USING MULTI-RELATIONAL EMBEDDINGS AS KNOWLEDGE GRAPH REPRESENTATIONS FOR ROBOTICS APPLICATIONS

A Dissertation
Presented to
The Academic Faculty

By

Angel Andres Daruna

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering
College of Engineering

Georgia Institute of Technology

August 2022

USING MULTI-RELATIONAL EMBEDDINGS AS KNOWLEDGE GRAPH REPRESENTATIONS FOR ROBOTICS APPLICATIONS

Thesis committee:

Dr. Sonia Chernova, Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Matthew Gombolay
School of Interactive Computing
Georgia Institute of Technology

Dr. Zsolt Kira
School of Interactive Computing
Georgia Institute of Technology

Dr. Devi Parikh
School of Interactive Computing
Georgia Institute of Technology

Dr. Mohan Sridharan
School of Computer Science
University of Birmingham, UK

Date approved: May 4, 2022

ACKNOWLEDGMENTS

I would like to thank the many people who made this thesis possible through their support. First and foremost I would like to thank my advisor, Sonia Chernova, who's advice guided me to the completion of my dissertation. I would like to thank the members of my thesis committee for their help in preparation of this work – Zsolt Kira, Mohan Sridharan, Matthew Gombolay, and Devi Parikh. Special thanks are due to the friends and colleagues who made this work possible. Weiyu Liu, whom I collaborated with across many works. My lab mates, which included but not limited to, Harish Ravichandar, Lakshmi Nair, Devleena Das, Sid Banerjee, David Kent, Asif Rana, Joanne Truong, Vivian Chu, Tesca Fitzgerald, Kalesha Bullard, Glen Neville, Jonathan Balloch, and Reza Ahmadzadeh. Lastly, I thank my family and friends whom inspired me to go above and beyond what was expected.

The author gratefully acknowledges the support for this work offered by the National Science Foundation under GRFP-DGE-1650044, by the GEM Fellowship, and by the Goizueta Fellowship. Any views and conclusions contained herein are those of the author, and do not necessarily represent the official positions, express or implied, of the funders.

