A Data Source for Reasoning Embodied Agents

Jack Lanchantin, Sainbayar Sukhbaatar, Gabriel Synnaeve, Yuxuan Sun, Kavya Srinet, Arthur Szlam

Meta AI

{jacklanchantin, sainbar, gab, yuxuans, ksrinet, aszlam}@meta.com

Abstract

Recent progress in using machine learning models for reasoning tasks has been driven by novel model architectures, large-scale pre-training protocols, and dedicated reasoning datasets for fine-tuning. In this work, to further pursue these advances, we introduce a new data generator for machine reasoning that integrates with an embodied agent. The generated data consists of templated text queries and answers, matched with world-states encoded into a database. The world-states are a result of both world dynamics and the actions of the agent. We show the results of several baseline models on instantiations of train sets. These include pre-trained language models fine-tuned on a text-formatted representation of the database, and graph-structured Transformers operating on a knowledge-graph representation of the database. We find that these models can answer some questions about the world-state, but struggle with others. These results hint at new research directions in designing neural reasoning models and database representations. Code to generate the data and train the models will be released at github.com/facebookresearch/neuralmemory.

Introduction

Advances in machine learning (ML) architectures (Vaswani et al. 2017), large datasets and model scaling for pre-training (Radford et al. 2019; Chowdhery et al. 2022; Zhang et al. 2022), modeling approaches (Nye et al. 2021; Wang et al. 2022), and multiple dedicated reasoning datasets (Yang et al. 2018a; Hudson and Manning 2019; Petroni et al. 2020; Vedantam et al. 2021) have driven progress in both building models that can succeed in aspects of "reasoning" and in automatically evaluating such capabilities. This has been evident particularly in the text setting, but also in computer vision (Krishna et al. 2016; Johnson et al. 2017; Yi et al. 2020).

In parallel, the last decade has seen advances in the ability to train embodied agents to perform tasks and affect change in their environments. These have been also powered in part by data, with many environments made available for exploring modeling approaches and benchmarking. In particular, with respect to "reasoning" in embodied agents, there have been works showing that adding inductive biases to support reasoning can lead to improved performance with end-to-end

Sequence Context: "agent is at location (10,7,8). agent has pitch 1.0 and yaw 2.5. agent has property agent. speaker is at location (8,7,0). speaker has pitch 0.8 and yaw 2.3. wilson is at location (0,7,9). wilson has pitch -0.8 and yaw 1.1. wilson has property rabbit. wilson has property brown. marbles is at location (10,7,3). marbles has pitch 1.1 and yaw 3.1. marbles has property chicken. marbles has property white. pebbles is at location (10,10,13). pebbles has pitch 0.5 and yaw -0.3. pebbles has property sheep. pebbles has property mottled. gus is at location (3,7,9). gus has pitch 0.4 and yaw -0.1. gus has property cow. gus has property black. inst_seg is at location (3,8,1). inst_seg has property arch. inst_seg is at location (10,8,12). inst_seg has property dome. inst_seg is at location (14,6,6). inst_seg has property hole."

Query: "what is closest from speaker?"
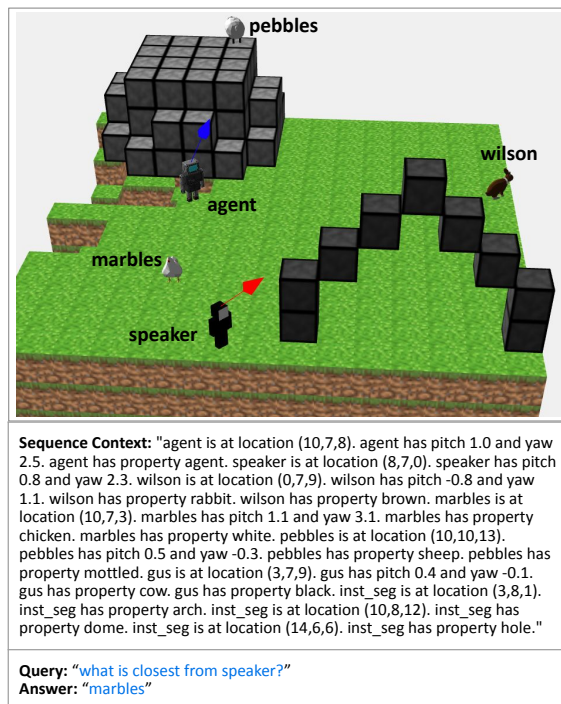**Answer:** "marbles"

Figure 1: (Top): Example of a generated scene in our 3d gridworld. (Middle): Given the 3d scene, we can convert the information in the render to a text or structured representation. Here we show the text Sequence Context representation. "inst_segs" represent the block items such as structures or holes. (Bottom): For a particular scene, we can generate a wide variety of queries. Here we show and example of a distance query asking which object is the closest to the speaker, which is the chicken named "marbles".

training (Zambaldi et al. 2018) and other works have shown how models can be augmented with extra supervision (Zhong, Rocktäschel, and Grefenstette 2019).

Recently, several works have shown how large language-model pre-training can be used to affect planners for embodied agents (Huang et al. 2022; Ahn et al. 2022). More generally, symbolic representations can be a hub for connecting perception, memory, and reasoning in embodied agents.

However, the growing literature in NLP reasoning models

is missing data grounded in a dynamic and agent-alterable world. Models trained on traditional text datasets struggle to handle physically grounded queries such as those that involve geometric reasoning. In other words, recent large language models trained on internet data are not well equipped to simple questions about a physical environment such as "who is to my left?". Grounding large-language models may allow them more powerful reasoning; and vice versa, may help us use them as agent controllers.

In this work we describe a data source (i.e., a toolbox to generate data) designed to help train ML models grounded in a physical environment, allowing them to make the connection between perception, memory, and reasoning. It consists of context-question-answer triples, where the context corresponds to a dynamic and agent-affected 3d gridworld, and the questions may involve temporal or spatial reasoning, as well as questions about the agent's own actions. A sample generated from our data source is shown in Fig. 1.

