Proceedings of the Fifteenth AAAI Conference on Artificial
Intelligence and Interactive Digital Entertainment (AIIDE-19)

Knowledge-Powered Inference of Crowd Behaviors in Semantically Rich Environments

Xun Zhang,[1] **Davide Schaumann,**[1] **Petros Faloutsos,**[2,3] **Mubbasir Kapadia**[1]
[1]Rutgers University, [2]York University, [3]UHN - Toronto Rehabilitation Institute
[1]Bowser Rd, Piscataway, NJ 08854, United States; [2]4700 Keele Street, Toronto, Ontario, M3J 1P3, Canada
{xz348, ds1540, mk1353}@cs.rutgers.edu; pfal@cse.yorku.ca

Abstract

Interactive authoring of collaborative, context-dependent virtual agent behaviors can be challenging. Current approaches often rely heavily on users' input, leading to cumbersome behavior authoring experiences and biased results, which do not reflect realistic space-people interactions in virtual settings. To address these issues, we generate an ontology graph from commonsense knowledge corpus and use it to automatically infer behavior distributions that determine agents' context-dependent interactions with the built environment. By means of a natural-language interface, users can interactively refine a building's design by adding semantic labels to spaces and populating rooms with equipment following suggestions that the system provides based on commonsense knowledge. Based on the chosen setup, an authoring system automatically populates the environment and allocates agents to specific behaviors while satisfying a behavior distribution inferred from the ontology graph. This approach holds promise to help architects, engineers, and game designers interactively author plausible agent behaviors that reveal the mutual interactions between people and the spaces they inhabit.

Introduction

Authoring multi-agent collaborative behaviors of heterogeneous virtual agents in semantically rich environments can be useful to evaluate building designs, author engaging story arcs in video games, and provide realistic animations of buildings in use. This, however, is a challenging task mainly due to the amount of input required to define context-dependent behaviors. This is a critical issue especially in architectural design where spatial, social and environmental factors affect human behavior (Schaumann et al. 2019). Current behavior authorinen rely heavily on user nput, which can define either (a) *building-centric* occupancy schedules that determine how many agents are located in a given space at a given time (Mahdavi and Tahmasebi 2015), (b) *behavior-centric* agent behaviors that represent the individual or collaborative activities that are performed in semantically rich settings (Kapadia et al. 2016), and (c) *agent-centric* decision-making that determines a set of activities that agents are likely to perform to satisfy individual desires and motivations (Kapadia et al. 2015).

Prior work integrated these approaches into a narrative authoring framework that jointly satisfies building occupancy specifications, behavior distributions, and agent motivations (Zhang et al. 2019). However, this approach relies on the cumbersome, prone-to-error, manual specification of templates to generate plausible behavior distributions.

This work significantly alleviates these issues by proposing a commonsense knowledge approach to automatically infer behavior distributions, given only natural language descriptions of environment semantics. Specifically, a *space-equipment-affordance* ontology graph, automatically derived from commonsense knowledge corpus (CKC), characterizes the relationship between semantically meaningful spaces (e.g., an office or a classroom), the equipment it contains (e.g., desks, chairs), and the inhabitant behavior it affords (e.g., working, meeting). A natural-language interface enables users to interactively modify a building's design by changing semantic labels of spaces and equipment location following suggestions that the system provides based on a generated ontology graph. A narrative synthesis method dynamically allocates agents to a discrete set of behaviors to satisfy the expected behavior distributions.

A case study showcases how users can interactively refine designs based on insights gained from the behavior simulation. While generative design approaches could also be incorporated in this framework, these methods are often limited to abstract space organization and cannot account for the full complexity of building design. For this reason, in this work, we focus on simulating crowd behavior in user-authored building designs. We argue that this approach can facilitate interactive human behavior authoring for architectural design, games and computer animation.

Related Work

We aim to empower content creators (e.g. architects and game designers) with ontologies generated from CKC to facilitate the automated synthesis of agent behaviors while fundamentally accounting for space semantics. In this section, we will briefly review recent work in behavior synthesis and knowledge engineering.

