Modeling Dynamic Environments with Scene Graph Memory

Andrey Kurenkov [1] **Michael Lingelbach** [1] **Tanmay Agarwal** [1] **Emily Jin** [1] **Chengshu Li** [1] **Ruohan Zhang** [1] **Li Fei-Fei** [1] **Jiajun Wu** [1] **Silvio Savarese** [2] **Roberto Martín-Martín** [3]

Abstract

Embodied AI agents that search for objects in large environments such as households often need to make efficient decisions by predicting object locations based on partial information. We pose this as a new type of link prediction problem: **link prediction on partially observable dynamic graphs**. Our graph is a representation of a scene in which rooms and objects are nodes, and their relationships are encoded in the edges; only parts of the changing graph are known to the agent at each timestep. This partial observability poses a challenge to existing link prediction approaches, which we address. We propose a novel state representation – Scene Graph Memory (SGM) – with captures the agent's accumulated set of observations, as well as a neural net architecture called a Node Edge Predictor (NEP) that extracts information from the SGM to search efficiently. We evaluate our method in the Dynamic House Simulator, a new benchmark that creates diverse dynamic graphs following the semantic patterns typically seen at homes, and show that NEP can be trained to predict the locations of objects in a variety of environments with diverse object movement dynamics, outperforming baselines both in terms of new scene adaptability and overall accuracy. The codebase and more can be found this URL.

1. Introduction

Temporal link prediction is the problem of estimating the likelihood of edges being present in the future in a dynamically changing graph based on past observed instances of the full graph (Divakaran & Mohan, 2020). This type of problem appears when analyzing social networks, communication networks, or even biological networks. We investigate a novel instance of this problem: temporal link prediction with partial observability, i.e. when the past observations of the graph contain only parts of it. This setting maps naturally to a common problem in embodied AI: using past sensor observations to predict the state of a dynamic environment represented by a graph. Graphs are used frequently as the state representation of large scenes in the form of *scene graphs* (Johnson et al., 2015; Armeni et al., 2019; Ravichandran et al., 2022a; Hughes et al., 2022), a relational object-centric representation where nodes are objects or rooms, and edges encode relationships such as `inside` or `onTop`. Link prediction could be applied to partially observed, dynamic scene graphs to infer relationships between pairs of objects enabling various downstream decision-making tasks for which scene graphs have been shown to be useful such as navigation (Amiri et al., 2022; Santos & Romero, 2022), manipulation (Agia et al., 2022; Zhu et al., 2021) and object search (Ravichandran et al., 2022a; Xu et al., 2022).

In this paper, we study link prediction in dynamic, partially observable graphs with a focus on using this formalism to perform **object search** with an embodied AI agent in a large scene. Although this is a popular problem, most past works assumes a static scene with the agent having no prior memory of it, whereas we focus on dynamic scenes with a continually learning agent. For that, we first **propose a novel state representation** named a **scene graph memory (SGM)** that encodes the nodes and edges the agent has observed—including those that may no longer be true—in a single graph with reference to the time the nodes and edges were lastly observed. The SGM enables the agent to gradually build up a representation of all of its observations, which can then be the input to a link prediction model.

Existing solutions for dynamic link prediction assume full observability (knowing all the nodes and edges in graphs from prior timesteps) and are not well suited to this new kind of link prediction problem. Therefore, we **introduce a novel solution for dynamic link prediction in partial observable settings** based on a neural net architecture we call a **Node Edge Predictor (NEP)**. The NEP is designed to predict the likelihood of a single node's edges with a self-attention mechanism that "compares" them.

[1]Department of Computer Science, Stanford University [2]Salesforce AI Research [3]Department of Computer Science, University of Texas at Austin. Correspondence to: Andrey Kurenkov <andreyk@stanford.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

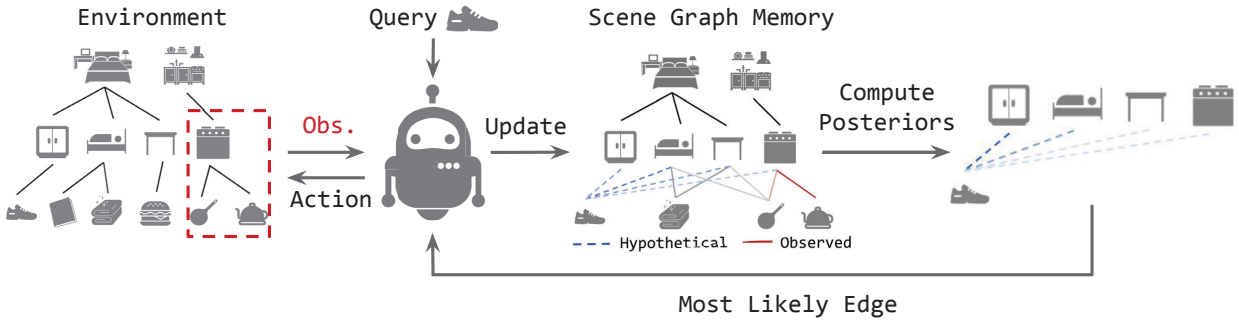

Figure 1. Our problem setup and proposed method: an agent is queried to find target objects in an unknown dynamic household environment where objects change location over time and objects may be added or removed – a specific instance of the general problem of link prediction in partially observable dynamic graphs. The agent can observe subsets of the true scene graph, which are aggregated in a Scene Graph Memory which is then passed into the Node Edge Predictor model that produces observation-conditioned posterior probabilities of where the query object is. Finally, the agent uses the posterior probabilities to decide on its next action.

Lastly, we **address the lack of existing benchmarks for link prediction with partially observable graphs** by introducing the **Dynamic House Simulator**. Existing simulators for evaluating embodied agents in household environments (Li et al., 2021a; Shen et al., 2021; Savva et al., 2019; Gan et al., 2020; Deitke et al., 2022) change only by actions of the embodied AI agent, whereas the Dynamic House Simulator enables sampling diverse dynamic scenes that realistically evolve over time.

In summary, the contribution of our work is three-fold:

- We propose a new state representation, the Scene Graph Memory, to enable embodied agents to aggregate their observations of a dynamic environment over time.
- We introduce a novel neural net architecture, the Node Edge Predictor, and show it can be used to perform link prediction learning on dynamic, partially observable graphs. To the best of our knowledge, this is the first solution to this type of problem.
- We present the Dynamic House Simulator, a novel benchmark to evaluate embodied agent performance in dynamic, partially observable environments, and show that our proposed method significantly outperforms multiple baselines.

2. Related Work

Temporal link prediction: Link prediction is the problem of predicting the likelihood of unknown links (edges) in graphs. While the community has studied the problem extensively in static graphs (we refer to Kumar et al. (2020) for a comprehensive review), link prediction in dynamic graphs, or temporal link prediction as it is also known, is also an important emerging problem (Divakaran & Mohan, 2020). Recently, there has been an increasing amount of approaches based on graph neural networks (GNNs) for non-temporal (Zheng et al., 2021; Opolka & Lió, 2022; Zhao et al., 2022; Atzeni et al., 2021) as well as temporal link prediction (Qu et al., 2020; Lei et al., 2019; Skarding et al., 2022; Singh et al., 2021) due to the capabilities of GNNs to encode and efficiently propagate graph information for inference. Several works also combine GNNs with transformers (Yang et al., 2021; Jin et al., 2022), which we also do with our Node Edge Predictor.

We focus on a modified version of the temporal link prediction problem. Motivated by challenges embodied AI agents typically face, we add the property of partial observability: only subsets of the prior input graphs are known. This makes the problem significantly more challenging, as the model must make its predictions based on less information. Furthermore, our problem formulation allows for not just the edges but also the nodes in the graph to be dynamic, which has rarely been the case in prior work (Haghani & Keyvanpour, 2017; Ran et al., 2022). To our knowledge, this is the first work to introduce the problem of temporal link prediction on partially observable dynamic graphs.

Modeling object relations in embodied AI research: Often in embodied AI tasks, the relations between objects in the environment provide critical information for decision-making. Previous works have designed various data structures or representations that embed co-occurrence statistics between objects such as probabilistic semantic maps (Li & Meng, 2012; Veiga et al., 2016), graphical models (Kim & Suh, 2019; Kollar & Roy, 2009; Aydemir et al., 2013; Lorbach et al., 2014), hierarchical models (Pronobis et al., 2017), extended POMDPs (Zheng et al., 2022), and scene graphs (Kurenkov et al., 2021).

Although object co-occurrence statistics can be learned by the agent from scratch in an environment, it is beneficial to extract such statistics prior to agent deployment from

knowledge sources that capture commonsense about object relations, such as hand-coded priors (Lorbach et al., 2014), online text (Zhou et al., 2012) or image (Kollar & Roy, 2009) datasets, or curated knowledge base (Toro et al., 2014). We introduce a novel way to extract a scene graph specific co-occurrence statistics through counting of observed object relations in the simulated environments of iGibson and ProcThor (Li et al., 2021a; Deitke et al., 2022).