TABLE OF CONTENTS

Acknowledgments iii

List of Tables ix

List of Figures x

List of Acronyms xiii

Summary xv

Chapter 1: Introduction 1

1.1 Dissertation Overview 3

1.2 Thesis Statement 4

1.3 Contributions 4

1.4 Outline of Dissertation Document 6

Chapter 2: Background and Related Work 7

2.1 Knowledge Graphs 7

2.2 Multi-Relational Embeddings 8

2.3 State of the art in Link Prediction 12

2.4 Continual Learning 13

2.5 Explainable AI Planning (XAIP) 15

2.5.1 Explainable Knowledge Base Completion 16

2.6 Robot Task Execution Stack 18

Chapter 3: Survey of KG Representations in Robotics 22

3.1 Introduction 22

3.2 Directed Graphs 24

3.3 Markov Network 26

3.4 Bayesian Networks 27

3.5 Partially Directed Acyclic Graphs 28

3.6 Logics 30

3.7 First-order Probabilistic Models 32

3.8 Neural Networks 34

3.9 Summary of KG Representations 36

3.10 Discussion & Conclusion 37

3.11 Findings & Contributions 38

Chapter 4: Multi-Relational Embeddings as KG Representations for Robots . . 39

4.1 Introduction 39

4.2 Background: Multi-Relational Embeddings 41

4.3 RoboCSE 42

4.4 Experimental Procedure 45

4.4.1 RoboCSE Knowledge Source: AI2Thor 45

4.4.2 Inferring Unseen Triples: Triple Generalization 46

4.4.3 Applying Common Sense: Environment Generalization 47

4.5 Experimental Results 48

4.5.1 Testing Triple Generalization 49

4.5.2 Testing Environment Generalization 51

4.5.3 Domain Transfer: Testing on MatterPort3D 52

4.5.4 Analyzing Memory Requirements 53

4.5.5 Implementation on a Mobile Manipulator 54

4.6 Discussion & Conclusion 55

4.7 Findings & Contributions 55

Chapter 5: Robust Task Execution using KG Representations 57

5.1 Introduction 57

5.2 Problem Definition 60

5.3 Approach 60

5.3.1 Knowledge Representation 61

5.3.2 Task Generalizations 62

5.3.3 Task Generalization Module 64

5.4 Experimental Evaluation 66

5.4.1 Comparison of Knowledge Representations 67

5.4.2 Ablation Studies of Task Generalization Algorithm 70

5.4.3 Validation on a Robot Platform 72

5.5 Discussion & Conclusion 73

5.6 Findings & Contributions 74

Chapter 6: Continual Knowledge Graph Embedding 76

6.1 Introduction 76

6.2 Background 78

6.3 Approaches to Continual Knowledge Graph Embedding 79

6.3.1 Architectural Modification Methods 80

6.3.2 Regularization Methods 81

6.3.3 Generative Replay Methods 82

6.4 Experimental Setup for Link-Prediction Evaluations 84

6.5 Experimental Results for Link-Prediction Evaluations 90

6.5.1 Benchmark Datasets: 91

6.5.2 Robotics Datasets: 94

6.6 Experimental Setup for Robust Task Execution 95

6.7 Experimental Results for Robust Task Execution 98

6.8 Discussion & Conclusion 101

6.9 Findings & Contributions 102

Chapter 7: Explainable Knowledge Graph Embedding 104

7.1 Introduction 104

7.2 Methodology 107

7.2.1 Knowledge Graph Representation 108

7.2.2 Interpretable Model 109

7.2.3 Extracting and Grounding Explanations 111

7.3 Experimental Evaluations 112

7.3.1 Evaluation of Interpretable Graph Feature Model 112

7.3.2 Evaluation of Explanation Preferences 116

7.3.3 Validation of Explanations for Downstream Tasks 119

7.4 Discussion & Conclusion 125

7.5 Findings & Contributions 126

Chapter 8: Conclusions and Future Work 127

8.1 Summary of Contributions 127

8.1.1 Survey of KG Representations in Robotics 127

8.1.2 Multi-Relational Embeddings as KG Representations for Robots . . 128

8.1.3 Robust Task Execution using KG Representations 128

8.1.4 Continual Knowledge Graph Embedding 128

8.1.5 Explainable Knowledge Graph Embedding 129

8.2 Open Questions 130

8.2.1 Semantic Reasoning using MREs as KG Representations Supporting Robots 130

8.2.2 Semantic Reasoning using KG Representations Supporting Robots . 131

8.2.3 Semantic Reasoning Supporting Robots 132

Appendices 133

Appendix A: Extended CKGE Experiments with Alternate Sampling Strategies . 134

References 142

LIST OF TABLES

4.1 RoboCSE Knowledge Source 45

5.1 Dataset extracted from VirtualHome to learn $\mathcal{G}$ 63

5.2 Metrics for 12,000 Environment Executions 68

5.3 Ablation Success Rates in 6 E_x Types 70

6.1 CKGE Datasets; Benchmarks 86

6.2 CKGE Datasets; Robotics 86

6.3 VirtualHome Dataset; 6 Households 96

6.4 VirtualHome Household Triple Overlap 96

7.1 Comparison of Explainable KBC Methods 113

7.2 Dataset gathered from VirtualHome to learn Θ 114

7.3 Preferences User Study Example Questions and Responses 117

7.4 Results of Preferences User Study 119

7.5 Incorrect Robot Generalization Behaviors Corrected by Non-experts 124

A.1 CKGE Datasets; Entity Sampling Benchmarks 136

A.2 CKGE Datasets; Relation Sampling Benchmarks 136

LIST OF FIGURES

2.1 Mobile manipulation system overview. Arrows denote ROS information flow, through publishers, subscribers, services, and actionlib. 19

2.2 Overview of the two packages in our the executive level. Arrows denote ROS information flow, through publishers, subscribers, services, and actionlib. 21

3.1 An example directed graph that includes multimodal data, different types of relations, and local confidence values defined for edges (shown as values in parentheses). The confidence scale is not bounded. 23

3.2 An example Markov network with undirected edges between random variables. The joint distribution can be factorized into clique potentials, denoted by ϕ's. . . . 25

3.3 An example Bayesian network with directed edges between random variables. The joint distribution can be factorized into conditional probabilities, denoted by P's. . 27

3.4 An example partially directed acyclic graph with both directed and undirected edges between random variables. The edge between the two nodes in the same chain component is undirected, while the edges between two nodes in different chain components are directed. The joint distribution can be factorized into conditional probabilities of chain components given their respective parents. 29

3.5 A graphical representation of an example description logic axioms. The T-Box contains axioms defining relations between class-level concepts. The A-Box contains a single fact, which is governed by the constraints defined in T-Box. 30

