difference. We therefore conducted a follow-up experiment where we compared a range of GPT-3 and GPT-3.5 models to identify a factor that contributes to the stark differ-

[9]For each number of operations n affecting a box, we sampled 100 states with at least one example with n operations.

ence between davinci and text-davinci-003.[10]

Training on Code Encourages Entity Tracking Behavior As Table 1 shows, two key dimensions of variation across models are additional training on human feedback and pretraining on code. If additional training on human feedback imbues language models with the ability to track entities, all models except for GPT-3 davinci and GPT-3.5 code-davinci-002 should be able to track entities. If, on the other hand, pretraining on code leads to better entity tracking, we expect all GPT-3.5 models to outperform GPT-3 on our task. As Figure 4 shows, GPT-3.5 models that have been trained on code systematically outperformed GPT-3 models, including code-davinci-002 that was not trained on human feedback. This suggests that a substantial representation of code in the pretraining data is beneficial for a language model's entity tracking capacity to surface.

A further question that our results so far do not answer is to what extent model size matters and whether models at the scale of Flan-T5 can also exhibit non-trivial entity tracking behavior. Since there exist no smaller models that have been trained with the same objective and training data as the GPT-3.5 models, we explore this question through finetuning experiments with T5.

5 Experiment 2: Finetuning

We investigated whether smaller models at the scale of T5 can *learn* to track entity states through a series of experiments where we provide supervised training to the models.

5.1 Train/test splits

As discussed in Section 3.1, one challenge of evaluating entity tracking abilities is distinguishing this capacity from simple heuristics such as templatic slot-filling. We therefore designed various types of training/evaluation mismatches that block several possible shortcuts as described below.

Base Split In the base split, we used the same format for training and evaluation examples. All initial states differed across training and evaluation to block simple slot-filling heuristics, as discussed in Section 3.2.

NumOps Split The NumOps split restricts the maximum number of operations within a single example in the training set to 2, but includes up to 12 operations in the evaluation set. This split was intended to test whether a finetuned model is able to generalize to longer sequences of operations than it has seen during finetuning.

Vocab Split The vocab split tests whether objects that are not part of the set of objects used during training can also be adequately tracked. We compiled a list of comparatively infrequent object names (e.g., *pomelo, furby, Flav-R-Straw*; not in BNC) and sampled the training and test sets using two completely disjoint sets of object names. The training set used the infrequent object list and the test set used the original object list.

AltForms Split This is a split identical in design to the disjoint demonstration/test setting described in Section 4.2, aiming to test whether the model learns to associate specific words/phrases with the operations or whether finetuning leads to more generalizable entity tracking behavior. We also created another split that combines the properties of this split with the "NumOps" split.

5.2 Models

We evaluated T5-base, the best performing model in Li et al. (2021), by finetuning it on each of the dataset splits described above. See Appendix C for details about the implementation. As an additional baseline, we compared against the T5 model with randomly initialized parameters.

5.3 Results and Discussion

Pretrained T5 can Learn to Perform Entity Tracking As shown in Figure 5 (left), finetuning T5 leads to near-perfect accuracy on the base split. This suggests that the model is capable of learning this task. Training a randomly initialized T5 did not yield the same result: the accuracy of a model trained from random weights is considerably lower, due to the model almost exclusively predicting that a box is empty. These two results suggest that pretraining is crucial for the model to be able to learn this task. Furthermore, the model's entity tracking capacity is robust to novel object names at test time, with only minor degradation on accuracy (Figure 5, middle). Training only on operation sequences with a maximum length of 2 (NumOps split) leads to a larger degradation in performance for longer operation sequences, but even for longer operation sequences, the model is able to infer the correct final state in more than 45% of the cases. Finally, the model performance

[10]To limit inference costs, we used the same subsample of data as in the AltForms experiment.