Modeling object locations in scenes: As summarized in Crespo et al. (2020), graphs have commonly been used in the embodied AI literature as memory for tasks that require semantic information. We highlight several recent works that are particularly related to ours. Wu et al. (2019) also uses observations to build a Bayesian probabilistic relation graph of the room connectivity in novel houses, but is focused on static environments with no objects, and does not used link prediction. Kurenkov et al. (2021) also uses link prediction on a scene graph for finding objects, but does so in the context of static scenes without the need to learn object dynamics. Du et al. (2022) also attempts to capture the dynamics of object movement by maintaining an object-based memory and training attention-based neural networks, but focuses on visual rather than semantic information and within a much smaller number of locations and objects than us. Rudra et al. (2022) also produces probabilities over object locations but does so with a contextual bandits approach over vantage points with no use explicit memory, and for just three comparatively simple environments. Lastly and most related, the concurrent work of Patel & Chernova (2022) also learns a predictive model of object dynamics with the use of scene graphs and GNNs as well as a household simulator, but assumes full knowledge of past states of the environment rather than dealing with partial observability, does not use the graph as a form of memory, does not make use of linguistic cues for generalization, and focuses on realistically simulating five households rather than procedural generation of any number of households.

3. Problem Formulation

The standard temporal link prediction problem for dynamic graphs is defined as follows (Liben-Nowell & Kleinberg, 2003; Divakaran & Mohan, 2020): The state of a dynamic graph at time t is represented as $G_t = (V, E_t)$, where E_t is the set of edges present at time t and V are the nodes. Given past observations of the state of G from time step 0 to t, $G_0, ..., G_t$, the goal is to predict the presence of all future edges in the next step, E_{t+1}.

Our problem formulation (see Fig. 1) is a derivation of the above with three major differences. First, the set of nodes in the graph can change between timesteps, so that $V_t \neq V_{t+1}$. Second, we assume partial observability; instead of having full access to past graph states, $G_0, ..., G_t$, our agent has only access to partial observations of these states, $O_0, ..., O_t$. Each partial observation, $O_t = (V_t^O, E_t^O)$, contains the current state of a subset of the nodes, $V_t^O \subseteq V$, and the edges, $E_t^O \subseteq E_t$. The content of an observation, O, is task-dependent; in our embodied AI context it will be the result of agent actions, representing a realistic sensing operation.

The third difference is in the scope of the prediction: instead of attempting at estimating the state of *all* edges, our goal is to predict the state of *a subset* of the edges, $E_t^Q \subseteq E_{t+1}$. We assume this subset is associated with query nodes V_t^Q of interest for a downstream task. The query edges as well as the query node may or may not have been seen in a past observation and may or may not change from G_t to G_{t+1}. This is a natural setup in several embodied AI tasks such as object search, where the agent does not need to infer the state of the entire environment but only the relevant information to find a specific target objects.

Object search with temporal link prediction in dynamic, partially observable graphs: While our method is applicable to temporal link prediction with any form of graph information, we focus on the problem of reasoning about scene graphs in household environments. In this domain, the nodes in a given scene graph are hierarchically organized: at the top are the room nodes, then furniture nodes, and then object nodes. The edges represent the kinematic relations between nodes, such as `inside` and `onTop`. The edges between furniture nodes and object nodes are dynamic, and the underlying environment dynamics modify the edges according to some unknown probability distribution. In other words, objects "move" over time as they do in real household settings populated by humans.

Given this household context, the observations $O_0, ..., O_t$ consist of one or more furniture nodes along with the objects connected to these furniture nodes with an edge. The goal of our agent is to correctly predict the furniture node that is connected to a target object node. The agent can choose furniture nodes to receive an observation from, which represents it exploring to observe where objects are. The agent's observations are noisy: some objects may not be observed even if connected to the chosen furniture node. This simulates realistic perception that may fail to detect objects due to occlusions or other factors.

4. Method

To study and address the dynamic link prediction problem with partial observability, our work includes several components: (1) The Dynamic House Simulator – a mechanism to simulate household environments with diverse object layouts and distributions, (2) Scene Graph Memory (SGM) – a state representation to aggregate agent observations and serve as a basis for learning to predict where objects are, and

(3) Node Edge Predictor (NEP) – a novel neural network architecture that is best suited for predicting the presence of query edges in the scene graph.

4.1. Dynamic House Simulator

Algorithm 1 Dynamic House Simulator Algorithm

Require: Initial prior probability graph P^{prior}
  for $i = 0, 1, \ldots, M$ **do**
    Create a noisy copy of P^{prior}: $P^i \leftarrow P^{prior} + N^i_{class}$
    Sample object instances from P^i and create scene graph SG^i_0 at its initial state ($t = 0$)
    Create environment dynamics: $T^i \leftarrow P^i + N^i_{instance}$
    Define a new env instance: $Env^i = \{SG^i_0, T^i\}$
  end for
  for $t = 1, 2, \ldots, t_{end}$ **do**
    for $i = 0, 1, \ldots, M$ **do**
      Evolve scene graph: $SG^i_t \leftarrow T^i(SG^i_{t-1})$
    end for
  end for

Due to there not being a well suited benchmark for our task, we created the Dynamic House Simulator to benchmark link prediction in dynamic, partially observable graphs in the context of object search in households. There exist several simulation frameworks for embodied AI in household environments (Li et al., 2021a; Shen et al., 2021; Savva et al., 2019; Gan et al., 2020; Deitke et al., 2022) but none of them simulate the object movement that results from other agents such as humans interacting and changing object locations. Our simulator supports sampling a wide variety of household environments with a distinct set of initial objects, locations, and patterns of object movement over time.

Priors graph. The first component of the simulator is a graph encoding the probabilities of room-furniture and furniture-object relationships for all households (Fig 5). We call this our prior probabilities graph P^{prior} because it represents the commonsense knowledge about any given household environment prior to observing it. The probability of an object being `inside` or `onTop` a piece of furniture depends on which room the furniture is in. We compute these probabilities via simple counting of the presence of relationships in different environments of iGibson 2.0 (Li et al., 2021a) and ProcTHOR-10k (Deitke et al., 2022), two simulators with realistic object placements in house environments, but the prior could come from any source of furniture-object-room distribution such as a large language model.

After obtaining the prior probabilities, we manually annotate each node in P^{prior} with the following attributes: a set of adjectives that could describe the object or piece of furniture (e.g. "blue", "metal", etc.), the min/max number of the object in the environment, and how likely the object is to move between furniture or to disappear from or appear in the environment over time. These attributes enable more varied and realistic simulation of the household environment. Additional details can be found in the Appendix (Sec. A).

Scene sampling and evolving. The priors probabilities, P^{prior}, are the basis for the simulator algorithm to sample diverse dynamic environments in the form of scene graphs, which we now describe (Alg. 1 and Fig. 5). To sample an environment instance, Env^i, we first inject class-level noise, N^i_{class}, to the prior graph to obtain the environment-specific relation probabilities, P^i. The class-level noise is a function of object class, furniture class, and relationship type. For example, for the specific environment Env^i, a jar has an 80% probability of being inside a shelf and a 20% probability of being inside a cabinet. The specifics of noise generation are included in the Appendix (Sec. A).

Then, the initial scene graph SG^i_0 will be sampled based on P^i in a procedural manner. Given the minimum and maximum bounds, P^i is used to first sample a set of rooms, then a set of furniture items within each room, and then objects for each piece of furniture. Note that there might be multiple instances of the same object type in SG^i_0. To capture the fact that different instances of the same object type are distinct in their appearance or material, each sampled node is associated with a description, in addition to its type. The description is generated by randomly sampling a subset of adjectives from the prior graph. For instance, a "mug" node may get the description "large red mug".

After this, we create the environment-specific object relation change probabilities, or dynamics for short, T^i by injecting instance-level noise $N^i_{instance}$ into P^i. The instance-level noise $N^i_{instance}$ is a function of object description, furniture description, and relationship type. For example, a "large red mug" has a $50\%, 30\%, 5\%, 15\%$ chance of being inside of shelf 1, shelf 2, cabinet 1, and cabinet 2 respectively.

At each timestep, the dynamics T^i will be used to modify the scene by changing the furniture-object relations in the scene graph SG^i_t to create SG^i_{t+1}. T^i can also potentially add or remove an object instance based on P^i. As a result, each environment is defined in terms of its unique initial scene graph and unique object dynamics $Env^i = \{SG^i_0, T^i\}$, which conceptually corresponds to a unique household.

iGridson. We implement a gridworld version of iGibson 2.0 (Li et al., 2021a), named iGridson, to ground the agent in an environment with support for 3D-object relations. In particular, given an initial scene graph, iGridson instantiates an environment such that every object in the environment corresponds to a node in the scene graph, and the objects are sampled to satisfy all edge relationships in the scene graph. The environment evolves according to the object dynamics. More details are included in the Appendix (Sec. F).

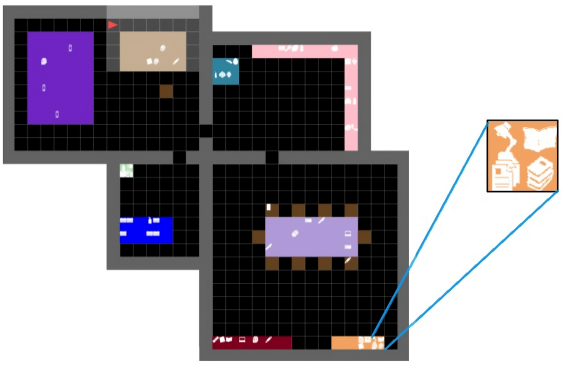

Figure 2. iGridson, our gridworld implementation of the iGibson 2.0 (Li et al., 2021a) simulator. An example household scene with four rooms is shown here with various furniture items and objects.