3.6 An example Markov logic network and its grounded Markov network. Each formula defining the Markov logic network has an associated weight that reflects how strong a constraint it is. The Markov network has 3 grounded atom: obj1, loc1, and loc2. 32

3.7 An example neural network word embedding where proximity of semantic meaning can be determined by vector space computations. Entities are vectors and relations can be embedded as vector offsets. 35

5.1 The task generalization module incrementally generalizes failed task plans by leveraging the learned KG to infer plan constituents (see section 5.3). The task execution module sequentially executes primitive actions to complete the task plan. 58

5.2 Moving average success rate of 60 executions for all tested task generalization approaches. 69

5.3 Ablation queue size tuning heatmap showing success rate vs number of attempts. . 71

5.4 Sample test environment set up similar to a household setting. Test objects shown in top right, from left to right: feather duster, towel rolled, scrubber, duster, and washing sponge. 73

6.2 Measures averaged for all datasets in Table 6.1 and graph embedding representations in section 6.4. Hits@10 used for ACC, FWT, +BWT, and REM. Best viewed in color. 90

6.4 Semantics-driven robotics application scenarios: (a) Unconstrained; (b) Data Constrained; and (c) Time and Data Constrained. Best viewed in color. 92

6.5 Semantics-driven robotics application scenarios: (a) Unconstrained; (b) Data Constrained; and (c) Time and Data Constrained. In each line plot, shading indicates standard deviation. Best viewed in color. 93

6.6 Robot Task Execution Performance in (a) Unconstrained and (b) Data Constrained scenarios. Best viewed in color. 99

7.1 Overview of our inference reconciliation framework which introduces a novel explainable KG embedding method (graph feature model), leveraging decision trees, to provide natural language explanations to users. 106

7.2 Preferences User Study GUI. 116

7.3 Non-expert Feedback User Study GUI. 121

7.4 Effect of Non-expert Correction Accuracy on MRR%. 122

A.1 Measures averaged for all datasets in Table A.1 and graph embedding representations in section 6.4. Hits@10 used for ACC, FWT, +BWT, and REM. Best viewed in color. 138

A.2 Measures averaged for all datasets in Table A.2 and graph embedding representations in section 6.4. Hits@10 used for ACC, FWT, +BWT, and REM. Best viewed in color. 139

LIST OF ACRONYMS

ACC	Average accuracy
AMT	Amazon's Mechanical Turk
ASP	Answer Set Programming
BLN	Bayesian Logic Networks
BN	Bayesian Network
BWT	Backwards Transfer
CKGE	Continual Knowledge Graph Embedding
CRF	Conditional Random Field
CWR	Copy Weight with ReInit
DGR	Deep Generative Replay
DL	Description Logics
ESF	Ensemble of Shape Functions
EWC	Elastic Weight Consolidation
FWT	Forward Transfer
Hits@K	Hits at top K
ICP	Iterative Closest Point
KBC	Knowledge Base [Graph] Completion
KG	Knowledge Graph
L2R	L2 Regularization
LCA	Learning Curve Area
MLN	Markov Logic Networks
MN	Markov Network

MP3D	MatterPort3D
MRE	Multi-Relational Embedding
MRR	Mean Reciprocal Rank
MS	Model Size
NN	Neural Network
PCA	Principal Component Analysis
PCL	Point Cloud Library
PDAG	Partially Directed Acyclic Graph
PID	Proportional-Integral-Derivative
PNN	Progressive Neural Networks
REM	Remembering
ROS	Robot Operating System
SFE	Subgraph Feature Extraction
SI	Synaptic Intelligence
SRL	Statistical Relational Learning
SSS	Samples Storage Size
SVM	Support Vector Machine
VAE	Variational Auto-Encoder
XAI	Explainable AI
XAIP	Explainable AI Planning
XKGE	Explainable Knowledge Graph Embedding

SUMMARY

User demonstrations of robot tasks in everyday environments, such as households, can be brittle due in part to the dynamic, diverse, and complex properties of those environments. Humans can find solutions in ambiguous or unfamiliar situations by using a wealth of common-sense knowledge about their domains to make informed generalizations. For example, likely locations for food in a novel household. Prior work has shown that robots can benefit from reasoning about this type of semantic knowledge, which can be modeled as a knowledge graph of interrelated facts that define whether a relationship exists between two entities. Semantic reasoning about domain knowledge using knowledge graph representations has improved the robustness and usability of end user robots by enabling more fault tolerant task execution. Knowledge graph representations define the underlying representation of facts, how facts are organized, and implement semantic reasoning by defining the possible computations over facts (e.g. association, fact-prediction).