While the environment allows rendering the world-context as a sequence of images, one of our goals is to support research toward answering the question "what are good formats for agent memory systems?". In pursuit of this, we abstract the context to a database format that does not require any perceptual modules, and provide code for converting the database into a templated text dump, as demonstrated in Fig. 1 (right top). Here, the order of the facts within each timestep are written randomly, and sequentially written according to the timesteps. Our hope is that the data source can be used for augmenting the training (or allowing the assembly) of reasoning embodied agents by bringing to bear the advances in reasoning in language models, or as a supplement to training language models with grounding from embodied agents.

We train baseline neural models to represent the database and process the queries. These include finetuning pre-trained language models on the text version of the database, and Transformers that input the structured database directly. We find that while certain queries are easily solved by these baselines, others, in particular - those having to deal with spatial geometry, are more difficult.

In short, the contributions of this paper are:

Environment: We introduce an environment for embodied agents and a data source for generating data to train agents in this environment (detailed in the Environment, Queries, and Data section). We provide the code to generate world contexts, as well as complex queries. We hope this will aid researchers to isolate and tackle difficult problems for reasoning for embodied agents.

Baselines: We evaluate the abilities of baseline models to answer queries in this environment (Experiments section). We compare different representations of the world context, including a pure text based representation as well as a more structured representation.

Environment, Queries, and Data

We propose a context-question-answer data generator for embodied agents. In this section, we outline the context, or environment we generate data in, the types of queries we create for the agent to solve, and specifics of the data samples.

Environment

We work in a finite three-dimensional gridworld. There is a primary agent, zero or more other agents, zero or more get/fetchable items, zero or more placeable/breakable blocks. The other agents come in two types: they might represent human "players" that can give commands to the agent and have the same capabilities as the agent, or animate non-player characters (NPCs) that follow simple random movement patterns. The placeable/breakable blocks have colors and integer grid coordinates; all other objects have float coordinates. Animate objects (the players, NPCs, and the agent) have a yaw and pitch pose representing the location they are looking, in addition to three location coordinates.

To build a scene, we generate some random objects (spheres, cubes, etc.), randomly place a number of NPCs, a player, and an agent. With some probability, the agent executes a command (to either: build an object, destroy an object, move to a location, dig a hole, follow an NPC, etc.) The execution of a command is scripted; the task executor is from the Minecraft agent in (Pratik et al. 2021). Whether or not the agent executes a task, the world steps a fixed number of times (so, e.g., NPCs may move or act). In the experiments described below, the world is fully observed at a fixed number of temporal snapshots, and all poses, object locations and NPC movements are recorded. However, not every world step is snapshotted, so the total sequence is not fully observed.

Following (Pratik et al. 2021), the environment is presented to the agent as an object-centered key-value store. Each object, NPC, and the agent's self have a "memid" keying a data structure that depends on the object type, and may contain string data (for example a name) or float or integer data (e.g. the pose of an NPC). The key-value store also has subject-predicate-object triples (e.g. "memid has tag mottled"); the triples themselves also have a unique memid as key.

The generated scene presented as the key-value store defines the agent's context, C. In our experiments below, we represent this context in one of two ways. The first is a text sequence (C_t), where for each snapshot step, all objects and their properties in the key-value store are flattened into a templated language, as shown in Fig. 1 (right). Multiple time snapshots of the context are represented by appending each successive event in the sequence. While this is in some sense simplistic, it allows ready application of large pre-trained language models for ingesting the context; and this kind of approach has been shown to be effective in (Thorne et al. 2021; Liu et al. 2021).

Alternatively, we can leverage the relational properties of the context by representing it as a graph where objects and properties are nodes, and the connections between them are the edges. For example, if we know that "bob" is a horse and is the color brown, the node for bob connects to a "horse" and a "brown" object. Specifically, this "Sequence Context" contains reference object nodes (C_R), which are the object instantiations, and triple nodes (C_T), which are the properties of the reference objects. Each reference object node holds the following information: *reference_object_hash* (R_id) is a unique identifier for each reference object, *reference_objects_words* holds identifier words of the object such as its name, and *reference_objects_float* is the floating point properties of the
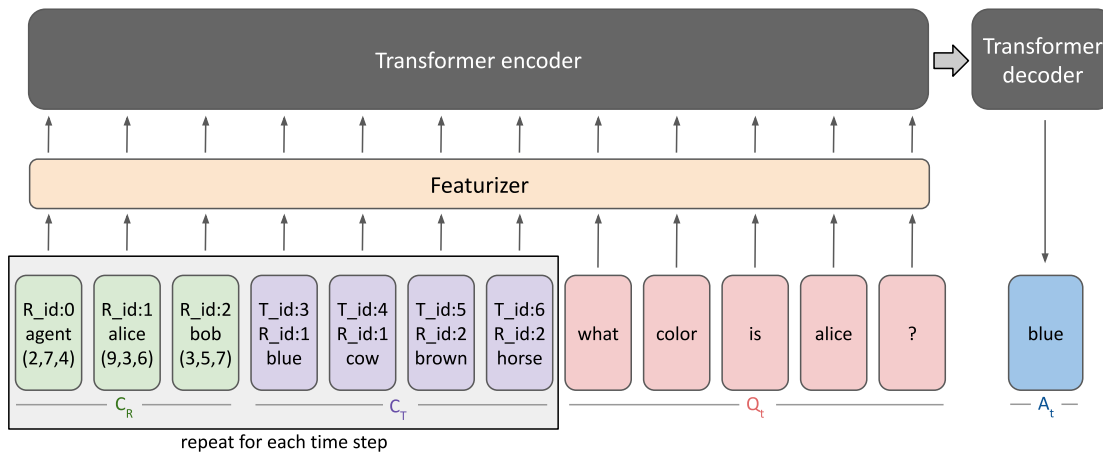
Figure 2: Structured context + Transformer model. The bottom left demonstrates the structured representation of the 3d gridworld, where R_id is the unique reference object identifier, and T_id is the unique triple property identifier (which connects to one of the reference objects via R_id). Context nodes ($C_{\{R,t\}}$) and query tokens (Q_t) are first featurized with learnable embedding layers. We process the featurized context and query jointly with a Transformer encoder that considers the context structure via relational embeddings (r_{ij}). Finally, a text decoder predicts tokens, and a memid decoder predicts relevant context memids (not pictured).