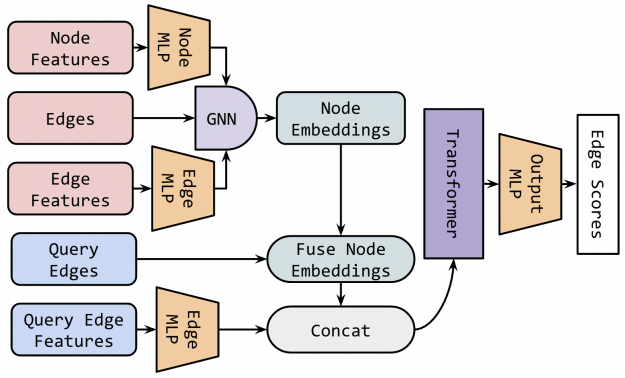

Figure 3. Node Edge Predictor (NEP) model architecture (GCN and HEAT variants). See the Appendix (Sec. B) for details on the node and edge features.

4.2. Scene Graph Memory (SGM)

As described in Sec. 3, the goal of the agent is to predict the probability of query edges in each scene graph SG_t^i. Since each environment has distinct probability distributions P^i and T^i, the agent needs to make use of the observations from the environment $O_0^i ... O_t^i$ to estimate these environment-specific distributions. We propose the Scene Graph Memory (SGM) data structure to enable the agents to do so.

To simplify notation, the following applies to an agent operating in a specific environment Env^i. At timestep t, an instance of a Scene Graph Memory $SGM_t = (V_t^{SGM}, E_t^{SGM})$ is composed of a set of nodes and edges of the same type as in the environment scene graphs. The SGM nodes $V_t^{SGM} = (\bigcup_{n=0}^{t} V_n^O) \bigcup V_t^Q$ are made up of all the observed nodes, as well as the set of query nodes – the latter may not or may not be a subset of the former, since the query may be related to objects that agent has not seen before. The SGM edges $E_t^{SGM} = (\bigcup_{n=0}^{t} E_n^O) \bigcup E_t^H$ are made up of all the observed edges up until timestep t, and any new hypothetical edges. Hypothetical edges are edges the agent can predict for the query nodes without actually having observed them according to some function $E_t^H = f_h(V_t^Q)$; these are needed when dealing with queries of nodes that have not been observed yet or have few observed edges. f_h could be implemented in various ways; we implement it by adding an edge to the SGM for every edge in P^{prior} with a probability above a certain threshold.

Each node and edge in the SGM is associated with features reflecting the semantic properties of the object or relationship it represents as well its observed dynamics. Semantic properties are captured by word embedding associated with the node's rooms, furniture, or object label (eg "mug"). We use the 96-dimensional Tok2Vec vectors optimized on `en_core_web_sm` from the spaCy python package (Honnibal & Montani, 2017). These semantic features are useful for recognizing nodes that are likely to share similar edges. The observed "temporal" features include the time since a node or edge has been observed, the number of times it has been observed, its state change frequency, and more. These observed features are necessary for the model to learn the environment-specific dynamics. More details are included in the Appendix (Sec. B).

A useful property of using the SGM representation is that it collapses the sequence of observations $O_0, ..., O_t$ into a single graph. Thus, unlike prior works that relied on hard-to-optimize recurrent models, we only need our model to reason about SGM_t.

4.3. Node Edge Predictor (NEP)

Model architecture. We are concerned with predicting the probabilities of query edges, with a focus on the case when these edges all connect to a single node. This motivates our design for the Node Edge Predictor (NEP) model (Fig. 3).

NEP consists of four modules: node embedding, edge embedding, feature fusion, and edge classification. The node and edge embedding layers consist of a three-layer MLP with ReLU activations. The node embedding layers also optionally include graph convolutional layers after the MLP. In other words, we first embed the raw input features for nodes and edges and then optionally perform message passing to propagate the embedded features across the graph.

We evaluate three variants of the model with different node embedding modules: **NEP-MLP** includes no graph convolution layers, **NEP-GCN** performs a two-layer graph convolution operation on top of the MLP embedding, and **NEP-HEAT** (Mo et al., 2021) replaces the graph convolution layers with heterogeneous edge-enhanced graph attentional (HEAT) operators, which condition the attention computed for message passing on both the node and edge features. **NEP-HEAT** is chosen as our primary variant because we expect that edge-conditioned attention will allow for more

effective feature propagation. HEAT was originally designed for trajectory prediction of traffic, so this is the first time it is being applied to link prediction.

After creating embeddings for all nodes and edges, we fuse the features of each pair of nodes associated with a query edge by averaging the nodes' embeddings and then concatenating that with the edge's embedding. Since different query nodes have different numbers of edges, we pad the input tensors for each batch before fusion. Lastly, the batch of fused features is passed to a 2-layer transformer encoder (Vaswani et al., 2017). The self-attention layers in the transformer enable the model to evaluate all the query edges jointly, which is important for cases in which a node's edges are mutually exclusive. NEP's use of a GNN followed by a transformer is most similar to GraphFormer from Yang et al. (2021) Heterformer from Jin et al. (2022), but differs in that NEP uses the transfomer only on the embedded query edges, as opposed to the costlier use of transformers in alteration with GNN layers. Lastly, the transformer's output are passed through a 3-layer MLP with a Sigmoid activation in order to yield the output logits across all edge candidates.

Model training. For each training iteration, we randomly sample a batch of SGMs from the training dataset, and batch all the query edges from each SGM. The loss is computed by calculating the Binary Cross-Entropy between the logits and the ground-truth labels. We optimize for binary classification per edge as opposed to N-way classification because a given node may have multiple edges that are true. Because the number of true edges is far lower than false edges, we multiply the losses corresponding to false edges by the ratio of true edges to false edges before backpropagation.

5. Experimental Setup

We design our experiments to test whether the proposed NES models can outperform alternative approaches across three downstream tasks that involve link prediction.

5.1. Training Data Collection

In order to train the model, we first collect data by having an agent gather observations in a variety of training environments. The agent's goal is to complete the same object search task that the SGM model is meant to help with, and can follow any policy during data collection, such as the heuristic baselines explained below. As the agent tries the complete the task, it gathers observations and constructs SGMs along the way. Besides the features described above, each query edge in the SGM_t^i is also annotated with a label – true or false – which corresponds to whether this edge is actually present in the ground-truth scene graph. Each node is also labeled as being a query node or not. Once a variety of SGMs with labels have been collected, they can be aggregated into an offline dataset for model training.

5.2. Tasks and Metrics

We define three tasks with associated metrics as the basis of our experiments.

Predict Object Location: At every timestep, the agent must predict the location (furniture node) of an object with a particular description. The agent is then able to observe the node it has predicted and its associated object nodes, regardless of whether its output is correct. The metric for the task is accuracy – whether the agent correctly predicted a furniture node that is connected to an object with the correct description. This task is designed to evaluate how well an embodied agent may work in practice in an unknown environment; predicting the correct location of a given query object is essential for downstream tasks like household chores.

Predict Relative Location Likelihood: At every timestep, the agent is queried for multiple objects and is required to predict the likelihood of each location that each object can be at. The metric for the task is the Normalized Discounted Cumulative Gain (NDCG), a popular method for measuring the quality of a set of search results(Järvelin & Kekäläinen, 2002). We choose this metric as we primarily care about the agent correctly predicting the ranking of the location options rather than the exact values in T. The agent is then able to observe the mostly likely nodes for each query node, as in the previous task. Compared to the previous task, this task and metric provide a more holistic evaluation of how well the agent models the entire environment at every timestep.

Find Object: The prior two tasks involve a single prediction at each time step. This task has the same objective as predict location, but the agent is now allowed multiple sequential choices. After each location choice, the agent can observe the location and update its internal state before making the prediction for the next action. The environment is static during this, so exhaustive search will always succeed. The metric for this task is the number of actions it takes to pick a correct node.

5.3. Baselines

We compare our models with six heuristic-based models as well as the most highly relevant prior work.

These are the heuristic baselines: **Random:** Randomly chooses an edge among all options. **Frequentist:** Records the number of times edges have been observed to be true and false, and chooses the option with the highest ratio of true observations to total observations. **Priors:** Chooses the most likely option according to P^{prior}. **Myopic:** Always chooses the last location each object was observed at, or at random if the object has not been observed. **Bayesian:** Treats each

edge in the SGM as having a distinct beta-binomial probability distribution. We create the distributions with a beta prior based on P^{prior} and compute the posterior distribution by treating observations as a sequence of Bernoulli trials. More details in the Appendix (Sec. D). **Oracle:** Uses ground truth knowledge about the dynamics of the scene as well as memory of past observed object locations to make the best choice possible.

Lastly, we also compare to the HMS model from Kurenkov et al. This model was also designed for link prediction in the context of scene graphs, but in the context of static scenes. Thus, it is a good point of comparison for our revised problem definition. Compared to our model, it does not use the SGM representation and uses a standard GCN neural net rather than the NEP.

6. Experiment Results

We evaluate our NEP variants against all baselines on the test set (unseen environments) for all three tasks. We train each NEP model with a dataset of 10,000 SGMs collected over 100 different environments, with 100 steps being taken per environment. The data collection is done by having the Bayesian baseline do the relevant task while creating and storing SGM graphs along the way.