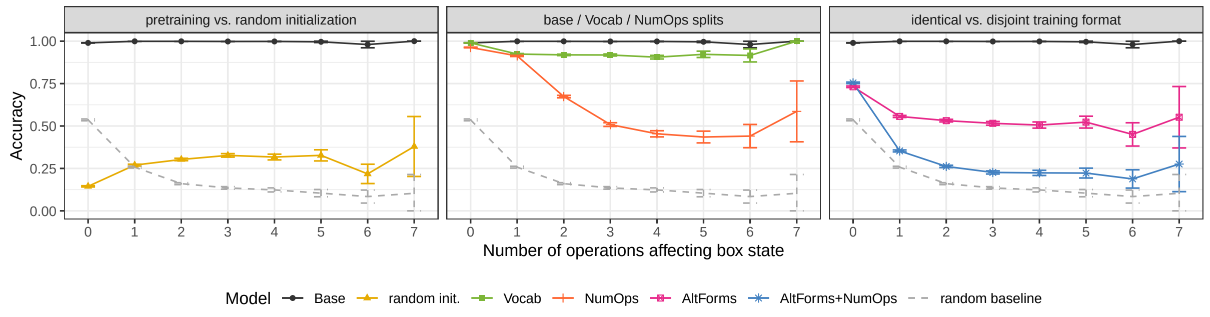


Figure 5: Results for finetuned T5 models.

does degrade substantially when the training examples have low lexical overlap with test examples (Figure 5, right). Nevertheless, the model predicts many non-trivial examples correctly, with ∼50% tracking accuracy after 1+ operations. The performance degradation was compounded if we trained only on up to two operations (blue line), but the performance remained above chance. These results suggest that finetuning on an entity tracking task does lead to entity tracking abilities that generalize to many challenging scenarios.

6 General Discussion

We set out to investigate whether pretrained language models exhibit entity tracking behavior. We developed a task that allowed us to evaluate whether language models can predict the state of an entity based on an initial state description and operations that act upon it. In the first set of experiments, we found that GPT-3 `davinci`, a vanilla pretrained language model, and Flan-T5, an instruction-finetuned language model, completely fail at this task and simply repeat the initial state description. The GPT-3.5 models (the core difference with the aforementioned models being code in pretraining corpora), on the other hand, exhibited non-trivial entity tracking behavior. Even after many operations affecting an entity state, they consistently performed above a strong random baseline.In the second set of experiments, we showed that this behavior can also be learned by much smaller models such as T5. When we finetune the model on this task, it is also able to perform entity tracking on examples that differ along several dimensions from the training data. Taken together, our results provide strong evidence that (a) vanilla language models that have mainly been trained on text data do not exhibit entity tracking abilities out of the box, (b) pretraining on code and text data considerably improves this ability, and (c) finetuning on this task can make this behavior also surface in smaller models that have primarily been trained on text data.

What are reasons behind the efficacy of training on both code and text? For producing executable and correct code, keeping track of the states of variables is important. Therefore, to speculate, this kind of pretraining data may provide a stronger signal for the model to track entities compared to pure text data. It could also be that, as speculated by Potts (2020) and Merrill et al. (2021), *i.a.*, pretraining on code provides additional grounding, which improves models' semantic and pragmatic abilities.

The present results also highlight the importance of transparency in documenting the factors involved in the pretraining procedure. Our results add support to the claim of Bender and Koller (2020) that there are aspects about understanding, such as entity state tracking, that are inherently challenging for vanilla LMs. At the same time, our results also suggest that additional signal in the form of code, which is common in the latest versions of LMs, does make state tracking ability surface. This highlights that arguments that may be valid for vanilla LMs are no longer necessarily valid for newer models that are also referred to as LMs.

Apart from these empirical results, we laid out several principles that should be followed when evaluating state tracking abilities. Apart from these specific principles, we make a more general point that in assessing abilities related to meaning, one needs to consider potential strategies that the model could use to solve the task and make sure that the test examples do not mimic the distributional patterns of the training data. Furthermore, any supervision provided for evaluating the model, should not allow the model to learn associations between specific forms and meaning. Only then can we properly assess meaning-related capacities of LMs.

Limitations

One limitation of this work is that we are only considering behavioral data which makes it difficult to establish a fully causal link between entity tracking capacities and high performance on our task. Entity tracking is a high-level linguistic behavior and many other capacities are necessary for achieving high accuracy on our task. Therefore, we cannot rule out that differences in some other capacity, such as interpreting sentences compositionally (see Bogin 2022 and Bogin et al. 2022 for evidence that GPT-3 and GPT-3.5 models differ in their compositional generalization behavior), are the main driver for the differences in behavior we see across models.