object such as its (x, y, z) coordinates and pitch/yaw. These are combined into a single node (detailed in the Models section). Similarly, each property triple is composed of the following elements: *triples_hash* (T_id) contains a unique identifier for the triple, as well as the reference object hash that it is linked to, and *triples_words* are a descriptive text of a reference object's property such as "has_color blue". These are also combined into a single node. We therefore have nodes of both reference objects and triples, and the hashes encompass the edges or relationships between them.

We do not consider the text sequence or graph-structured representations to be canonical. On the contrary, our goal is stimulate research into what the correct representation of world state should be to allow easier model training and transfer between agents. We simply provide these two representations as examples and baselines.

Given the determined set of snapshots, queries are designed to be answerable. That is, all information is known to answer the questions. We leave to future work making ambiguous queries, but note here that it is not difficult to purposefully build and record queries that could theoretically be answered in some scene but cannot be answered in the particular scene instantiation, for example because they refer to an event that occurred between snapshots. It would otherwise be easy to restrict full observability within each snapshot.

Queries

The embodied agent environment allows for a rich set of possible queries to ask the agent. We structure the types of queries we use into three main categories, as covered in Table 1. *Property* queries are those which operate on the current state of the memory such as the current properties or locations of an object, and are given with an explicit relation. Property queries with a single clause can be read directly from the database or text dump without any "reasoning". *Temporal* queries are those which operate over spans of the

memory such as the movement of object. *Geometric* queries are those concerned with the geometric structure of the environment such as how far away two object are from each other. Note that many queries are mixes of these types, and the categorization is blurred.

Within each query class, there are several different "clause" types. These can be combined as applicable into a multi-clause query, where the clauses are combined by an "and" or "or" conjunction randomly. In addition, each clause can be negated by prepending the word "not". For example, "what are the name of the objects that do not have the property brown and where the x coordinate is less than 4".

The query, Q, is composed of one or both of the following representations: *query_text* (Q_t): a text representation of the query (e.g. "find the reference_objects with color brown"), *query_tree_logical_form* (Q_{lf}) : a tree logical form representation of the query (e.g. two clauses in an "and" or "or" query are connected by a parent in the tree).

Given the context and query, the agent should return some answer, A. Depending on the query type, we randomly ask for one of the following answer types in the query: *name* ("what is the name of..."), *properties* ("what are the properties of..."), *location* ("what is the location of..."), *distance* ("how far is..."), and *count* ("how many..."). In general, we are interested in predicting the text answer such as "brown" for the query "what is the color of the horse?". However, we may also be interested in pointing to the relevant context objects or properties (in the structural context representation). Thus, for each query, we provide both the text answer as well as the relevant context node IDs (from C_R and C_T).

Data

With this data generation framework, we can create arbitrary amounts of simulated data. Each data sample contains a (C, Q, A) triple. There are several parameters of the world that affect the difficulty of question answering, including the size

Query Class	Clause types	Example
Property	name	what are the properties of the objects that have the name alice?
	tag	what are the names of the objects that has the property brown?
	absolute cardinal	what are the locations of the objects where the x coordinate is less than 4?
Temporal	cardinal	what is the name of the object that increased x the most?
	relative	what is the name of the object that moved to my left the most?
	farthest moved object	what is the name of the object that moved the farthest?
	location at time	what was the location of bob at the beginning?
	action	what did you do?
	object tracking	where would the ball be if i moved to (4,7,2)?
Geometric	absolute distance	what is the count of the objects where the distance to (2, 6, 5) is greater than 3?
	direction	what are the names of the objects to my right?
	closest object	what is the name of the object that is closest to the cow?
	max direction	what is the name of the object that is the most to my right?
	distance between	how far is the horse from you?
	distance from position	what is the location 3 steps to your right?

Table 1: Query clause types. We categorize the queries we can ask the agent into three separate classes. Within each class, there are several clause types. *Italicized* clauses cannot be combined with others.

of the 3d gridworld (e.g., 15x15x15), the set of possible objects in the world (e.g., bob, alice, ...), and the set of possible properties that can belong to the objects (e.g. cow, horse, blue, green, ...). The number of time-steps and number of snapshots also crucially affect the difficulty of the problem. Similarly, the distributions over the queries (e.g., how many and what kind of clauses) can make the problem more difficult.

Related Work

Real-world QA datasets have long been used to test different aspects of ML model performance such as reading comprehension (Rajpurkar et al. 2016; Hill et al. 2016), commonsense reasoning (Talmor et al. 2019), multi-hop reasoning (Yang et al. 2018b), and visual understanding (Agrawal et al. 2015; Hudson and Manning 2019). While real-world datasets can provide reliable performance benchmarks and better approximate the problems faced by practitioners, synthetic datasets can allow for more control and the ability to isolate the exact limitations of current models. Notably, bAbI (Weston et al. 2016) is a set of toy QA tasks testing various reasoning abilities over short text stories that showed the limitations recurrent neural networks. Since proposed, all bAbI tasks have been solved by novel memory architectures (Henaff et al. 2017; Dehghani et al. 2019). An gridworld environment for embodied agents with language instructions for tasks is described in (Chevalier-Boisvert et al. 2018); our work here is complementary, giving question-answer pairs based on abstracted environment histories.