6.1. Task Performance

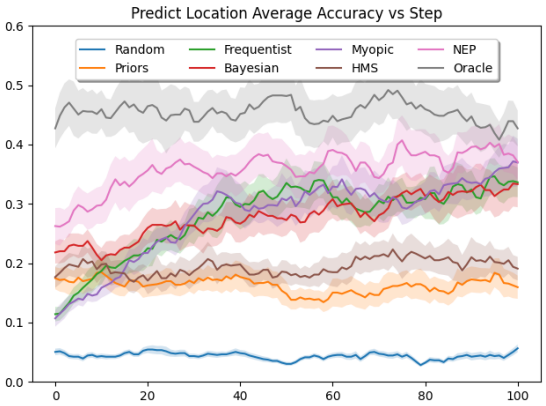

Figure 4. The average accuracy and variance for the Predict Object Location task, averaged across 100 different environments. The x-axis is the environment steps. The y-axis is smoothed with averaging over 10 steps. NEP results are with the HEAT variant.

The main results can be seen in Fig. 4 and Table 1. We have the following observations.

First, as seen in Fig. 4, the use of SGM enables the NEP to improve over time, unlike models that completely rely on the prior information and do not make any use of observations (Priors and HMS). The model also outperforms the baselines that do make use of observations (Frequentist, Myopic, and Bayesian), which we hypothesize is due to the NEP's understanding of object semantics, use of GNN-based feature propagation, and the self-attention mechanism that allows for edge comparisons.

Second, the priors embedded in the SGM as well as learned semantic patterns make the NEP able perform better than all heuristics from the outset. This gives NEP a head start compared to the models that purely rely on observations made during test time (Myopic and Frequentist). These baselines do approach the performance of NEP over time, since eventually it's possible to model the dynamics perfectly. However, NEP still has the advantage of faster and better adaptation to new environments.

Third, NEP also significantly outperforms the Bayesian method, which is given access to both the priors and observations over time. We believe this is because the Bayesian approach reasons on a per-edge basis, whereas NEP is able to propagate information about edge features with the use of a GNN and fuse information about query edges with the use of self-attention. Thus, it is possible to share information about the dynamics of nodes with similar semantics.

Lastly, the detailed results in Table 1 demonstrate several things. As could be expected, making the environment dynamic both in terms of object location and object presence makes the tasks harder, compared to always dealing with the same set of objects. The HEAT NEP variant generally performs best, but the simpler GCN variant performs slightly better on the predict relative likelihood task, possibly because producing a ranking of edges is easier than specifically predicting the most likely edge. Lastly, the first two tasks do correspond to better downstream performance on finding objects through a sequence of actions, as is our ultimate goal.

The results support our initial hypothesis that combining the proposed representation (SGM) with the NEP model is suitable for the temporal link prediction tasks in our setting, as this combination allows generalization to unseen environments as well as online adaptation.

6.2. Ablations

We further perform several ablation studies to test the relative importance of the components in our model (Fig. 3). The results are shown in Table 2. It is evident that all of our model designs contribute significantly to the model performance. The prior probability information is most critical, not only because it is included in node edge, but crucially also because it is used for selection of hypothetical edges; with no priors, hypothetical edges are sampled at random and so may not include the true object location. Even though the priors do not match the test environments' true dynamics, they are highly beneficial for improving performance.

Table 1. The mean accuracy and standard deviation (averaged across 100 runs/environments) are shown for three tasks and different environment conditions. Lower is better for Predict Object Location and Find Object. Dynamic nodes refer to whether nodes are added and removed throughout scene evolution, which makes the problem significantly more challenging.

Task	Predict Object Location ↑		Predict Relative Location Likelihood ↑		Find Object ↓	
Dynamic Nodes?	Y	N	Y	N	Y	N
Random	0.046 ± 0.005	0.044 ± 0.005	0.381 ± 0.001	0.384 ± 0.001	8.769 ± 1.092	8.750 ± 1.100
Priors	0.159 ± 0.019	0.180 ± 0.025	0.536 ± 0.001	0.546 ± 0.002	5.911 ± 1.433	5.869 ± 1.438
Frequentist	0.271 ± 0.029	0.315 ± 0.038	0.668 ± 0.001	0.728 ± 0.002	4.550 ± 1.571	4.218 ± 1.624
Myopic	0.282 ± 0.030	0.339 ± 0.043	0.029 ± 0.000	0.017 ± 0.000	6.333 ± 2.202	6.048 ± 2.186
HMS	0.194 ± 0.023	0.210 ± 0.033	0.529 ± 0.001	0.582 ± 0.002	5.357 ± 1.379	5.254 ± 1.458
Bayesian	0.286 ± 0.028	0.308 ± 0.034	0.699 ± 0.002	$0.725 \pm 0.0\ 02$	3.469 ± 0.761	3.610 ± 0.970
NEP-MLP	0.302 ± 0.032	0.336 ± 0.038	0.661 ± 0.002	0.736 ± 0.002	3.650 ± 1.035	3.445 ± 0.948
NEP-GCN	0.324 ± 0.034	0.359 ± 0.041	$\mathbf{0.728 \pm 0.001}$	$\mathbf{0.792 \pm 0.002}$	3.629 ± 1.042	3.191 ± 0.866
NEP-HEAT	$\mathbf{0.351 \pm 0.033}$	$\mathbf{0.391 \pm 0.041}$	0.724 ± 0.002	0.783 ± 0.002	$\mathbf{3.570 \pm 1.032}$	$\mathbf{3.186 \pm 0.910}$
Oracle	0.454 ± 0.039	0.475 ± 0.063	0.932 ± 0.002	0.939 ± 0.002	3.001 ± 0.772	2.921 ± 0.764

Table 2. Ablation study results. The mean accuracy and variance (averaged across 100 runs/environment) are reported for for the Predict Object Location task with dynamic nodes. Higher is better.

	NEP-MLP	NEP-GCN	NEP-HEAT
Full Model	0.302 ± 0.032	0.324 ± 0.034	0.351 ± 0.033
(-) Prior probability	0.260 ± 0.028	0.233 ± 0.024	0.267 ± 0.029
(-) Transformer	0.263 ± 0.028	0.310 ± 0.034	0.314 ± 0.032
(-) Temporal features	0.288 ± 0.032	0.320 ± 0.033	0.322 ± 0.031
(-) Semantic features	0.263 ± 0.027	0.322 ± 0.030	0.328 ± 0.030

Table 3. Mean success rate, number of actions, and path length ($\pm$ standard deviation) to reach the query objects in the iGridson environment averaged over 10k runs. The agent is given at most 10 actions to reach the target object.

	Success Rate	# of Actions	Path Length
Random	0.57	9.36 ± 4.09	141.74 ± 63.86
Myopic	0.57	9.31 ± 4.14	140.92 ± 64.40
Bayesian	0.84	5.06 ± 4.50	70.90 ± 63.06
NEP-HEAT	**1.00**	$\mathbf{1.56 \pm 0.86}$	$\mathbf{30.05 \pm 24.67}$

The use of the transformer layers, a key aspect of the NEP model, turns out to be the second most important design choice. This validates our hypothesis that it is useful to perform self-attention over query edges prior to predicting their likelihoods. Interestingly, temporal features are the most important for the HEAT variant, suggesting it makes the most use of them for adapting to novel environments. Lastly, semantic features in the SGM turn out to be the least important, perhaps because temporal features alone are sufficient to infer semantics.

6.3. Downstream Task Performance in iGridson

We also evaluate the agent's performance on the Find Object task in the iGridson environment shown in Fig 2. The quantitative results are shown in Table 3. This environment contains only 21 furniture locations, so the agent is successful if it predicts the correct location by searching up to roughly half the options. Unlike the Find Object in Table 1, the iGridson representation includes a spatial layout rather than just a symbolic one. Therefore, it is possible to compute the path length the agent traverses across all its actions. The NEP method is always able to find the object, and does so within fewer actions and shorter paths. The variances for these metrics are surprisingly high, potentially because some objects are much harder to find than others. We hope to explore performance in iGridson more in the future.

7. Conclusion

We proposed a new problem setup, temporal link prediction for dynamic and partially observable graphs, as well as a first solution to this problem in the context of object search in embodied AI. Our work includes a new benchmark, the Dynamic House Simulator, where it is possible to evaluate model performance on dynamic link prediction in the embodied AI context. Our solution to predict object locations and environment dynamics efficiently includes a new representation, Scene Graph Memory (SGM), and a novel net learned architecture, the Node Edge Predictor (NEP). Our SGM-based models significantly outperform all the alternative approaches due to their ability to (1) learn scene statistics (commonsense knowledge about object relations) during training, and (2) adapt online by leveraging noisy, partial observations. These features help embodied AI agents perform object searches in unseen, dynamic, and partially observable environments, as well as link prediction models with partially observable graphs in general.

Acknowledgments. This work is in part supported by Stanford Institute for Human-Centered Artificial Intelligence (HAI), NSF RI #2211258, ONR MURI N00014-22-1-2740, AFOSR YIP FA9550-23-1-0127, Analog Devices, JPMorgan Chase, Meta, and Salesforce.

References

Agia, C., Jatavallabhula, K. M., Khodeir, M., Miksik, O., Vineet, V., Mukadam, M., Paull, L., and Shkurti, F. Taskography: Evaluating robot task planning over large 3d scene graphs. In *Conference on Robot Learning*, pp. 46–58. PMLR, 2022.

Amiri, S., Chandan, K., and Zhang, S. Reasoning with scene graphs for robot planning under partial observability. *IEEE Robotics and Automation Letters*, 7(2): 5560–5567, 2022.