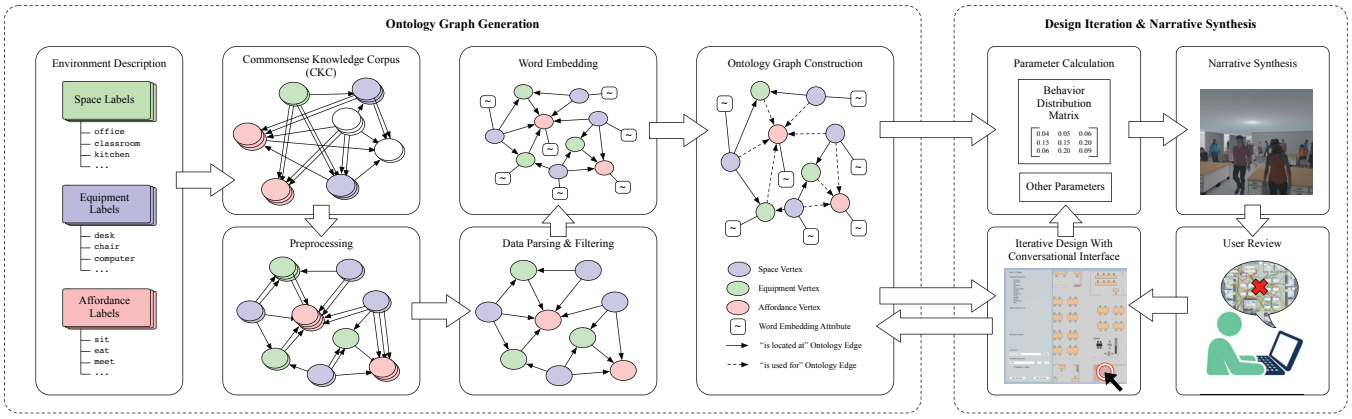

Figure 1: Leveraging commonsense knowledge to infer crowd behaviors in semantically rich environments

Behavior Synthesis. Seminal work on behavior authoring frameworks for intelligent virtual agents proposed the concepts of Smart Objects that encapsulate information on how agents can interact with them (Kallmann and Thalmann 1999; 2002). To create consistent object manipulation information that can be shared between different modelers, Belint and Allbeck connected a taxonomy of actions with operational information on simulation objects using natural language lexical databases and text and knowledge corpora (Balint and Allbeck 2015; 2017). Kapadia et al. (2017) describe a comprehensive framework to generate event-centric behavior atoms which are abstracted in a graphical storyboarding interface and combined to generate larger narratives. Partlan et al. (2018) developed an automated analysis of narrative structure and organization using a multi-graph representation. Zhang et al. (2019) outline a framework to author human behavior narratives that satisfy occupancy specifications and behavior distributions in semantically rich spaces as well as agent motivations. Nevertheless, this approach relies on manually input templates for behavior distributions and does not support interactive design modification and subsequent behavior distribution adaptations. While we rely on the seminal contributions in multi-agent behavior synthesis, such as event-centric modeling (Kapadia et al. 2016), we focus on informing the interactive behavior authoring process with knowledge corpora that capture the mutual relations between spaces, the equipment contained, and the activities afforded.

Knowledge Engineering for Intelligent Behavior Authoring Systems. We seek to apply established tools and methods of knowledge representation and reasoning to inform interactive behavior authoring (Brachman and Levesque 2004). Recent work generates knowledge graphs from events reported in the news to foster user's understanding of a specific domain and facilitate the generation of news storylines (Rospocher et al. 2016). Winer and Young (2017) automatically annotate screenplays to formalize storytelling knowledge. Li et al. (2013) generate behavior narratives by using a domain model automatically learned from a crowdsourced corpus of story examples. These approaches, however, do not support the authoring of behaviors that fundamentally account for the spatial characteristics of built environments.

To address this issue, we turn our attention to Commonsense Knowledge Corpora (CKC) obtained from crowdsourcing and/or expert curation, which can be used to infer how people use spaces. Speer and Havasi (2012) presented a semantic network collecting commonsense knowledge data from various sources. The knowledge is represented as a graph that connects words and phrases of natural language with labeled edges, providing rich information for inferring and constructing a customized knowledge graph to inform human spatial behaviors. Gerz et al. (2016) provide a large empirical library evaluating the similarity among individual pairs of verbs that can be used to describe human behavior in spaces. In this work, we propose a framework that makes novel use of CKC to help generating plausible behaviors of virtual populations in semantically rich environments with an application to architectural design.

Framework Overview

Our framework contains 3 phases (Figure 1): (i) description of the environment to be populated with virtual agents (ii) offline creation and processing of an ontology graph, and (iii) online design modification and narrative synthesis based on information extracted from the ontology graph. We detail these steps in the following sections.