CLEVR (Johnson et al. 2017) is a popular synthetic data for testing visual reasoning given text queries. (Yi et al. 2020) extends CLEVR to reasoning over temporal events in videos. Embodied question answering (EmbodiedQA) (Das et al. 2018) proposes a task where an agent must navigate an environment in order to answer a question. VideoNavQA (Cangea et al. 2019) was proposed in the EmbodiedQA domain to evaluate short video-question pairs. Our work has elements of each these. The agent is embodied, and might need to answer questions about its actions or hypotheticals, but does not need to act or change the current state of the environment to answer (as in EmbodiedQA). In comparison to the EmbodiedQA dataset where the agent has to query the environment to get more information, our setting doesn't require the agent to interact and get more information. As in (Yi et al. 2020), the agent needs to be able to reason over spatio-temporal events, in our case, including its own actions. As in CLEVR, we use programmatically generated queries to probe various reasoning modalities. One large difference between this work and those is that we do not focus on computer vision. While it is possible to render the scenes from our data generator, our goal is to be agnostic about perceptual modality and abstract away perceptual modeling, and to the extent possible, focus on the reasoning aspects of the data. Within the vision community, other works have approached the VQA problem from this angle (Yi et al. 2018).

Because the agent's abstracted world representation has a database-like structure, our work falls into the literature on ML for question answering on structured data, for example (Pasupat and Liang 2015). Our structured Transformer baseline is inspired by the literature on neural database representation, for example (Wang et al. 2019; Yin et al. 2020), and references therein. Our LM baseline is inspired by the many works that flatten or otherwise textify databases, and use pretrained language models as bases for neural query executors, e.g. (Thorne et al. 2020, 2021; Liu et al. 2021). There are other fundamentally different approaches than these for neural query execution, for example (Ren, Hu, and Leskovec 2020); our hope is that our data source is useful for exploring these. (Tuan et al. 2022) introduce a Transformer to generate responses to questions by reasoning over differentiable knowledge graphs in both task-oriented and domain specific chit-chat dialogues. Our structured neural memory baseline follows works such as (Locatello et al. 2020; Santoro et al. 2018). In this work, the relational "objects' do not need to be discovered by the learning algorithm, and their properties explicitly given to the model to use in featurizing the objects. Our work is most related to the pigpen environment
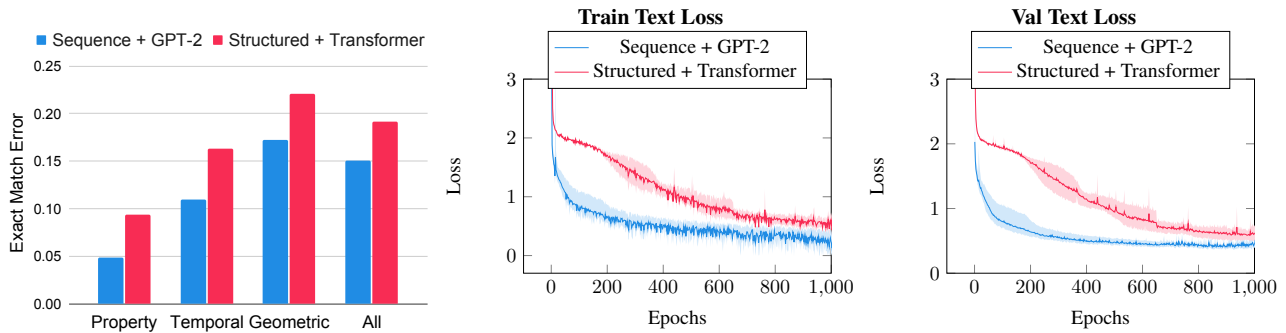
Figure 3: (left): Exact match error for the four different generated datasets. Sequence Context + GPT-2 outperform the Structured + Transformer method in all datasets. (middle, right): Loss curves for the All queries dataset. We show the mean loss with min/max error bars over all hyperparameters for the first 1,000 epochs. The pre-trained GPT-2 model learns much faster than the from-scratch relational model.

of (Zellers et al. 2021). In comparison to that work, ours uses more impoverished templated language, but has a larger and more flexible space of queries. The orientation of our work is also different: in (Zellers et al. 2021) the learner is given only a few labeled QA examples (or dynamics prediction examples) but can use many "unsupervised" state-transitions to build a world-dynamics model. In our work, we allow large numbers of labeled context-query-answer examples; but the difficulty of the queries makes the task non-trivial.

TextWorld (Côté et al. 2018) and and QAit (Yuan et al. 2019) are text-based environments for game play and question answering that require interactive reasoning. The main difference is that our generator is grounded in a 3D gridworld scene, and there is both a user and interactive agent.

We build our data generator on top of the Droidlet agent (Pratik et al. 2021), in part using the grammar in (Srinet et al. 2020) to generate the queries and using the Droidlet agent memory to execute them. This work points the way towards using neural networks to execute the functions of the Droidlet agent memory (and hopefully can be used as a resource for training other agent-memory models).

Experiments

Since the agent's memory or state can take different forms (sequence and structured), we compare two separate models for answering queries about the context of the world. We consider four different datasets for our experiments, as covered in Table 1: Property queries, Temporal queries, Geometric queries, and All queries, where *All queries* is the three previous categories combined (each query type has roughly the same likelihood of occurring - we provide the configuration files in the code).

Each of these except for Properties (which do not require temporal information) is generated using two time snapshots with 50 world steps, which gives enough steps for actions to occur. Properties queries use one snapshot and zero world steps. For all queries, we place five NPCs in the world, one of which is a "player" that might have given a command. For all query types, we choose the world to be 15x15x15.

Models

We generate data as described in the Environment, Queries, and Data section, and analyze the performance of some baseline models trained on this data.