Armeni, I., He, Z.-Y., Gwak, J., Zamir, A. R., Fischer, M., Malik, J., and Savarese, S. 3d scene graph: A structure for unified semantics, 3d space, and camera. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5664–5673, 2019.

Atzeni, D., Bacciu, D., Errica, F., and Micheli, A. Modeling edge features with deep bayesian graph networks. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2021.

Aydemir, A., Pronobis, A., Göbelbecker, M., and Jensfelt, P. Active visual object search in unknown environments using uncertain semantics. *IEEE Transactions on Robotics*, 29(4):986–1002, 2013.

Bear, D., Fan, C., Mrowca, D., Li, Y., Alter, S., Nayebi, A., Schwartz, J., Fei-Fei, L. F., Wu, J., Tenenbaum, J., et al. Learning physical graph representations from visual scenes. *Advances in Neural Information Processing Systems*, 33:6027–6039, 2020.

Chevalier-Boisvert, M., Willems, L., and Pal, S. Minimalistic gridworld environment for gymnasium, 2018. URL https://github.com/Farama-Foundation/Minigrid.

Crespo, J., Castillo, J. C., Mozos, O. M., and Barber, R. Semantic information for robot navigation: A survey. *Applied Sciences*, 10(2):497, 2020.

Deitke, M., VanderBilt, E., Herrasti, A., Weihs, L., Salvador, J., Ehsani, K., Han, W., Kolve, E., Farhadi, A., Kembhavi, A., et al. Procthor: Large-scale embodied ai using procedural generation. *arXiv preprint arXiv:2206.06994*, 2022.

Divakaran, A. and Mohan, A. Temporal link prediction: A survey. *New Generation Computing*, 38(1):213–258, 2020.

Du, Y., Lozano-Perez, T., and Kaelbling, L. P. Learning object-based state estimators for household robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

Fey, M. and Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Gan, C., Schwartz, J., Alter, S., Schrimpf, M., Traer, J., De Freitas, J., Kubilius, J., Bhandwaldar, A., Haber, N., Sano, M., et al. Threedworld: A platform for interactive multi-modal physical simulation. *arXiv preprint arXiv:2007.04954*, 2020.

Haghani, S. and Keyvanpour, M. R. Temporal link prediction: techniques and challenges. *Computer science and information technologies. Yerevan*, 2017.

Honnibal, M. and Montani, I. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

Hughes, N., Chang, Y., and Carlone, L. Hydra: A real-time spatial perception engine for 3d scene graph construction and optimization. *arXiv preprint arXiv:2201.13360*, 2022.

Järvelin, K. and Kekäläinen, J. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

Jin, B., Zhang, Y., Zhu, Q., and Han, J. Heterformer: A transformer architecture for node representation learning on heterogeneous text-rich networks. *arXiv preprint arXiv:2205.10282*, 2022.

Johnson, J., Krishna, R., Stark, M., Li, L.-J., Shamma, D., Bernstein, M., and Fei-Fei, L. Image retrieval using scene graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3668–3678, 2015.

Kim, M. and Suh, I. H. Active object search in an unknown large-scale environment using commonsense knowledge and spatial relations. *Intelligent Service Robotics*, 12(4): 371–380, 2019.

Kollar, T. and Roy, N. Utilizing object-object and object-scene context when planning to find things. In *2009 IEEE International Conference on Robotics and Automation*, pp. 2168–2173. IEEE, 2009.

Kumar, A., Singh, S. S., Singh, K., and Biswas, B. Link prediction techniques, applications, and performance: A survey. *Physica A: Statistical Mechanics and its Applications*, 553:124289, 2020.

Kurenkov, A., Martín-Martín, R., Ichnowski, J., Goldberg, K., and Savarese, S. Semantic and geometric modeling with neural message passing in 3d scene graphs for

hierarchical mechanical search. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11227–11233. IEEE, 2021.

Lei, K., Qin, M., Bai, B., Zhang, G., and Yang, M. Gcn-gan: A non-linear temporal link prediction model for weighted dynamic networks. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 388–396. IEEE, 2019.

Li, C., Xia, F., Martín-Martín, R., Lingelbach, M., Srivastava, S., Shen, B., Vainio, K., Gokmen, C., Dharan, G., Jain, T., et al. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. *arXiv preprint arXiv:2108.03272*, 2021a.

Li, K. and Meng, M. Q.-H. Indoor scene recognition via probabilistic semantic map. In *2012 IEEE International Conference on Automation and Logistics*, pp. 352–357. IEEE, 2012.

Li, W., Song, X., Bai, Y., Zhang, S., and Jiang, S. Ion: Instance-level object navigation. In *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 4343–4352, 2021b.

Liben-Nowell, D. and Kleinberg, J. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management*, pp. 556–559, 2003.

Lorbach, M., Höfer, S., and Brock, O. Prior-assisted propagation of spatial information for object search. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2904–2909. IEEE, 2014.

Mo, X., Xing, Y., and Lv, C. Heterogeneous edge-enhanced graph attention network for multi-agent trajectory prediction. *CoRR*, abs/2106.07161, 2021. URL https://arxiv.org/abs/2106.07161.

Opolka, F. and Lió, P. Bayesian link prediction with deep graph convolutional gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pp. 4835–4852. PMLR, 2022.

Pal, A., Qiu, Y., and Christensen, H. Learning hierarchical relationships for object-goal navigation. In *Conference on Robot Learning*, pp. 517–528. PMLR, 2021.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Patel, M. and Chernova, S. Proactive robot assistance via spatio-temporal object modeling. *arXiv preprint arXiv:2211.15501*, 2022.

Pronobis, A., Riccio, F., and Rao, R. P. Deep spatial affordance hierarchy: Spatial knowledge representation for planning in large-scale environments. In *ICAPS 2017 Workshop on Planning and Robotics*, pp. 1–9, 2017.

Qu, L., Zhu, H., Duan, Q., and Shi, Y. Continuous-time link prediction via temporal dependent graph neural network. In *Proceedings of The Web Conference 2020*, pp. 3026–3032, 2020.

Ran, Y., Liu, S.-Y., Yu, X., Shang, K.-K., and Jia, T. Predicting future links with new nodes in temporal academic networks. *Journal of Physics: Complexity*, 3(1):015006, 2022.

Ravichandran, Z., Peng, L., Hughes, N., Griffith, J. D., and Carlone, L. Hierarchical representations and explicit memory: Learning effective navigation policies on 3d scene graphs using graph neural networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2022a.

Ravichandran, Z., Peng, L., Hughes, N., Griffith, J. D., and Carlone, L. Hierarchical representations and explicit memory: Learning effective navigation policies on 3d scene graphs using graph neural networks. In *2022 International Conference on Robotics and Automation (ICRA)*, pp. 9272–9279. IEEE, 2022b.

Rudra, S., Goel, S., Santara, A., Gentile, C., Perron, L., Xia, F., Sindhwani, V., Parada, C., and Aggarwal, G. A contextual bandit approach for learning to plan in environments with probabilistic goal configurations. *arXiv preprint arXiv:2211.16309*, 2022.

Santos, I. B. d. A. and Romero, R. A. A deep reinforcement learning approach with visual semantic navigation with memory for mobile robots in indoor home context. *Journal of Intelligent & Robotic Systems*, 104(3):1–21, 2022.

Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9339–9347, 2019.

Seymour, Z., Mithun, N. C., Chiu, H.-P., Samarasekera, S., and Kumar, R. Graphmapper: Efficient visual navigation by scene graph generation. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pp. 4146–4153. IEEE, 2022.

Shen, B., Xia, F., Li, C., Martín-Martín, R., Fan, L., Wang, G., Pérez-D'Arpino, C., Buch, S., Srivastava, S., Tchapmi, L., et al. igibson 1.0: a simulation environment for interactive tasks in large realistic scenes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7520–7527. IEEE, 2021.

Singh, A., Huang, Q., Huang, S. L., Bhalerao, O., He, H., Lim, S.-N., and Benson, A. R. Edge proposal sets for link prediction. *arXiv preprint arXiv:2106.15810*, 2021.

Skarding, J., Hellmich, M., Gabrys, B., and Musial, K. A robust comparative analysis of graph neural networks on dynamic link prediction. *IEEE Access*, 10:64146–64160, 2022.

Toro, W., Cozman, F. G., Revoredo, K., and Costa, A. H. R. Probabilistic relational reasoning in semantic robot navigation. In *URSW*, pp. 37–48, 2014.

Trivedi, R., Farajtabar, M., Biswal, P., and Zha, H. Dyrep: Learning representations over dynamic graphs. In *International conference on learning representations*, 2019.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Veiga, T. S., Miraldo, P., Ventura, R., and Lima, P. U. Efficient object search for mobile robots in dynamic environments: Semantic map as an input for the decision maker. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2745–2750. IEEE, 2016.

Wu, Y., Wu, Y., Tamar, A., Russell, S., Gkioxari, G., and Tian, Y. Bayesian relational memory for semantic visual navigation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2769–2779, 2019.

Xu, T., Yang, X., and Zheng, S. Learning hierarchical graph convolutional neural network for object navigation. In *International Conference on Artificial Neural Networks*, pp. 544–556. Springer, 2022.

Yang, J., Liu, Z., Xiao, S., Li, C., Lian, D., Agrawal, S., Singh, A., Sun, G., and Xie, X. Graphformers: Gnn-nested transformers for representation learning on textual graph. *Advances in Neural Information Processing Systems*, 34:28798–28810, 2021.