Environment Description. We model the functional properties of an environment independent of the geometry so that it's applicable to different environment designs. An environment $\mathcal{E} =< S, Q, A >$ contains (i) a set S of functional *spaces* where each individual space $s \in S$ is assigned with a semantic *label*; (ii) a set Q of interactable *equipment* that supports different affordances; (iii) a set A of *affordances* that can be performed by agents as they interact with the built environment. For instance, if we consider $\mathcal{E}$ as a research lab in a university, the following entities could be modeled: $S = \{\texttt{office}, \texttt{classroom}, \texttt{kitchen}\}$, $Q = \{\texttt{chair}, \texttt{computer}, \texttt{desk},\}$, and $A = \{\texttt{eat}, \texttt{meet}, \texttt{talk}\}$.

Ontology Graph Generation. An ontology graph is built based on CKC (Speer, Chin, and Havasi 2016) and a nat-

ural language parser (Manning et al. 2014). Parsing, filtering, and word embedding techniques are applied to improve the quality of the information. The heterogeneous attributed graph $G =< V, E >$ contains 3 types of vertices and 2 types of weighted directed edges. The vertices are (i) equipment vertices V_q, which represent equipment instances $q \in Q$; (ii) space vertices V_s, which represents the spaces $s \in S$; (iii) affordance vertices V_a, which represents the affordances $a \in A$. Vertices are linked using 2 types of unidirectional edges: (i) Edges E_s that represent the ontology of "q is located at s" where $q \in Q$ and $s \in S$; (ii) Edges E_a that represent the ontology of "sq is used for a" where $sq \in S \cup Q$ and $a \in A$.

Design Iteration and Narrative Synthesis. The ontology graph informs the interactive modification of designs by suggesting possible equipment to allocate in a space based on the semantic label input by the user. After completing a design configuration, the graph is used to calculate a behavior distribution in semantically rich spaces. The behavior distribution is used to synthesize a plausible narrative of inhabitants in the space while conforming to the specifications and constraints of the behavior distribution.

Ontology Graph Generation

Our framework considers a list of space, equipment and affordance labels as the initial input for a given environment. The ontology graph is constructed by the following steps: (i) starting from the space labels, we obtain from the CKC the equipment located in the different space types, (ii) obtain from the equipment a list of affordances that the equipment supports, (iii) obtain from the space a list of affordances it can host, (iv) for each affordance and equipment vertex, find the highest similarity between it and the local equipment and affordance library, and use it as a mapping from commonsense to local library.

Commonsense Knowledge Corpus. ConceptNet (Speer, Chin, and Havasi 2016) as an open-source large semantic network provides a large variety of commonsense knowledge data. Each vertex in the database represents a concept entity, and the vertices are interconnected with relation edges that describe the relationship between two vertices. Based on the connections of the vertices in the database, in some cases, ConceptNet provides a rather complete knowledge of the objects presenting in people's life. For instance, it contains relation edges that enumerate objects located in a place, and also enumerates the supported interactions of each object, which helps to construct and estimate the place-object-interaction relationship. However, the raw data stored in ConceptNet are collected from various sources with limited consistency, and some relation edges are significantly biased. Hence the information obtained from it need to be further processed to minimize these drawbacks.

Parsing and Filtering Noisy Data. Raw data obtained from ConceptNet contains the following issues: (i) duplicated entries that describe the same object, but stored as different words; (ii) metadata from the original data resource; (iii) spelling and other issues. Our framework fixes these issues by parsing the data by natural language annotation. First, we built a metadata trimmer to remove all superfluous information in the metadata while keeping useful semantics such as if the word is a verb or a noun for ambiguous cases. Then a proofreader removes any misspelled words. Eventually, the extracted word with proper metadata are passed through a part-of-speech and lemma annotator to extract the clean, singular form for nouns, or original form for verbs. This process is applied for each vertex before adding it to the ontology graph and creating edges with other vertices to maintain the consistency of the ontologies.

Algorithm 1: Ontology graph construction and processing. Note for different types of vertices the edges.

input : space vertices S
output: ontology graph G

$V \leftarrow \emptyset, E \leftarrow \emptyset, G \leftarrow < V, E >$
foreach $s \in S$ **do**
  $V \leftarrow V \cup \{s\}$
  foreach $q \in Equipment(s)$ **do**
    if $q \notin V$ **then**
      $V \leftarrow V \cup \{q\}$
      $E \leftarrow E \cup \{< q, s, w_{q,s} >\}$
      foreach $a \in Affordance(q)$ **do**
        if $a \notin V$ **then**
          $V \leftarrow V \cup \{a\}$
          $E \leftarrow E \cup \{< q, a, w_{q,a} >\}$
        else
          $< q, a, w > \leftarrow < q, a, w + w_{q,a} >$
    else
      $< q, s, w > \leftarrow < q, s, w + w_{q,s} >$
  foreach $a \in Affordance(s)$ **do**
    if $a \notin V$ **then**
      $V \leftarrow V \cup \{a\}$
      $E \leftarrow E \cup \{< s, a, w_{a,s} >\}$
    else
      $< s, a, w > \leftarrow < s, a, w + w_{a,s} >$
$G \leftarrow MergeEquipmentVertices(G)$
$G \leftarrow MapVertices(G)$
return G