A further limitation of our setup is that it requires short-term memory capacities that exceed the memory capacities of most, if not all, humans. That is, if we presented humans with the same input as the model, we would not expect them to be able to keep track of the contents of all 7 boxes due to memory limitations. Therefore we are potentially expecting models to do super-human entity tracking, a setup that has been criticized for model evaluations of other linguistic abilities (Lampinen, 2022). We nevertheless believe that our task is justified given the architecture of the evaluated models. Transformer-based models can look back to any token in the entire input sequence within their context window, so a proper comparison between humans and models would be to present humans with the full description in written form and let them re-read the description after being prompted to state the contents of a box. While we did not formally evaluate whether humans have this ability on a larger population, we personally did not have any trouble tracking the contents of boxes when we had access to the written description.

Relatedly, we designed our task such that the entire description fits within the context window of pretrained language models. However, as we mentioned in the introduction, entity tracking is an important ability for understanding long contexts and given the limited context window, our results do not apply to texts whose length exceeds a model's context window, and likely different model architectures will be necessary to perform proper entity tracking for longer texts.

Further, while we found that the GPT-3.5 models as well as the finetuned T5 models can track entities in our task with higher accuracy than a strong random baseline, our results also indicate that this behavior is not very stable once several operations act on an entity. Our results should therefore not be taken as justification for using these models for critical applications where much higher accuracy is needed.

Lastly, we only evaluated English models in this work. Given that we showed that even without high lexical overlap between the training and evaluation examples, models can keep track of entities to some extent, it seems likely that our results also apply to other languages but whether this actually the case, remains an open question.

References

Laura Aina, Carina Silberer, Ionut-Teodor Sorodoc, Matthijs Westera, and Gemma Boleda. 2019. What do entity-centric models learn? insights from entity linking in multi-party dialogue. 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 3772–3783, Minneapolis, Minnesota. Association for Computational Linguistics.

David Bamman, Olivia Lewke, and Anya Mansoor. 2020. An annotated dataset of coreference in English literature. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 44–54, Marseille, France. European Language Resources Association.

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.

Ben Bogin. 2022. How does GPT-3 perform on COVR-10 splits with in-context learning? Twitter thread.

Ben Bogin, Shivanshu Gupta, and Jonathan Berant. 2022. Unobserved local structures make compositional generalization hard. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP 2022)*, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2018. Simulating action dynamics with neural process networks. In *International Conference on Learning Representations (ICLR 2018)*.

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.

Hong Chen, Zhenhua Fan, Hao Lu, Alan Yuille, and Shu Rong. 2018. PreCo: A large-scale dataset in preschool vocabulary for coreference resolution. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 172–181, Brussels, Belgium. Association for Computational Linguistics.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. arXiv:2210.11416.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2019. Textworld: A learning environment for text-based games. In *Computer Games*, pages 41–75, Cham. Springer International Publishing.

Bhavana Dalvi, Niket Tandon, Antoine Bosselut, Wen-tau Yih, and Peter Clark. 2019. Everything happens for a reason: Discovering the purpose of actions in procedural text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4496–4505, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jeroen Groenendijk and Martin Stokhof. 1991. Dynamic predicate logic. *Linguistics and Philosophy*, 14:39–100.

Aditya Gupta and Greg Durrett. 2019a. Effective use of transformer networks for entity tracking. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 759–769, Hong Kong, China. Association for Computational Linguistics.

Aditya Gupta and Greg Durrett. 2019b. Tracking discrete and continuous entity state for process understanding. In *Proceedings of the Third Workshop on Structured Prediction for NLP*, pages 7–12, Minneapolis, Minnesota. Association for Computational Linguistics.

Aria Haghighi and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 385–393, Los Angeles, California. Association for Computational Linguistics.

Irene Heim. 2002. File change semantics and the familiarity theory of definiteness. In *Formal semantics: The essential readings*, chapter 9, pages 223–248. Blackwell Publishing Oxford.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2017. Tracking the world state with recurrent entity networks. In *International Conference on Learning Representations (ICLR 2017)*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The Goldilocks principle: Reading children's books with explicit memory representations. In *International Conference on Learning Representations (ICLR 2016)*.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A. Smith. 2017. Dynamic entity representations in neural language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1830–1839, Copenhagen, Denmark. Association for Computational Linguistics.

Hans Kamp, Josef Van Genabith, and Uwe Reyle. 2011. *Discourse Representation Theory*, pages 125–394. Springer Netherlands, Dordrecht.