Yi, K., Gan, C., Li, Y., Kohli, P., Wu, J., Torralba, A., and Tenenbaum, J. B. Clevrer: Collision events for video representation and reasoning. In *International Conference on Learning Representations*, 2020.

Zhao, T., Liu, G., Wang, D., Yu, W., and Jiang, M. Learning from counterfactual links for link prediction. In *International Conference on Machine Learning*, pp. 26911–26926. PMLR, 2022.

Zheng, K., Chitnis, R., Sung, Y., Konidaris, G., and Tellex, S. Towards optimal correlational object search. In *2022 International Conference on Robotics and Automation (ICRA)*, pp. 7313–7319. IEEE, 2022.

Zheng, W., Huang, E. W., Rao, N., Katariya, S., Wang, Z., and Subbian, K. Cold brew: Distilling graph node representations with incomplete or missing neighborhoods. *arXiv preprint arXiv:2111.04840*, 2021.

Zhou, F., Liu, H., Zhao, H., and Liang, L. Long-term object search using incremental scene graph updating. *Robotica*, pp. 1–14, 2022.

Zhou, K., Zillich, M., Zender, H., and Vincze, M. Web mining driven object locality knowledge acquisition for efficient robot behavior. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3962–3969. IEEE, 2012.

Zhu, Y., Tremblay, J., Birchfield, S., and Zhu, Y. Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6541–6548. IEEE, 2021.

A. Dynamic House Simulator

A visual representation of Alg. 1 is shown in Fig 5.

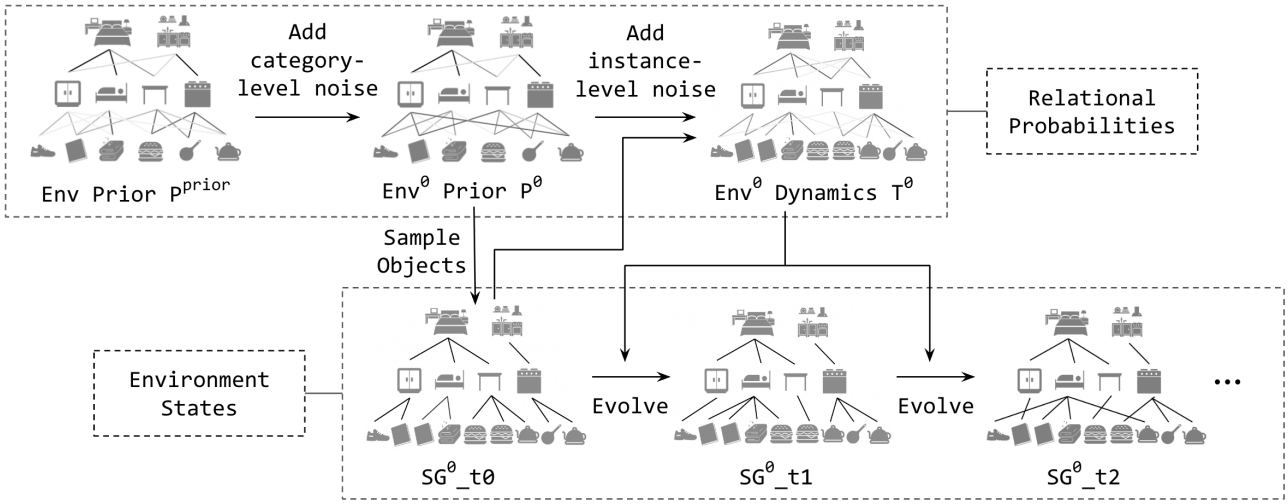

Figure 5. Illustration of Dynamic House Simulator scene Sampling and Evolving process. As outlined in Sec. 3 and Algo. 1, for each environment Env^i, we first inject object class-level noise N^i_{class} to the global prior P^{prior} to generate environment-specific relation probabilities P^i. The initial scene graph SG^i_0 will be sampled based on P^i in a procedural manner. Then we inject object instance-level noise $N^i_{instance}$ into the environment-specific relation probabilities P^i to generate the environment dynamics T^i, which will evolve the scene graph across time steps.

Object placement probabilities For ProcThor-10k, we counted the occurrences of object relationships and normalized to get point probabilities. iGibson already includes the probabilities as metadata, so we use that directly. For relationships that are in both, we use the ProcThor-10k. We compute object placement probabilities for ProcThor-10k by counting the number of occurrences in each relationship type within the 10,000 published train environments and normalizing the counts of all room-furniture relationships and furniture-object relationships. Specifically, we create counts of furniture-object edges counts per room and divide by the sum of all object instance in each room. iGibson 2.0 already encodes such probabilities in its metadata, so we only need to average the F-O probabilities for the coarse graph. We filter out any furniture nodes with fewer than 3 outgoing edges to focus computation on simulating busier aspects of the environment. The resulting P_{prior} graphs is visualized in Fig. 6.

We also create a the "coarse" version of the priors graph in which furniture-object edges are counted irrespective of rooms and divided by the sum of all object instances. The resulting P_{prior} graphs is visualized in Fig. 7. We did not use these priors in our experiments, since they lead to object dynamics that are strictly simpler than the "detailed" priors and therefore the results were less informative.

Sampling noise There are two types of class-level noise: sparsification and randomization. Sparsification means we zero out probabilities for some edges, e.g. a certain object will never be inside a certain piece of furniture. If an edge's probability is not zeroed out, we then randomly scale it up or down by a certain amount. After the application of noise, all edge probabilities for the object are normalized. The goal is to make each environment instance have its own specific locations for various objects. In our experiments we use a probability of 25% for both zeroing out edges and as the maximum and minimum by which probabilities may be scaled.

Environment Sampling An environment consists of 1 floor, 4 rooms (kitchen, living room, bedroom, bathroom), 8 furniture items per room, and 6 objects per piece of furniture. While our approach is compatible with different counts for each category, we chose this specific ratio (floor:room:furniture:item) to standardize our experiments. The furniture and object items are sampled from the candidate lists in Table 4 and Table 5. Adjectives are additional descriptors belonging to an adjective category such as size or color which are then sampled and attached to both furniture and objects by the following procedure. First, the number of adjective categories to attach to a given object or furniture items is sampled from a uniform

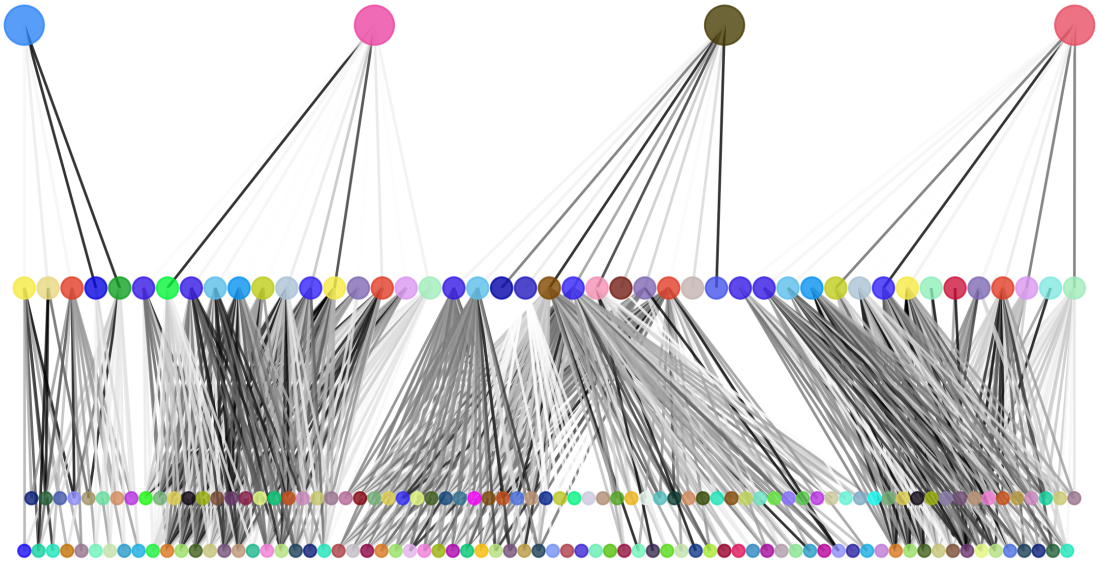

Figure 6. The household object placement probability priors. Nodes at the top correspond to rooms, nodes at the middle correspond to furniture, and nodes at the bottom correspond to objects. Node color a unique room, furniture, or object label. Edge color represents the likelihood of a piece of furniture being sampled for a room or the likelihood of an object being sampled for a piece of furniture, with darker edges having a higher likelihood. To improve visualization, a node with a given label is created per room that it is connected to.