Text Sequence Context. Since the text sequence form of the context is English text, we use a language model to read the context (C_t) and query (Q_t), and predict the correct answer tokens sequentially (if there are multiple outputs, they are ordered alphabetically). We use the pretrained GPT-2 small model (Radford et al. 2019) from the HuggingFace library (Wolf et al. 2019) (licensed under the Apache License 2.0) to predict all relevant tokens sequentially:

$$\hat{W} = \text{GPT2}([C_t, Q_t]), \qquad (1)$$

where $\hat{W} \in \mathbb{R}^{L \times V}$ is a soft-max normalized matrix of sequence length L and vocabulary size V, and $[\,]$ is the concatenation operation. This model is fine-tuned using a sum of the cross entropy between each token prediction $\hat{W}_i$ and ground truth token W_i:

$$\mathcal{L}_{text} = -\sum_{i \in S} \sum_{j \in V} W_{ij} \log \hat{W}_{ij}. \qquad (2)$$

Structured Context. While the text Sequence Context is useful in that it allows easy application of standard pretrained models, it may be that for certain tasks other representations are more appropriate. We also show results with simple models that are designed around the relational structure of the context. Given a set of ρ reference objects and τ triples, we first featurize the nodes using a learned convolutional layer given the word, float, and hash values as outlined in the Environment, Queries, and Data section. The output of the featurizer layer gives reference object embeddings $C_R \in \mathbb{R}^{\rho \times d}$, and triple embeddings $C_T \in \mathbb{R}^{\tau \times d}$. Similarly, text queries $Q_t \in \mathbb{R}^{r \times d}$ are created using a learned lookup table from the query tokens. We use a Transformer (Vaswani et al. 2017) encoder to process the context and query. The output of the encoder is then used to predict both: the text answer as well as the relevant context memory IDs. We use a Transformer decoder for the text prediction, and a simple linear layer to predict the memory values:
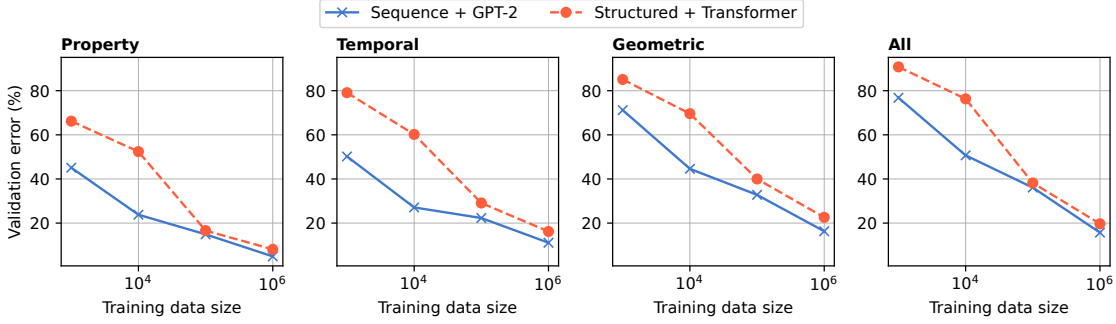
Figure 4: Validation set exact match error using varying amounts of training data. The pre-trained GPT-2 model is particularly useful when the number of training samples is small, and when reasoning over proprty queries. On the other hand, non-pre-trained GPT-2 style models do not learn, see Table 2. Both models require more than 100K training samples to achieve the best results.

$$(C'_T, C'_R, Q'_t) = \text{Encoder}([C_T, C_R, Q_t]) \qquad (3)$$

$$\hat{\boldsymbol{m}} = \text{MemidDecoder}([C'_R, C'_T]) \qquad (4)$$

$$\hat{\boldsymbol{w}} = \text{TextDecoder}([C'_T, C'_R, Q'_t]), \qquad (5)$$

where $\hat{\boldsymbol{m}} \in \mathbb{R}^\rho$ is predicted relevant memids, and $\hat{\boldsymbol{w}} \in \mathbb{R}^V$ represents predicted relevant text tokens. This model is trained using a cross-entropy text token loss (eq 2) and cross-entropy memid loss from the relevant memids $\boldsymbol{m} \in \mathbb{R}^\rho$:

$$\mathcal{L}_{memid} = -\sum_{i=1}^{C} \hat{m}_i \log (m_i). \qquad (6)$$

The total loss is a weighted sum of the two losses: $\mathcal{L}_{text} + \lambda \cdot \mathcal{L}_{memid}$, where $\lambda = 0.5$.

In the Sequence Context, the entities are represented as a set, with defined relationships between them. Therefore, encoding temporal information is not straightforward. To do so, we add a special "time" embedding to each Sequence Context node. That is, for timestep 0, we add the $time$=0 embedding, for timestep 1, we add the $time$=1 embedding.

For the Sequence Context, the model will return the relevant memory IDs, a text sequence answer, or both. The text output prediction is shown in Fig. 2.

Relational Embedding. One important form of structure in data is relations where entities are connected to other entities. In our data, for example, triple nodes are connected to object nodes by R_id. Since the vanilla Transformer without positional embeddings treats input tokens as a set, it lacks a mechanism for taking account of this relational information. Thus, we propose a novel way of encoding relational information directly into the self-attention mechanism of the Sequence Context Transformer model. Specifically, we add an extra term to the softmax attention:

$$a_{ij} = \text{Softmax}(q_i^T k_j + q_i^T r_{ij}) \qquad (7)$$

Here r_{ij} is a *relation embedding* vector corresponding to the relation type between token i and j. Relation embeddings can be used to encode various types of relations. We note that the commonly used relative position embeddings (Sukhbaatar et al. 2015; Shaw, Uszkoreit, and Vaswani 2018) are a special case of relation embeddings where $r_{ij} = e_{i-j}$; more

sophisticated relation embeddings have appeared in (Bergen, O'Donnell, and Bahdanau 2021).