Lauri Karttunen. 1976. Discourse referents. In J. D. McCawley, editor, *Syntax and Semantics Vol. 7*, pages 363–386. Academic Press.

Chloé Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer, and Yejin Choi. 2015. Mise en place: Unsupervised interpretation of instructional recipes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 982–992, Lisbon, Portugal. Association for Computational Linguistics.

Andrew Kyle Lampinen. 2022. Can language models handle recursively nested grammatical structures? a case study on comparing models and humans. arXiv:2210.15303.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 28–34, Portland, Oregon, USA. Association for Computational Linguistics.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Geoffrey Leech, Paul. Rayson, and Andrew Wilson. 2001. *Word frequencies in written and spoken English: based on the British National Corpus*. Longman.

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.

Tal Linzen. 2020. How can we accelerate progress towards human-like linguistic generalization? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5210–5217, Online. Association for Computational Linguistics.

Sharid Loáiciga, Anne Beyer, and David Schlangen. 2022. New or old? exploring how pre-trained language models represent discourse entities. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 875–886, Gyeongju, Republic of Korea. International Committee on 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.

William Merrill, Yoav Goldberg, Roy Schwartz, and Noah A. Smith. 2021. Provable limitations of acquiring meaning from ungrounded form: What will future language models understand? *Transactions of the Association for Computational Linguistics*, 9:1047–1060.

Mante S. Nieuwland and Jos J. A. Van Berkum. 2006. When peanuts fall in love: N400 evidence for the power of discourse. *Journal of Cognitive Neuroscience*, 18(7):1098–1111.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. arXiv:2203.02155.

Christopher Potts. 2020. Is it possible for language models to achieve language understanding? Medium blog post.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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*, 21:1–67.

Altaf Rahman and Vincent Ng. 2012. Resolving complex cases of definite pronouns: The Winograd schema challenge. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 777–789, Jeju Island, Korea. Association for Computational Linguistics.

Sebastian Schuster and Tal Linzen. 2022. When a sentence does not introduce a discourse entity, transformer-based models still sometimes refer to it. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 969–982, Seattle, United States. Association for Computational Linguistics.

Ionut-Teodor Sorodoc, Kristina Gulordava, and Gemma Boleda. 2020. Probing for referential information in language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4177–4189, Online. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Shubham Toshniwal, Sam Wiseman, Allyson Ettinger, Karen Livescu, and Kevin Gimpel. 2020. Learning to Ignore: Long Document Coreference with Bounded Memory Neural Networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8519–8526, Online. 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 AAAI Conference on Artificial Intelligence*, volume 36, pages 11385–11393.

Olga Uryupina, Ron Artstein, Antonella Bristot, Federica Cavicchio, Francesca Delogu, Kepa J Rodriguez, and Massimo Poesio. 2020. Annotating a broad range of anaphoric phenomena, in a variety of genres: the arrau corpus. *Natural Language Engineering*, 26(1):95–128.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. ACE 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia: LDC2006T06*.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart Van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards AI-complete question answering: a set of prerequisite toy tasks. arXiv:1502.05698.

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.

Wei Wu, Fei Wang, Arianna Yuan, Fei Wu, and Jiwei Li. 2020. CorefQA: Coreference resolution as query-based span prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6953–6963, Online. Association for Computational Linguistics.

A Additional Analyses of Results from Li et al. (2021)

As mentioned in Footnote 4, Li et al. (2021) conducted two more experiments that prima facie provided additional evidence for implicit meaning representations and state tracking abilities in language models. However, the data and setup of these two experiments also likely overestimates models' abilities.

In the second set of probing classifier experiments, Li et al. (2021) used data generated using the TextWorld engine (Côté et al., 2019). In this setup, there is a textual description of several entities in a text-based game (e.g., *a wooden door*) as well as actions that a player took (e.g., *opening the wooden door*). Their probing classifier takes the representations of either one entity and a property of that entity (e.g., that *the wooden door is **closed***) or the representations of two entities and a relation between them (e.g., that the *king-sized bed **is in** the bedroom*) and from these representations, the classifier has to predict whether a given proposition is true or false considering the initial description and the series of actions that a player took. Only propositions that involve entities that have been mentioned are probed. The issue with this setup is that there are many propositions that are always true in both the training and evaluation splits (e.g., in all the game simulations, the chest drawer is in the bedroom, so the probing classifier should always return true for this input independent of the previous context). Furthermore, even for the entity-property propositions and entity-relation-entity propositions which are not always true in the training and evaluation data, the data is very biased and a baseline that predicts the most common answer in the training data without taking the initial descriptions or the user actions into account (a violation of Desideratum 3, see Section 3.1), already achieves an accuracy of 88.5%, a number that puts the reported probing classifier accuracy of 96.9% into a bit more context.