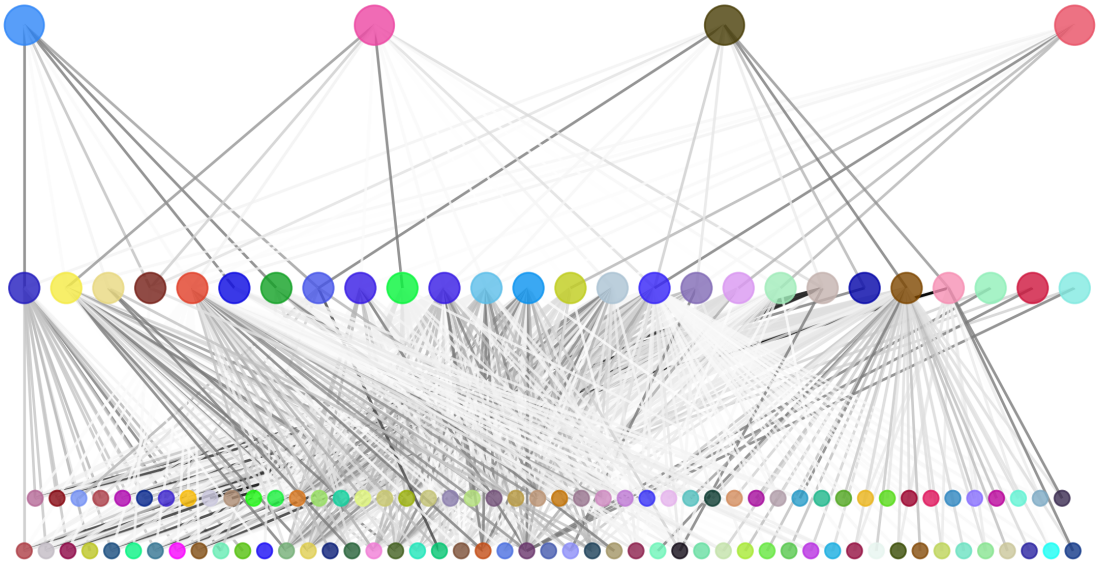

Figure 7. The "coarse" household object placement probability priors. The layout and use of color is the same as in Fig. 6. In the this version of the priors, the furniture-object relationships (i.e. edges at the bottom level) are independent from the room-furniture relationships (i.e. edges on the top level). Nodes with a given label are not replicated per room, meaning that each node represents a distinct room, furniture, or object type. Some nodes have primarily white edges because they have a large number of outgoing edges, which makes the probability of each edge lower compared to nodes with few outgoing edges.

distribution between 1 and the maximum number of categories. Then a specific adjective is uniformly sampled for each of these categories. This can result in multiple distinct objects sharing the same class existing in the scene, such as a "small green bag" or a "large red bag".

Evolution of the environment The evolution of the environment is determined by the move frequency and add/remove probability outlined in Table 5. First objects are removed based on the remove probability, with a constraint that the object count cannot dip below 95% of the original object count. In the second step, 5% of all objects are moved. The distribution of which objects move is determined by the move frequency of the individual move probabilities based on the object class as outlined in Table 5. Finally, we iterate over every possible object and spawn objects according to the add probability with the constraint that the total number of objects in the scene cannot be higher than 105% of the initial object count.

B. SGM representation

The SGM consists of a set of node (V_t^{SGM}) and edges (E_t^{SGM}). The node features are the following:

- The text embedding for the node is the 96-dimensional Tok2Vec vectors optimized on `en_core_web_sm` from the spaCy python package (Honnibal & Montani, 2017): 96 dimensional vector of floats in the range 0.0-1.0.
- The number of timesteps since the node was last observed: integer scalar in the range 0-100 (max timesteps).
- The number of times the node was observed: integer scalar in the range 0-100.
- The number of timesteps since the object was observed to have moved, which occurs during an evolution of the environment. This requires the agent to have observed the object to at least twice: integer scalar in the range 0-100.
- The observed frequency for which an object has moved: scalar with value in the range 0.0-1.0.
- The node type ('house', 'floor', 'room', 'furniture', 'object'): 5 dimensional 1-hot vector.

The edge features are the following:

- The cosine similarity between the Tok2Vec text embedding of the edge's two nodes: scalar in the range 0.0-1.0.
- The number of timesteps since the edge was last observed: Integer scalar in the range 0-100.
- The number of timesteps since the observed last state change: scalar with the value 0.0 or 1.0.
- The number of times the edge was observed: integer scalar in the range 0-100.
- The number of times the edge state was observed to be true (present in the scene graph): integer scalar in the range 0-100.
- The frequency with which the edge observed to be true (present in the scene graph): scalar in the range 0.0-1.0.
- The number of times the edge state was observed to have changed: integer scalar in the range 0-100.
- The last observed state of the edge: scalar with value 0.0 or 1.0.
- The prior probability that the edge exists derived from the ProcThor-10k and iGibson datasets: scalar with value in the range 0.0-1.0.
- The edge type ('in', 'contains', 'onTop', 'under'): 4 dimensional 1-hot vector.

All features are concatenated to form the feature vectors of nodes and edges. Prior to concatenation, temporal features (integers in the range 0-100) are first normalized to be roughly in the range 0.0-1.0 via multiplication by a manually derived scaling factor. For nodes this factor is $average\_scene\_graph\_num\_nodes(200)/(steps\_per\_scene(100) * 10.0)$, and for edges this is $average\_scene\_graph\_num\_edges(500)/(steps\_per\_scene(100) * 10.0)$.

C. Training hyperparameters

Implementation All models are implemented and trained using PyTorch (Paszke et al., 2019) and PyTorch-Geometric (Fey & Lenssen, 2019). Unless otherwise stated, we use the default parameters of these two packages.

Model parameters The node embedding network and edge embedding networks are two-layer feedforward neural networks with 64 units and ReLU activations following each layer. The HEAT and GCN graph neural networks have one graph convolution layer with 64 units. The transformer encoder is a standard self-attention model with 2 heads with a 64 unit feedforward network.

Training parameters The model was trained for 25 epochs with a batch size of 100. The Adam optimizer was used with a learning rate of 1×10^{-4}.

D. Task and Baseline Implementation Details

Detection error For all tasks, we simulate detection error by randomly skipping 25% of the objects during observation.

Choice of query object For the predict object location task, the query object is sampled at random either from the set of objects that has moved since the last step or from all object nodes. In the former case, the node is sampled uniformly, and in the latter case, it is sampled with a weight proportional to its movement probability. This ensures the agent sometimes has to predict the location of an object that is guaranteed to have been moved since it was last observed, which is a capability we wish to be able to evaluate.

Bayesian policy The bayesian baseline is based on a Bernoulli model with a beta prior. The beta prior parameters are computed to match a desired variance. Given prior probability μ and desired variance v, the compute the following:

$\alpha \leftarrow \mu^2 \times \left(\frac{1-\mu}{v} - \frac{1}{\mu}\right)$
$\beta \leftarrow \alpha \times \left(\frac{1}{\mu} - 1\right)$

This variance is chosen to be relatively large (0.05) to reflect that the priors don't in general match the dynamics of environments. The posterior predictive for an edge being true is then computed as follows:

$\alpha_n \leftarrow \alpha + \sum_{n=1}^{N} x_n$
$\beta_n \leftarrow N - \sum_{n=1}^{N} x_n + \beta$
$p \leftarrow \frac{\alpha_n}{\alpha_n + \beta_n}$

E. Additional Discussion

E.1. Combining our method with Reinforcement Learning

Multiple approaches have been proposed for integrating scene graphs with RL agents for navigation tasks, and as scene graph memory is the same sort of data structure it can be used in the RL scenario in the same sorts of ways. Examples of such approaches include (Ravichandran et al., 2022b; Li et al., 2021b; Seymour et al., 2022; Pal et al., 2021). As in these works, the SGM data structure can be used as input to the RL agent alongside its raw observations to aid the agent's decision making. Further, these works all utilize GNN layers to process the scene graphs as part of the policy network, and our NEP architecture could be used to fulfill this purpose. We performed some exploratory experiments in training an RL agent in the iGridson environment, and found convergence to be non-trivial. Therefore, we leave this problem to future work.

E.2. The complexity of Dynamic House Simulator tasks

While our simulator may seem simple, the environments the simulator generates are in many ways more complicated than any existing benchmarks for object search. Concretely, our generated environments are complex in terms of:

- Diversity of scenes – most benchmarks in this space only support a set number of pre-generated scenes, whereas ours can generate endless scene variations through controllable sampling
- Size of scenes – per Appendix A, our experiments are conducted in scenes with 4 rooms, 32 furniture items, and 192 objects. Prior benchmarks are typically limited to smaller spaces with significantly fewer objects.
- Variety of furniture and object types - our simulator supports over 20 furniture types and over 100 object types, with most of them having 3 or more possible modifying adjectives. This is far larger than the furniture or object variety in other benchmarks.
- Dynamics evolution - no other benchmark supports continual evolution of the scene state.

While our experiments focus on demonstrating results for symbolic scene graphs and the semi-realistic 2D iGridson environment, the approach we take for the latter can be directly extended to object search in realistic 3D spaces, as discuss next.

E.3. Applicability of our approach to realistic 3D embodied object search

Our work abstracts away the challenges of perception and navigation that are part of embodied object search in order to focus on the problem of modeling dynamics with the use of memory. Complementary to our focus, many recent works have demonstrated impressive performance on embodied instances of this problem that do require robust perception and navigation, but do not require modeling of environment dynamics of long term memory. A possible future direction for our research is to demonstrate the usefulness of scene graph memory and the node edge predictor model in the context of realistic embodied instances of this problem that these works address.

One possible approach for doing this is a direct extension of our approach for implementing the iGridson agent discussed in section 6.3 and Appendix F. Analogous to our 2D iGridson agent implementation, an agent can continually maintain a scene graph memory via processing of embodied observations, and then use NEP predictions to decide on navigation goals, which can be achieved via robust point-goal navigation. The works cited above could likely form the basis for implementing the necessary perception and point-goal navigation. In particular, the recent paper "Long-term object search using incremental scene graph updating" (Zhou et al., 2022) demonstrated the feasibility of incremental scene graph updating from visual observations, which can be directly extended to incrementally building scene graph memory.