Word Embedding. There is a disparity between the words used by CKC and the labels used by the designer when specifying environment semantics or population behavior. For instance, the designer specifies 3 core affordances supported by the narrative synthesis engine: `work`, `eat`, and `talk`. However, the CKC contains several more affordance entries which may relate to the same core affordances, while having different labels. Hence, we first need to develop a mapping of entity labels extracted from CKC to labels used in the narrative engine. We resolve this issue with word embedding (Mikolov et al. 2013) and use the cosine similarity between the affordances extracted from CKC and the labels provided by the designer. For each affordance vertex in the ontology graph, we assign a property that stores the matched local affordance. During the simulation, we can group the affor-

dance weights by accumulating all the matched affordance in the library, and use these weights for distribution estimation. This holds true for the equipment vertices, i.e., techniques for processing affordance vertices are also applicable to equipment vertices.

Algorithm 2: Mapping vertices (line 23 of algorithm 1)

input : ontology graph G, target equipment library $\hat{Q}$, target affordance library $\hat{A}$
output: processed ontology graph G'

$G' \leftarrow G$
$Q \leftarrow Equipment(G)$
$A \leftarrow Affordance(G)$
foreach $a \in A$ **do**
  $h \leftarrow -1$
  foreach $\hat{a} \in \hat{A}$ **do**
    $s \leftarrow CosineSimilarity(a, \hat{a})$
    if $s > h$ **then**
      $h \leftarrow s, a_h \leftarrow \hat{a}$
  $AddMapping(G', a, a_h, h)$
foreach $q \in Q$ **do**
  $h \leftarrow -1$
  foreach $\hat{q} \in \hat{Q}$ **do**
    $s \leftarrow CosineSimilarity(q, \hat{q})$
    if $s > h$ **then**
      $h \leftarrow s, q_h \leftarrow \hat{q}$
  $AddMapping(G', q, q_h, h)$
return G'

Ontology Graph Construction. The ontology graph is constructed using the aforementioned techniques, starting with the space vertices, which corresponds to the spaces (e.g., rooms in an office building) in the virtual environment. Using CKC, all of the equipment that is likely to be located in the space will be extracted with corresponding weights. We then search the CKC for the affordances that spaces and the corresponding equipment can support. The graph construction algorithm is shown in algorithm 1.

Line 4–15 creates the equipment vertices obtained by querying the commonsense knowledge base with the space vertices and links them correspondingly. Furthermore, line 8–13 links the equipment vertices with queried affordance vertices. Then line 16–21 does the same for the affordances obtained by querying space vertices. After these steps, a preliminary ontology graph is constructed.

Before proceeding to utilize the constructed ontology graph for the simulation, 2 more steps are required (line 22 and 23): (i) cleaning up and merging the equipment vertices due to the noise in the CKC, and (ii) mapping the raw affordance vertices to the affordance animation library in the simulation engine, which is a smaller library compared to the raw affordance data. These 2 steps are shown in algorithm 3 and 2, respectively. Line 10 and line 17 in algorithm 2 respectively attach the mapped result to the original affordance and equipment as an attribute (as is shown in the "Word Embedding" block in Figure 1).

Algorithm 3: Merging equipment vertices (line 22 of algorithm 1)

input : ontology graph $G =< V, E >$
output: processed ontology graph G

foreach *equipment* $q \in V$ **do**
  $q_l \leftarrow ParseLemma(q), V \leftarrow V \setminus \{q\}$
  if $q_l \notin V$ **then**
    $V \leftarrow V \cup \{q_l\}$
    foreach $s \in Space(q)$ **do**
      $E \leftarrow E \cup \{< q_l, s, w_{q_l,s} >\}$
    foreach $a \in Affordance(q)$ **do**
      $E \leftarrow E \cup \{q_l, a, w_{q_l,a}\}$
  else
    foreach $s \in Space(q)$ **do**
      $f \leftarrow 0$
      foreach $s' \in Space(q_l)$ **do**
        if $s = s'$ **then**
          $f \leftarrow f + 1$
          $< q_l, s', w >\leftarrow< q_l, s', w + w_{q_l,s} >$
      if $f = 0$ **then**
        $E \leftarrow E \cup \{< q_l, s, w_{q_l,s} >\}$
    foreach $a \in Affordance(q)$ **do**
      $f \leftarrow 0$
      foreach $a' \in Affordance(q_l)$ **do**
        if $a = a'$ **then**
          $f \leftarrow f + 1$
          $< q_l, a', w >\leftarrow< q_l, a', w + w_{q_l,a} >$
      if $f = 0$ **then**
        $E \leftarrow E \cup \{< q_l, a, w'_{q_l,a} >\}$
return G