Model and Training Details

All our models are trained using Adam (Kingma and Ba 2014) for 5,000 epochs, where each epoch is over a chunk of 10,000 training samples. Since we are generating the data, we vary the training samples from 1k to 1M, and use a validation set of 10k samples. We use a linear warmup of 10,000 steps and cosine decay (Loshchilov and Hutter 2016). For the GPT2 model, we consider learning rates {1e-4, 5e-4, 1e-5} using a batch size of 32. For the structured model, we consider learning rates {1e-4, 5e-4, 1e-5}, batch size 32, layers {2, 3}, and embedding dimensions {256, 512}. Hyperparameters were chosen with arbitrary initial values and increased until validation performance decreased or resources were depleted. The best performing structured model has 74.58M parameters, whereas GPT-2 small has 325.89M. All words are encoded with the GPT-2 tokenizer.

Results

Fig. 3 (left) shows the results for the four different dataset versions we consider. We report the exact match error for all data splits. That is,

$$\text{Exact Match Error} = \frac{1}{n} \sum_{i=1}^{n} I\left(\boldsymbol{y}^i \neq \hat{\boldsymbol{y}}^i\right), \qquad (8)$$

where $\boldsymbol{y}^i \in \mathbb{R}^N$ are the N ground truth tokens for sample i, and $\hat{\boldsymbol{y}}^i$ are the top N predicted tokens. Fig. 3 (right) shows the loss curves for the All queries dataset.

For property and geometric queries, the Sequence Context + GPT-2 method performs the best. Since GPT-2 is pre-trained on millions of samples, it can easily learn basic property queries such as "what is the color of bob?". GPT-2 has a near zero test set loss for the properties queries. Geometric queries, while difficult for both models to answer is solved more effectively by GPT-2.

Table 2 (top) shows GPT-2 model variation studies for the All queries dataset. Notably, we test the performance of a Sequence Context + randomly initialized GPT-2 model (i.e. one with the same architecture, but not pretrained). We see

Model	Variation	Train Loss	Test Loss	Test Err
GPT-2	sm + rand init	0.361	1.912	47.1%
	sm + pretrain	0.015	0.710	14.9%
	med + pretrain	0.012	0.635	13.8%
Transfomer	relational emb	0.230	0.921	19.7%
	random hash	0.630	1.393	30.0%

Table 2: Variations of models (sm=small, med=medium). Pretraining is a key component of the GPT-2 model success, even when given large numbers of training examples. Relational embeddings result in a slightly lower test loss than random-hash in the graph-structured models.

that the performance is worse than that of the Structured Context + Transformer. This indicates that the primary reason for the GPT-2 model achieving the best results is due to the pretraining. Table 2 (bottom) shows a Structured Context + Transformer method variation for the All queries dataset. We consider an alternative to the relational embeddings proposed in the Models section, which is adding random hash embeddings to the nodes, or tokens, which are connected to each other in the Sequence Context representation. This method performs slightly worse than relational embeddings, hinting that a more explicit representation of the context structure is important. Fig. 5 shows a 2-snapshot scene from our test set.

Finally, Fig. 4 shows the result of our baseline models when using varying amounts of training samples. Since our data source can generate arbitrary amounts of data, it's important to understand how much data is needed to solve certain problems. Notably, most of the datasets require at least 100,000 samples to achieve a reasonable validation loss. In addition, the Sequence + GPT-2 model significantly outperforms the Structured + Transformer model on the datasets with small amounts of training samples.

Conclusion

In this work, we introduced a framework for generating world state contexts with agents, queries, and their answers. This provides researchers with a flexible sandbox for training and testing reasoning in embodied agents. Notably, our sequential data format lets us easily evaluate the ability of large language models such as GPT to understand a physical world.

We show baseline results with two representations of the world state context: a pure text sequence representation that can be used by any off-the-shelf language model, as well as a structured representation of the entities in the world. We demonstrate the ability of the models to answer queries in several query domains, such as temporal or geometric.

We emphasize that our goal in this work is not to create a fixed static data set, but a resource for generating data. Many of our experimental choices (w.r.t. environment and query difficulty) were aimed to allow the baseline models some traction, but not able to solve the tasks completely.

In particular, we find it likely that large pre-trained language models could do better than the GPT-2 base models we used with the described data generation parameters. On the other hand, the difficulty of the problem can be scaled trivially by increasing the number of possible reference ob-
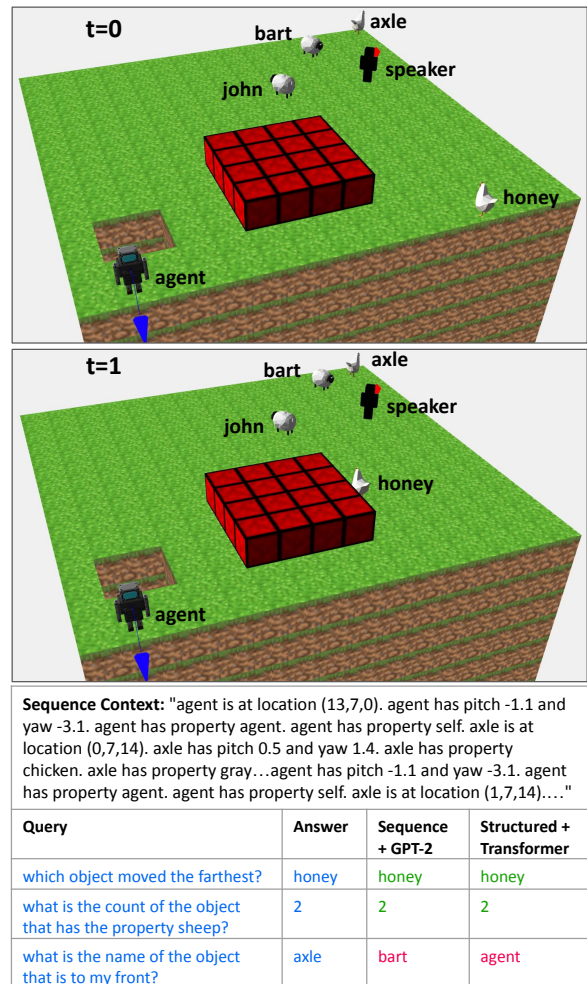

Sequence Context: "agent is at location (13,7,0). agent has pitch -1.1 and yaw -3.1. agent has property agent. agent has property self. axle is at location (0,7,14). axle has pitch 0.5 and yaw 1.4. axle has property chicken. axle has property gray…agent has pitch -1.1 and yaw -3.1. agent has property agent. agent has property self. axle is at location (1,7,14)…."