Further, Li et al. (2021), also presented an experiment where they manipulated specific entity representations of a synthetic version of the Alchemy dataset. In this experiment, they first encoded an initial description D and an operation O which affected a beaker X using a language model that had been finetuned to predict the next operation, resulting in representation R_1. Then, they encoded the same initial representation D and an operation of the form *Drain* n *from beaker* Y, where beaker Y was always different from beaker X and n was the amount of liquid that was in beaker Y according to the initial description. This resulted in representation R_2. Then, they extracted the representation of the initial description of beaker Y from representation R_2 and replaced the representation of beaker Y in R_1 with the corresponding representation in R_2 to obtain R_{mixed}. They then used R_{mixed} as an input to T5 and showed that the predicted next operation based on R_{mixed} was considerably more often compatible with both operations (the operation encoded in R_1 and the operation encoded in R_2) compared to predicting the next operation from R_1 or R_2, which they took as evidence that the token representations of the initial description encoded the entities state after performing the operation. The issue with this experiment is that the operation from which R_2 was computed was always of the form *Drain* n *from* Y *th beaker*, so the final state of beaker Y was always empty. Therefore, this experiment primarily shows that the token *drain* affected the representation of the initial state (and subsequently the prediction of the next operation) but not more generally, that the actual state of the manipulated beaker Y is fully encoded in its initial state description. To answer that question, one would have to repeat this experiment with more complex operations that do not give away the final state (a violation of Desideratum 2, see Section 3.1).

In summary, the additional experiments in Li et al. (2021) also violate some of the desiderata we laid out in Section 3.1, and therefore it is difficult to draw conclusions about state tracking abilities from these experiments.

B Example Input-Output Pair

(3) shows an example input-output pair from our dataset (NumOps on Box 6 = 2).

(3) a. INPUT: *Box 0 contains the painting, Box 1 contains the bell, Box 2 contains the guitar, Box 3 contains the egg and the mirror and the sheet, Box 4 contains the chemical, Box 5 contains the disk and the wire, Box 6 contains the glass and the knife. Move the glass from Box 6 to Box 4. Put the gift into Box 5. Move the guitar from Box 2 to Box 6. Put the milk into Box 4. Remove the mirror and the sheet from Box 3. Box 6* __

b. OUTPUT: *contains the guitar and the knife*

C Implementation Details

In-context For GPT-3, we used the OpenAI API. We used greedy decoding with the temperature parameter set to 0, used a maximum target generation length of 150, and used the newline character as an additional stop token. Inference time and model size is not available for GPT-3, but we generated about 16 million tokens in total. All GPT-3 experiments that we report here in the paper were conducted in January 2023.

For Flan-T5, we used a beam size of 3 and a maximum target generation length of 256. Other hyperparameters were kept as the default values of `T5ConditionalGeneration` in the HuggingFace library. Inference for T5-XL took about 5 hours on a single A100 GPU, and for T5 base, about 2 hours.

Finetuning We finetuned T5 for a single epoch using a batch size of 8 and a learning rate of 1×10^{-4}. In initial explorations, increasing the number of finetuning epochs did not yield substantial gains on development set performance, and was sometimes even harmful. Training and inference took around 3 hours on a single RTX8000 GPU.

D Prompts

Table 3 shows the 2-shot prompts used for in-context experiments. Table 4 shows the 2-shot prompts used for the AltForms experiments, where the demonstrations contain descriptions that have lower lexical overlap with the test examples.

E Dataset Statistics and License Information

See Tables 5 and 6 for descriptive statistics of our dataset. The dataset will be released under GNU General Public License v3.0.

2-shot prompt with all boxes queried at once (GPT-3 experiments)

Given the description after "Description:", write a true statement about all boxes and their contents to the description after "Statement:".