Design Iteration and Narrative Synthesis

After finishing the offline commonsense knowledge processing, the user can iteratively refine a building design and synthesize behavior narratives.

Iterative Design via Conversational Interaction. After specifying a geometrical building layout and equipment models which are associated to the labels specified in the offline phase, the user can interactively label spaces semantically and add or remove equipment following suggestions provided by the ontology graph.

We provide an intuitive conversational interface where the user can give simple natural language guidelines to provide additional semantics of the space, its equipment, and what behavior is expected. For example, the user can set the semantics of the space by selecting a space and saying "this room is an office room." Or, "this room can be used for working." We used the same natural language parser that is used for lemma extraction during the parsing and filtering process of noisy CKC data. Using the part-of-speech and dependency information, we estimate the user command type (either setting space label, or equipment expectation, or behavior expectation), then use the extracted entity to set the corresponding properties.

During the design and interaction process, our system

uses the current semantic labels provided by the user to provide "prompts" or suggestions of possible design interventions which may be appropriate for the current design. This is powered using the ontology graph generated. For example, the system uses the space and/or affordance labels provided by the designer to suggest plausible equipment instances which may be required to furnish the room. The idea is, based on the space label specified, the framework queries our ontology graph to fetch all the supported equipment with proper weights. On the other hand, after instantiating equipment into the space, we query the ontology graph again to acquire all the affordances supported.

Behavior Distribution Calculation. Based on the current design configuration, the system generates a behavior distribution for each space using Algorithm 4, which takes the generated ontology graph as the main input. Other parameters include the environment setup (e.g., space labels, furnishing) and the affordance library (i.e., agent animations). In line 1 of Algorithm 4, we use 2 matrices, $\mathbf{M}$ and $\mathbf{M}'$, to store the final calculated behavior distribution in a space and a temporary behavior distribution based on equipment furnishing, respectively. The shape of the matrix is $m \times n$, where m is the number of spaces and n is the number of behaviors. Each entry in the matrices shows the estimated behavior distribution in the corresponding space. In line 2–4, we generate an initial estimate of the behavior distribution given by the space-affordance ontology edges, and we store this distribution in $\mathbf{M}$. From line 6 to 11, the equipment instances in the spaces are taken into consideration and by accumulating weights of all the equipment-affordance ontology edges, we generate another space-affordance matrix, $\mathbf{M}'$. We use 2 distribution matrices and eventually merge them due to the biased information given by either space-affordance edges or equipment-affordance edges. Therefore, we use $\mathbf{M}$ to set up a baseline of the affordances, then use $\mathbf{M}'$ to finetune it (line 12).

Algorithm 4: Generating the behavior distribution with the ontology graph

input : ontology graph G, environment spaces $\hat{S}$, affordance library $\hat{A}$, weighting factor w
output: distribution matrix $\mathbf{M}$

1 $\mathbf{M} \leftarrow \mathbf{0}, \mathbf{M}' \leftarrow \mathbf{0}$
2 **foreach** $s \in \hat{S}$ **do**
3 **foreach** $a \in \hat{A}$ **do**
4 $\mathbf{M}(s, a) \leftarrow w_{s,a}$
5 $Q \leftarrow Equipment(G)$
6 **foreach** $q \in Q$ **do**
7 $s \leftarrow Space(q)$
8 **foreach** $s \in S$ **do**
9 $A \leftarrow Affordance(q)$
10 **foreach** $a \in \hat{A}$ **do**
11 $\mathbf{M}'(s, a) \leftarrow \mathbf{M}'(s, a) + w_{e,s} w_{e,a}$
12 $\mathbf{M} \leftarrow w\mathbf{M} + (1 - w)\mathbf{M}'$
13 **return** $\mathbf{M}$

Narrative Synthesis. The behavior distribution matrix is combined with additional input to author multi-agent narratives using an established narrative synthesis approach (Zhang et al. 2019). The additional input consists of (i) occupancy distribution, which defines how many agents are expected to be located in a specific room at a certain time, (ii) agent motivations, which determine a preferred activity for the agent at a given time, and (iii) a library of Parametric Behavior Trees (PBTs), a behavior authoring system that enables to coordinate the activities of virtual agents as they interact with the built environment (Shoulson et al. 2011). An optimization approach allocates agents to a discrete number of activities (modeled as PBTs) while satisfying the occupancy specification, behavior distribution, and agent motivations. This optimization process is repeated at adaptive time steps to minimize the difference between the expected behavior distributions and the actual ones (Zhang et al. 2019).