Query	Answer	Sequence + GPT-2	Structured + Transformer
which object moved the farthest?	honey	honey	honey
what is the count of the object that has the property sheep?	2	2	2
what is the name of the object that is to my front?	axle	bart	agent

Figure 5: Sample scene with two snapshots. The chicken named "honey" moves from the bottom right toward the center, and some other objects move slightly. Both models correctly answer the query "which object moved the farthest?".

jects and properties, by increasing the number of reference objects in the world and increasing the number of instantiated properties; by making the world bigger, and by increasing the time-length of episodes and the number of snapshots recorded. If nothing else, these changes would quickly lead to the context being too large to fit in the memory of a standard LM, and necessitating a large-memory LM (Lample et al. 2019; Rae et al. 2019; Beltagy, Peters, and Cohan 2020); or leading to other approaches e.g. (Ji et al. 2017). Other, more subtle difficulties can be introduced in straightforward ways by reducing full observability, or otherwise asking queries that cannot be answered with the information in the context, requiring agents some amount of meta-cognition and/or environment actions to find the answers, as in (Das et al. 2018).

In future work, we plan to introduce more query types, including arithmetic and hypotheticals. We hope that researchers will use the data generator as a flexible resource for augmenting their own agent training or LM pre-training, or for exploring new models for database reasoning.

References

Agrawal, A.; Lu, J.; Antol, S.; Mitchell, M.; Zitnick, C. L.; Parikh, D.; and Batra, D. 2015. VQA: Visual Question Answering. *International Journal of Computer Vision*, 123: 4–31.

Ahn, M.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.; David, B.; Finn, C.; Gopalakrishnan, K.; Hausman, K.; Herzog, A.; et al. 2022. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. *arXiv preprint arXiv:2204.01691*.

Beltagy, I.; Peters, M. E.; and Cohan, A. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Bergen, L.; O'Donnell, T.; and Bahdanau, D. 2021. Systematic generalization with edge transformers. *Advances in Neural Information Processing Systems*, 34.

Cangea, C.; Belilovsky, E.; Liò, P.; and Courville, A. 2019. Videonavqa: Bridging the gap between visual and embodied question answering. *arXiv preprint arXiv:1908.04950*.

Chevalier-Boisvert, M.; Bahdanau, D.; Lahlou, S.; Willems, L.; Saharia, C.; Nguyen, T. H.; and Bengio, Y. 2018. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv preprint arXiv:1810.08272*.

Chowdhery, A.; Narang, S.; Devlin, J.; Bosma, M.; Mishra, G.; Roberts, A.; Barham, P.; Chung, H. W.; Sutton, C.; Gehrmann, S.; et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Côté, M.-A.; Kádár, A.; Yuan, X.; Kybartas, B.; Barnes, T.; Fine, E.; Moore, J.; Hausknecht, M.; Asri, L. E.; Adada, M.; et al. 2018. Textworld: A learning environment for text-based games. In *Workshop on Computer Games*, 41–75. Springer.

Das, A.; Datta, S.; Gkioxari, G.; Lee, S.; Parikh, D.; and Batra, D. 2018. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–10.

Dehghani, M.; Gouws, S.; Vinyals, O.; Uszkoreit, J.; and Kaiser, L. 2019. Universal Transformers. *ArXiv*, abs/1807.03819.

Henaff, M.; Weston, J.; Szlam, A. D.; Bordes, A.; and LeCun, Y. 2017. Tracking the World State with Recurrent Entity Networks. *ArXiv*, abs/1612.03969.

Hill, F.; Bordes, A.; Chopra, S.; and Weston, J. 2016. The Goldilocks Principle: Reading Children's Books with Explicit Memory Representations. *CoRR*, abs/1511.02301.

Huang, W.; Abbeel, P.; Pathak, D.; and Mordatch, I. 2022. Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. *arXiv preprint arXiv:2201.07207*.

Hudson, D. A.; and Manning, C. D. 2019. GQA: A New Dataset for Real-World Visual Reasoning and Compositional Question Answering. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 6693–6702.

Ji, Y.; Tan, C.; Martschat, S.; Choi, Y.; and Smith, N. A. 2017. Dynamic entity representations in neural language models. *arXiv preprint arXiv:1708.00781*.

Johnson, J.; Hariharan, B.; van der Maaten, L.; Fei-Fei, L.; Zitnick, C. L.; and Girshick, R. B. 2017. CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1988–1997.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krishna, R.; Zhu, Y.; Groth, O.; Johnson, J.; Hata, K.; Kravitz, J.; Chen, S.; Kalantidis, Y.; Li, L.; Shamma, D. A.; Bernstein, M. S.; and Fei-Fei, L. 2016. Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations. *CoRR*, abs/1602.07332.

Lample, G.; Sablayrolles, A.; Ranzato, M.; Denoyer, L.; and Jégou, H. 2019. Large memory layers with product keys. *Advances in Neural Information Processing Systems*, 32.

Liu, Q.; Chen, B.; Guo, J.; Lin, Z.; and Lou, J. 2021. TAPEX: Table Pre-training via Learning a Neural SQL Executor. *CoRR*, abs/2107.07653.

Locatello, F.; Weissenborn, D.; Unterthiner, T.; Mahendran, A.; Heigold, G.; Uszkoreit, J.; Dosovitskiy, A.; and Kipf, T. 2020. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 33: 11525–11538.

Loshchilov, I.; and Hutter, F. 2016. SGDR: Stochastic Gradient Descent with Restarts. *CoRR*, abs/1608.03983.