Our simulator can also form the basis for adding dynamic objects to the benchmarks used in prior works (Trivedi et al., 2019; **?**; Yi et al., 2020; Bear et al., 2020; Zhou et al., 2022). As we demonstrate with the iGridson component of our simulator, it is possible to translate the symbolic scene graphs of the environment into embodied spaces. While we only demonstrate this for the simplified 2D setting, the same approach can be extended to more realistic 3D simulators by sampling object placements according to the scene graph. For instance, iGibson 2.0 (Li et al., 2021a) supports sampling object placements to satisfy object placement constraints. The primary difficulty with implementing this would be acquiring sufficient 3D assets for all the objects our simulator supports.

F. iGridson Embodied Environment Details

The iGridson simulator is a 2D embodiment of a home environment laid out in a grid format, built using the minigrid library (Chevalier-Boisvert et al., 2018). It enables translating the scene graphs generating by the Dynamic House Simulator into a spatial arrangement, which is what embodied agents actually have to deal with.

As shown in Fig.2, the environment has a fixed layout of the home with four rooms, each with a predetermined number and type of furniture objects. In total, the environment consists of 21 furniture objects spread across different rooms, all of which remains static throughout the evolution of the environment. A set of objects are then placed on top of these furniture objects, and are dynamically moved to a different furniture at each time step based on their priors and specific dynamics as described in section 4.1. Note that in our experiments, the overall set of objects itself is static. That is, no new objects are added to the environment, nor are any existing objects removed, they simply move from one furniture object to another.

Our experiments in this environment follow the Find Object task, wherein each agent is given multiple attempts to find a target object. If the agent correctly predicts the furniture object on which the object is placed, we count the event as a success and the task is terminated. In this setup, given a target object, the agent makes N predictions of furniture objects, which are sequentially visited by the agent. The cost returned by the environment is therefore the total path length traversed by the agent in an episode, and the number of attempts it took to find the target object ($N + 1$ if object could not be found in N steps). It should be noted that the path length itself is not a direct indicator of performance, as the furniture node on which an object is placed can sometimes be far away from the agent's current position. Hence, the optimal solution in this case would incur a high path length. As a result, even optimal agents which can find the target object in a few steps can have a high variance for the average path length when aggregated over thousands of episodes.

Table 4. Furniture types used in Dynamic House Sim and their associated metadata

label	# possible adjectives	sample probability	max count	# edges
counter	3	0.80	3	27
table	1	0.80	3	17
shelf	7	0.80	10	6
fridge	3	0.90	2	30
top cabinet	3	0.50	10	8
coffee table	4	0.60	3	34
cooktop	3	0.90	1	3
counter top	5	0.80	3	67
dining table	7	1.00	1	74
chair	7	0.75	12	40
tv stand	2	1.00	1	33
sofa	1	0.75	2	18
bed	3	0.90	2	17
dresser	7	0.75	3	36
toilet	1	1.00	1	15
sink	2	1.00	2	6
shelving unit	4	0.50	4	28
desk	7	0.70	4	30
chair	4	0.60	2	17
side table	7	0.70	3	64
chairs	7	0.75	12	40
couch	1	0.75	2	18

Table 5: Objects types used in Dynamic House Sim and their associated metadata

label	# adjectives	sample prob	max count	move frequency	add/remove prob	# edges
apple	2	0.80	15	0.40	0.20	9
box	6	0.70	4	0.40	0.01	11
cereal	9	0.50	4	0.20	0.10	2
dishtowel	11	0.75	8	0.20	0.00	1
flour	8	0.80	4	0.20	0.10	2
jar	10	0.75	6	0.15	0.05	3
kettle	6	0.75	4	0.40	0.00	4
lettuce	1	0.80	5	0.10	0.01	3
milk	6	0.80	2	0.10	0.10	2
mug	10	0.80	12	0.60	0.01	11
oil	9	0.80	4	0.10	0.05	2
pasta	6	0.75	8	0.10	0.00	2
rice	11	0.20	6	0.15	0.00	2
soda	12	0.50	8	0.10	0.10	2
ladle	9	0.80	6	0.40	0.00	1
toy	3	0.80	12	0.50	0.01	0
egg	2	0.80	12	0.05	0.10	2
spray bottle	6	0.50	4	0.33	0.00	6
salt shaker	4	0.50	2	0.25	0.00	3
wine bottle	7	0.40	6	0.10	0.10	3
potato	1	0.75	8	0.10	0.00	3
pencil	4	0.50	12	0.50	0.01	11
soap bottle	7	0.50	4	0.10	0.01	4
plate	7	0.80	12	0.40	0.00	9
fork	6	0.80	8	0.20	0.20	3
book	7	0.80	20	0.20	0.01	14
pan	9	0.75	6	0.20	0.05	2
towel roll	2	0.75	4	0.25	0.00	3
butter knife	2	0.75	4	0.20	0.01	3
spoon	9	0.75	16	0.20	0.00	2
watch	4	0.50	2	0.40	0.00	8
phone	7	0.75	2	0.90	0.00	13
pen	6	0.75	8	0.50	0.00	9
credit card	4	0.50	4	0.20	0.10	9
candle	8	0.60	12	0.40	0.20	6
tissue box	6	0.20	4	0.10	0.10	5
newspaper	8	0.60	5	0.60	0.20	12
remote control	4	0.75	4	0.75	0.00	12
house plant	8	0.75	12	0.01	0.01	8
laptop	13	0.60	4	0.75	0.10	12
desk lamp	4	0.50	4	0.01	0.01	7
alarm clock	4	0.20	1	0.01	0.00	13
soap bar	7	0.50	4	0.15	0.01	2
toilet paper	1	0.50	4	0.10	0.00	1
baseball bat	2	0.20	1	0.20	0.00	8
dish sponge	7	0.80	4	0.10	0.01	4

tennis racket	1	0.25	4	0.20	0.00	5
basket ball	1	0.20	1	0.20	0.00	11
coffee machine	3	0.60	2	0.01	0.00	2
knife	1	0.60	12	0.20	0.01	2
bread	4	0.50	4	0.14	0.10	2
cup	11	0.80	16	0.40	0.10	3
pot	9	0.50	4	0.10	0.00	4
bottle	11	0.90	15	0.25	0.10	5
toaster	2	0.80	1	0.01	0.00	2
cloth	8	0.90	6	0.20	0.01	2
microwave	2	0.80	1	0.00	0.00	2
apples	2	0.80	15	0.40	0.20	9
oranges	2	0.80	15	0.40	0.20	9
bananas	2	0.80	15	0.40	0.20	9
orange	2	0.80	15	0.40	0.20	9
banana	2	0.80	15	0.40	0.20	9
lemon	2	0.80	15	0.40	0.20	9
garlic	2	0.80	15	0.40	0.20	9
peach	2	0.80	15	0.40	0.20	9
grapes	2	0.80	15	0.40	0.20	9
avocado	2	0.80	15	0.40	0.20	9
towels	11	0.75	8	0.20	0.00	1
beet	1	0.80	5	0.10	0.01	3
radish	1	0.80	5	0.10	0.01	3
eggplant	1	0.80	5	0.10	0.01	3
basil	1	0.80	5	0.10	0.01	3
tomato	1	0.80	5	0.10	0.01	3
kale	1	0.80	5	0.10	0.01	3
squash	1	0.80	5	0.10	0.01	3
yogurt	6	0.80	2	0.10	0.10	2
whole fat milk	6	0.80	2	0.10	0.10	2
zero fat milk	6	0.80	2	0.10	0.10	2
pop	12	0.50	8	0.10	0.10	2
teddy bear	3	0.80	12	0.50	0.01	0
legos	3	0.80	12	0.50	0.01	0
action figure	3	0.80	12	0.50	0.01	0
dinosaur	3	0.80	12	0.50	0.01	0
jigsaw	3	0.80	12	0.50	0.01	0
animal	3	0.80	12	0.50	0.01	0
butter	2	0.80	12	0.05	0.10	2
pepper shaker	4	0.50	2	0.25	0.00	3
paprika shaker	4	0.50	2	0.25	0.00	3
bottle of soap	7	0.50	4	0.10	0.01	4
plates	7	0.80	12	0.40	0.00	9
binder	7	0.80	20	0.20	0.01	14
document	7	0.80	20	0.20	0.01	14
books	7	0.80	20	0.20	0.01	14
binders	7	0.80	20	0.20	0.01	14
documents	7	0.80	20	0.20	0.01	14
spoons	9	0.75	16	0.20	0.00	2
smartphone	7	0.75	2	0.90	0.00	13
wallet	4	0.50	4	0.20	0.10	9
debit card	4	0.50	4	0.20	0.10	9
candles	8	0.60	12	0.40	0.20	6
box of tissues	6	0.20	4	0.10	0.10	5
pc	13	0.60	4	0.75	0.10	12
bar of soap	7	0.50	4	0.15	0.01	2
soap	7	0.50	4	0.15	0.01	2
dish soap	7	0.80	4	0.10	0.01	4
sponge	7	0.80	4	0.10	0.01	4
baguette	4	0.50	4	0.14	0.10	2
bottles	11	0.90	15	0.25	0.10	5