User Review and Refinement. Based on the simulation output, the user can evaluate the extent to which a building design supports the activities of its intended inhabitants. The simulation outcomes can reveal design issues including spatial bottlenecks, under-used spaces or resources (e.g., desks and chairs), and crowded areas. This analysis can inform additional design iterations aimed at modifying equipment location, semantic labels, and the geometrical layout.

Demonstration and Evaluation

Figure 2 shows the capability of our approach to supporting the design and evaluation of a lab design. The user starts by using the conversational interface to add semantics to the rooms which have not been labeled yet (Figure 2a). Semantic labels include "office", "classroom", "kitchen" and "meeting room". Based on the input semantic, the system automatically suggests which types of furniture to add into the room, such as "chairs", "tables" and "shelves" (Figure 2b). This operation is repeated until all rooms have been semantically labeled and furnished (Figure 2c). At this stage, the user initiates the simulation process. Our system automatically generates behavior distributions using the previously generated ontology graph. Possible behaviors include "eat", "meet", "work" and "lecture". The behavior distribution is used as input together with predefined occupancy specification and user motivations to author an interactive multi-agent narrative. Based on the simulation output, the user discovers that the classrooms are significantly congested (Figure 2d). To address this issue, the user changes the semantics of a "meeting room" into a "classroom" (Figure 2e) and refurnishes the room accordingly (Figure 2f). The new simulation shows reduced congestion in classrooms and slightly increased occupancy in the remaining meeting room. The walking paths of all the agents are visualized to reveal agents' movement patterns (Figure 2f).

Table 1 compares the affordance distributions across design iterations. Table 1(a) shows the behavior distribution when some of the spaces are unlabeled. The table rows show the affordance distribution within the space, while the columns represent the affordance distribution among spaces. The matrix is normalized at the environment level and the af-

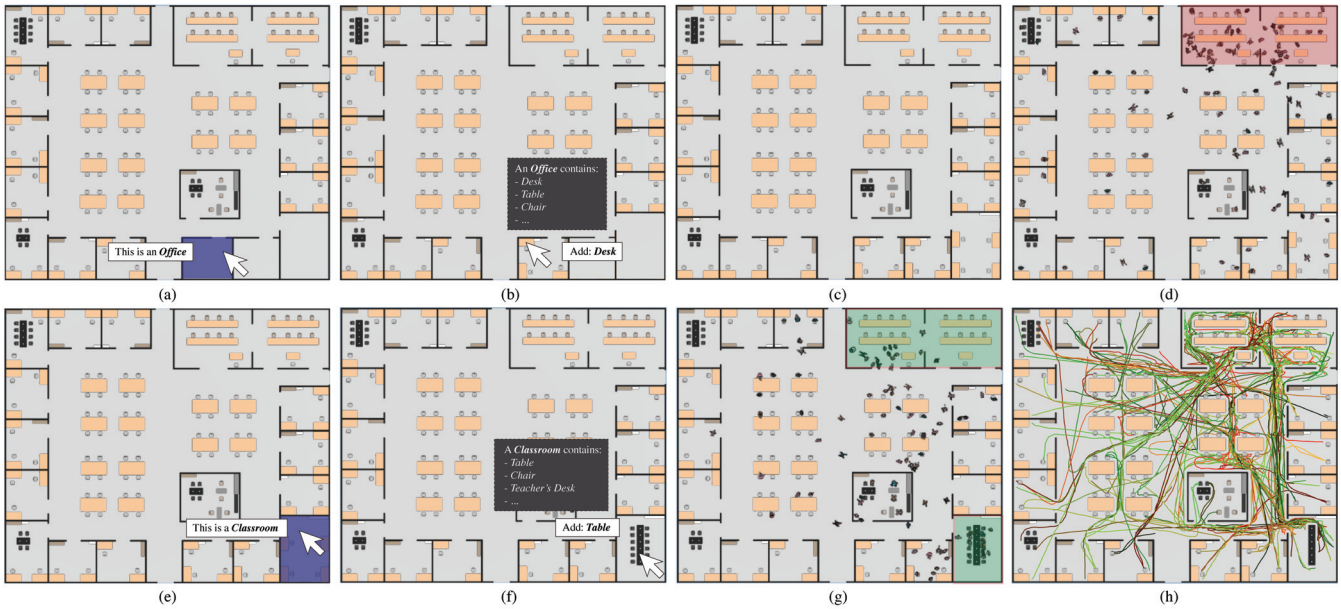

Figure 2: Different steps of an iterative design refinement using a conversational interface for narrative authoring.