Description: Box 0 contains the car, Box 1 contains the cross, Box 2 contains the bag and the machine, Box 3 contains the paper and the string, Box 4 contains the bill, Box 5 contains the apple and the cash and the glass, Box 6 contains the bottle and the map.
Statement: Box 0 contains the car, Box 1 contains the cross, Box 2 contains the bag and the machine, Box 3 contains the paper and the string, Box 4 contains the bill, Box 5 contains the apple and the cash and the glass, Box 6 contains the bottle and the map.

Description: Box 0 contains the car, Box 1 contains the cross, Box 2 contains the bag and the machine, Box 3 contains the paper and the string, Box 4 contains the bill, Box 5 contains the apple and the cash and the glass, Box 6 contains the bottle and the map. Remove the car from Box 0. Remove the paper and the string from Box 3. Put the plane into Box 0. Move the map from Box 6 to Box 2. Remove the bill from Box 4. Put the coat into Box 3.
Statement: Box 0 contains the plane, Box 1 contains the cross, Box 2 contains the bag and the machine and the map, Box 3 contains the coat, Box 4 contains nothing, Box 5 contains the apple and the cash and the glass, Box 6 contains the bottle.

Description: `{description}`
Statement: Box 0 contains

2-shot prompt with boxes queried individually (T5 experiments)

Given the description after "Description:", write a true statement about a box and the contents of this box according to the description after "Statement:".

Description: Box 0 contains the car, Box 1 contains the cross, Box 2 contains the bag and the machine, Box 3 contains the paper and the string, Box 4 contains the bill, Box 5 contains the apple and the cash and the glass, Box 6 contains the bottle and the map.
Statement: Box 1 contains the cross.

Description: Box 0 contains the car, Box 1 contains the cross, Box 2 contains the bag and the machine, Box 3 contains the paper and the string, Box 4 contains the bill, Box 5 contains the apple and the cash and the glass, Box 6 contains the bottle and the map. Remove the car from Box 0. Remove the paper and the string from Box 3. Put the plane into Box 0. Move the map from Box 6 to Box 2. Remove the bill from Box 4. Put the coat into Box 3.
Statement: Box 2 contains the bag and the machine and the map.

Description: `{description}`
Statement: Box `{boxnum}` contains

Table 3: Prompts with 2-shot in-context demonstrations.

2-shot prompt with disjoint surface forms from test examples
Given the description after "Description:", write a true statement about all containers or boxes and their contents to the description after "Statement:". Description: The biscotti is in Container A, the icicle is in Container B, the granite and the machine are in Container C, the folio and the encyclopedia are in Container D, the bill is in Container E, the spork and the jackknife and the frappuccino are in Container F, the clipper and the ladybug are in Container G. Statement: Container A contains the biscotti, Container B contains the icicle, Container C contains the granite and the machine, Container D contains the folio and the encyclopedia, Container E contains the bill, Container F contains the spork and the jackknife and the frappuccino, Container G contains the clipper and the ladybug. Description: The biscotti is in Container A, the icicle is in Container B, the granite and the machine are in Container C, the folio and the encyclopedia are in Container D, the bill is in Container E, the spork and the jackknife and the frappuccino are in Container F, the clipper and the ladybug are in Container G. Take the biscotti out of Container A. Take the folio and the encyclopedia out of container D. Place the tetrapod inside Container A. Pick up the ladybug in Container G and place it into Container C. Take the bill out of Container E. Place the gumball inside Container D. Statement: Container A contains the tetrapod, Container B contains the icicle, Container C contains the granite and the machine and the ladybug, Container D contains the gumball, Container E contains nothing, Container F contains the spork and the jackknife and the frappuccino, Container G contains the clipper. Description: `{description}` Statement: Box 0 contains

Table 4: Prompts with 2-shot in-context demonstrations that have disjoint surface forms.

	Scenarios		Examples	
Dataset	**Demonstration**	**Test**	**Demonstration**	**Test**
Complete	1	990	14	90,090
Subsample	1	491	14	5,012

Table 5: Dataset statistics for Experiment 1.

	Scenarios			Examples		
Dataset	**Train**	**Dev**	**Test**	**Train**	**Dev**	**Test**
Base	990	220	990	90,090	20,020	90,090
NumOps	990	220	990	20,790	20,020	90,090
Vocab	990	220	990	90,090	20,020	90,090
AltForm	990	220	990	90,090	20,020	90,090
AltForm+NumOps	990	220	990	20,790	20,020	90,090

Table 6: Dataset statistics for Experiment 2.