Nye, M.; Andreassen, A. J.; Gur-Ari, G.; Michalewski, H.; Austin, J.; Bieber, D.; Dohan, D.; Lewkowycz, A.; Bosma, M.; Luan, D.; et al. 2021. Show Your Work: Scratchpads for Intermediate Computation with Language Models. *arXiv preprint arXiv:2112.00114*.

Pasupat, P.; and Liang, P. 2015. Compositional Semantic Parsing on Semi-Structured Tables. In *ACL*.

Petroni, F.; Piktus, A.; Fan, A.; Lewis, P.; Yazdani, M.; De Cao, N.; Thorne, J.; Jernite, Y.; Karpukhin, V.; Maillard, J.; et al. 2020. KILT: a benchmark for knowledge intensive language tasks. *arXiv preprint arXiv:2009.02252*.

Pratik, A.; Chintala, S.; Srinet, K.; Gandhi, D.; Qian, R.; Sun, Y.; Drew, R.; Elkafrawy, S.; Tiwari, A.; Hart, T.; et al. 2021. droidlet: modular, heterogenous, multi-modal agents. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 13716–13723. IEEE.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Rae, J. W.; Potapenko, A.; Jayakumar, S. M.; and Lillicrap, T. P. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *EMNLP*.

Ren, H.; Hu, W.; and Leskovec, J. 2020. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. *arXiv preprint arXiv:2002.05969*.

Santoro, A.; Faulkner, R.; Raposo, D.; Rae, J.; Chrzanowski, M.; Weber, T.; Wierstra, D.; Vinyals, O.; Pascanu, R.; and Lillicrap, T. 2018. Relational recurrent neural networks. *Advances in neural information processing systems*, 31.

Shaw, P.; Uszkoreit, J.; and Vaswani, A. 2018. Self-Attention with Relative Position Representations. In *NAACL-HLT (2)*.

Srinet, K.; Jernite, Y.; Gray, J.; and Szlam, A. 2020. CraftAssist instruction parsing: Semantic parsing for a voxel-world assistant. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 4693–4714.

Sukhbaatar, S.; Weston, J.; Fergus, R.; et al. 2015. End-to-end memory networks. *Advances in neural information processing systems*, 28.

Talmor, A.; Herzig, J.; Lourie, N.; and Berant, J. 2019. CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge. *ArXiv*, abs/1811.00937.

Thorne, J.; Yazdani, M.; Saeidi, M.; Silvestri, F.; Riedel, S.; and Halevy, A. 2020. Neural databases. *arXiv preprint arXiv:2010.06973*.

Thorne, J.; Yazdani, M.; Saeidi, M.; Silvestri, F.; Riedel, S.; and Halevy, A. 2021. Database reasoning over text. *arXiv preprint arXiv:2106.01074*.

Tuan, Y.-L.; Beygi, S.; Fazel-Zarandi, M.; Gao, Q.; Cervone, A.; and Wang, W. Y. 2022. Towards Large-Scale Interpretable Knowledge Graph Reasoning for Dialogue Systems. *arXiv preprint arXiv:2203.10610*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Vedantam, R.; Szlam, A.; Nickel, M.; Morcos, A.; and Lake, B. M. 2021. Curi: A benchmark for productive concept learning under uncertainty. In *International Conference on Machine Learning*, 10519–10529. PMLR.

Wang, B.; Shin, R.; Liu, X.; Polozov, O.; and Richardson, M. 2019. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. *arXiv preprint arXiv:1911.04942*.

Wang, X.; Wei, J.; Schuurmans, D.; Le, Q.; Chi, E.; and Zhou, D. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Weston, J.; Bordes, A.; Chopra, S.; and Mikolov, T. 2016. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. *arXiv: Artificial Intelligence*.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W. W.; Salakhutdinov, R.; and Manning, C. D. 2018a. HotpotQA: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W. W.; Salakhutdinov, R.; and Manning, C. D. 2018b. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *EMNLP*.

Yi, K.; Gan, C.; Li, Y.; Kohli, P.; Wu, J.; Torralba, A.; and Tenenbaum, J. B. 2020. CLEVRER: CoLlision Events for Video REpresentation and Reasoning. *ArXiv*, abs/1910.01442.

Yi, K.; Wu, J.; Gan, C.; Torralba, A.; Kohli, P.; and Tenenbaum, J. 2018. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. *Advances in neural information processing systems*, 31.

Yin, P.; Neubig, G.; Yih, W.-t.; and Riedel, S. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. *arXiv preprint arXiv:2005.08314*.

Yuan, X.; Côté, M.-A.; Fu, J.; Lin, Z.; Pal, C.; Bengio, Y.; and Trischler, A. 2019. Interactive language learning by question answering. *arXiv preprint arXiv:1908.10909*.

Zambaldi, V.; Raposo, D.; Santoro, A.; Bapst, V.; Li, Y.; Babuschkin, I.; Tuyls, K.; Reichert, D.; Lillicrap, T.; Lockhart, E.; et al. 2018. Deep reinforcement learning with relational inductive biases. In *International conference on learning representations*.

Zellers, R.; Holtzman, A.; Peters, M.; Mottaghi, R.; Kembhavi, A.; Farhadi, A.; and Choi, Y. 2021. Piglet: Language grounding through neuro-symbolic interaction in a 3d world. *arXiv preprint arXiv:2106.00188*.

Zhang, S.; Roller, S.; Goyal, N.; Artetxe, M.; Chen, M.; Chen, S.; Dewan, C.; Diab, M.; Li, X.; Lin, X. V.; et al. 2022. OPT: Open Pre-trained Transformer Language Models. *arXiv preprint arXiv:2205.01068*.

Zhong, V.; Rocktäschel, T.; and Grefenstette, E. 2019. Rtfm: Generalising to novel environment dynamics via reading. *arXiv preprint arXiv:1910.08210*.