	#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	Type	O	O	O	O	O	O	M	O	O	C	C	O	O	O	—	—	—	K	W	W
	W	1.4	0.8	0.9	1.6	0.9	1.0	1.6	2.0	0.9	5.5	5.0	2.6	1.0	1.2	0.0	0.0	0.0	1.1	8.6	5.9
(a)	Eat	1.0	0.6	0.7	1.2	0.7	0.8	1.2	1.5	0.7	5.0	4.5	1.9	0.8	0.9	0.0	0.0	0.0	1.6	4.6	3.2
	Meet	0.4	0.2	0.2	0.4	0.2	0.3	0.4	0.6	0.2	2.1	1.9	0.7	0.3	0.3	0.0	0.0	0.0	0.3	2.2	1.5
	Lecture	0.2	0.1	0.1	0.2	0.1	0.2	0.2	0.3	0.1	5.3	4.8	0.4	0.2	0.2	0.0	0.0	0.0	0.2	1.4	0.9
	Type	O	O	O	O	O	O	**M**	O	O	C	C	O	O	O	**M**	**O**	**O**	K	W	W
	W	1.2	0.7	0.8	1.3	0.8	0.9	***1.4***	1.7	0.7	***4.4***	***4.1***	2.2	0.9	1.0	***1.2***	1.8	2.0	1.0	7.4	5.3
(b)	Eat	0.9	0.5	0.6	1.0	0.6	0.7	1.0	1.3	0.6	4.0	3.8	1.7	0.7	0.8	2.5	1.3	1.5	1.4	4.0	2.9
	Meet	0.3	0.2	0.2	0.4	0.2	0.2	0.4	0.5	0.2	1.8	1.6	0.6	0.2	0.3	2.1	0.5	0.5	0.3	1.9	1.4
	Lecture	0.2	0.1	0.1	0.2	0.1	0.1	0.2	0.3	0.1	4.4	4.1	0.4	0.1	0.2	0.4	0.3	0.3	0.2	1.2	0.5
	Type	O	O	O	O	O	O	**C**	O	O	C	C	O	O	O	M	O	O	K	W	W
	W	1.2	0.7	0.8	1.4	0.8	0.9	***2.9***	1.8	0.8	***2.7***	***3.6***	2.3	0.9	1.1	***2.9***	1.6	1.9	1.0	7.6	5.3
(c)	Eat	0.9	0.5	0.6	1.0	0.6	0.7	2.2	1.3	0.6	2.5	3.3	1.7	0.7	0.8	2.6	1.2	1.4	1.5	4.1	2.8
	Meet	0.3	0.2	0.2	0.4	0.2	0.2	0.8	0.5	0.2	1.1	1.4	0.6	0.2	0.3	1.1	0.4	0.5	0.3	2.0	1.4
	Lecture	0.2	0.1	0.1	0.2	0.1	0.1	0.5	0.3	0.1	2.7	3.6	0.4	0.1	0.2	2.8	0.3	0.3	0.2	1.2	0.9

Table 1: Behavior distribution changes (in %) corresponding to changes in space semantics for each step of the study (a, b, c). Semantic changes are in gray. Relevant distribution changes are in bold, italic gray. Space types: O: office, M: meeting room, C: classroom, K: kitchen, W: work space.

fordance variance is a result of the ontologies combined with current space setup and the occupancy specification. Table 1(b) shows the behavior distributions after labeling all the spaces. Table 1(c) reveals the changes in behavior distributions after modifying a "meeting room" into a "classroom".

Conclusion and Future Work

We presented a behavior synthesis framework that helps architects and virtual world builders author multi-agent narratives while leveraging commonsense knowledge which captures the mutual relations between spaces, the equipment contained and the activities afforded. Our system enables users to iteratively modify building designs using a conversational interactive interface and author narratives that reveal to what extent a space supports the activities of the building inhabitants. Our framework uses commonsense knowledge encoded in ConceptNet (Speer, Chin, and Havasi 2016), which can be limited and sometimes biased. Additionally, it requires manual input of occupancy specifications and user motivations, and it considers a single semantic for each space at a given time. Future work will involve a deeper evaluation of the usability and efficacy of the tool when used by professionals. Furthermore, we aim to develop a design chatbot that helps designers test different behavior scenarios and proactively suggests design modifications to optimize a building's operational efficiency and spatial utilization.

References

Balint, T., and Allbeck, J. M. 2015. Automated generation of plausible agent object interactions. In *International Conference on Intelligent Virtual Agents*, 295–309. Springer.

Balint, J. T., and Allbeck, J. 2017. Alet: Agents learning their environment through text. *Computer Animation and Virtual Worlds* 28(3-4):e1759.

Brachman, R. J., and Levesque, H. J. 2004. *Knowledge Representation and Reasoning*. Elsevier.

Gerz, D.; Vulic, I.; Hill, F.; Reichart, R.; and Korhonen, A. 2016. Simverb-3500: A large-scale evaluation set of verb similarity. *CoRR* abs/1608.00869.

Kallmann, M., and Thalmann, D. 1999. Direct 3d interaction with smart objects. In *Proceedings of the ACM symposium on Virtual reality software and technology*, 124–130. ACM.

Kallmann, M., and Thalmann, D. 2002. Modeling behaviors of interactive objects for real-time virtual environments. *Journal of Visual Languages & Computing* 13(2):177–195.

Kapadia, M.; Pelechano, N.; Allbeck, J.; and Badler, N. 2015. Virtual crowds: Steps toward behavioral realism. *Synthesis lectures on visual computing: computer graphics, animation, computational photography, and imaging* 7(4):1–270.

Kapadia, M.; Shoulson, A.; Steimer, C.; Oberholzer, S.; Sumner, R. W.; and Gross, M. 2016. An event-centric approach to authoring stories in crowds. In *Proceedings of the 9th International Conference on Motion in Games*, 15–24. ACM.

Kapadia, M.; Poulakos, S.; Gross, M.; and Sumner, R. W. 2017. Computational Narrative. In *ACM SIGGRAPH 2017 Courses*, SIGGRAPH '17, 4:1–4:118. New York, NY, USA: ACM. event-place: Los Angeles, California.

Li, B.; Lee-Urban, S.; Johnston, G.; and Riedl, M. 2013. Story Generation with Crowdsourced Plot Graphs. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Mahdavi, A., and Tahmasebi, F. 2015. Predicting people's presence in buildings: An empirically based model performance analysis. *Energy and Buildings* 86:349–355.

Manning, C. D.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S. J.; and McClosky, D. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, 55–60.

Mikolov, T.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Efficient estimation of word representations in vector space.

Partlan, N.; Carstensdottir, E.; Snodgrass, S.; Kleinman, E.; Smith, G.; Harteveld, C.; and El-Nasr, M. S. 2018. Exploratory automated analysis of structural features of interactive narrative. In *Fourteenth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Rospocher, M.; van Erp, M.; Vossen, P.; Fokkens, A.; Aldabe, I.; Rigau, G.; Soroa, A.; Ploeger, T.; and Bogaard, T. 2016. Building event-centric knowledge graphs from news. *Journal of Web Semantics* 37-38:132–151.

Schaumann, D.; Pilosof, N. P.; Sopher, H.; Yahav, J.; and Kalay, Y. E. 2019. Simulating multi-agent narratives for pre-occupancy evaluation of architectural designs. *Automation in Construction* 106.

Shoulson, A.; Garcia, F. M.; Jones, M.; Mead, R.; and Badler, N. I. 2011. Parameterizing behavior trees. In Allbeck, J. M., and Faloutsos, P., eds., *Motion in Games*, 144–155. Berlin, Heidelberg: Springer Berlin Heidelberg.

Speer, R., and Havasi, C. 2012. Representing general relational knowledge in conceptnet 5. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, 3679–3686. Istanbul, Turkey: European Language Resources Association (ELRA).

Speer, R.; Chin, J.; and Havasi, C. 2016. Conceptnet 5.5: An open multilingual graph of general knowledge. *Proceedings of 31St AAAI Conference on Artificial Intelligence*.

Winer, D. R., and Young, R. M. 2017. Automated Screenplay Annotation for Extracting Storytelling Knowledge. In *Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Zhang, X.; Schaumann, D.; Haworth, B.; Faloutsos, P.; and Kapadia, M. 2019. Coupling agent motivations and spatial behaviors for authoring multi-agent narratives. In *32nd International Conference on Computer Animation and Social Agents*.