This thesis **examines the use of multi-relational embeddings as knowledge graph representations within the context of robust task execution and develops methods to explain the inferences of and sequentially train multi-relational embeddings**. To support this claim, this thesis contributes: (i) a survey of knowledge graph representations that model semantic domain knowledge in robotics, (ii) the development and evaluation of our knowledge graph representation based on multi-relational embeddings, (iii) the integration of our knowledge graph representation into a robot architecture to enable robust task execution, (iv) the development and evaluation of methods to sequentially update multi-relational embeddings, and (v) the development and evaluation of an inference reconciliation framework for multi-relational embeddings.

CHAPTER 1
INTRODUCTION

Everyday environments and users present a challenging problem domain for autonomous robot operation. Users want robots to perform a wide variety of everyday and specialized tasks, such as cleaning, organizing, and cooking [1]. However, it is challenging for users (i.e. non-experts) to demonstrate robot tasks such that the tasks are executable in all future environment instances a robot will encounter. User demonstrated tasks are brittle in part because everyday environments, such as households, are dynamic, diverse, and complex. Humans can find solutions in ambiguous or unfamiliar situations by using a wealth of facts about their domains to make informed generalizations. For example, when making a bowl of soup at a friend's house one might consider that bowls are likely found in a kitchen cabinet, a microwave can be used to cook, soup ingredients are likely to be found in a fridge, a spoon can be used to stir, and so on. This dissertation considers common-sense and situational facts pertaining to objects and their attributes that are structured as directed semantic relationships between two symbolic entities. Examples include actions that can be performed on and with objects, states objects can take, and locations objects can be found in. Prior work has shown that computational systems, including robots, can benefit from reasoning about these types of interrelated facts, which can be modeled as a Knowledge Graph (KG) [2].

KG representations are class of data structure that enable semantic reasoning by modeling KGs [3, 4, 5, 6, 7, 8]. KG representations define the underlying representation of facts, how facts are organized, and the possible computations over facts. Together these definitions result in a core set of reasoning abilities about facts a KG representation can provide. Some examples include deciding the satisfiability of a set of logical statements [9], conditioned fact prediction [10], estimating the prior probability of facts [11], approximating semantic similarity between fact constituents [12], and providing the confidence for a set of facts [13]. Robots can be endowed with semantic knowledge about a domain by interfacing with KG representations that computationally model the semantic knowledge about the domain [14, 4, 9]. Additionally, robots can perform different forms of semantic reasoning about domain knowledge by leveraging the core set of reasoning abilities a KG representation provides.

Semantic reasoning about domain knowledge using KG representations has improved the robustness and usability of end user robots. Prior works have enabled more fault tolerant autonomous execution in ambiguous or unfamiliar situations by leveraging semantic reasoning abilities provided by KG representations. In [15, 4], a Bayesian logic network is used to find objects in the most likely alternate locations during tasks. In [16] a directed graph is used to select alternate tools for actions executed during tasks. In [13], a Markov logic network is used to interpolate ambiguous end user commands into executable robot plans. In [10], a deep transformer-based neural network is used to infer conditional object properties during tasks, like object material given object class and color. In each referenced work, the robustness of the robot's autonomy is improved by informing the robot's decision-making of domain knowledge using a KG representation.

Several broad qualities of KG representations are worth consideration to support the non-expert use of robot autonomy in everyday environments. First, to accurately inform a robot's decision making, KG representations should sufficiently account for the *uncertainty of semantic knowledge*. Second, KG representations should be *adaptable to incorporate new domain knowledge* robots obtain from their dynamic environments. Third, everyday environments are rich with domain knowledge and require KG representations that can *scale reasoning* to KGs about those domains. Fourth, KG representations should *enable interpretable and explainable robot decision making* to promote trust between robots and their users. These often opposing objectives make designing a KG representation to support autonomous robots in everyday environments challenging. Tradeoffs exist in the design of KG representations that support autonomous robots due to the opposing nature of the above qualities. For example, accurate probabilistic modeling of the semantic knowledge uncertainties is at odds with the desire to scale reasoning to large KGs about realistic problem domains. As a result, no KG representation to date excels simultaneously in all performance characteristics. Instead, a wide range of KG representations [14, 4, 5, 17, 18, 10] have been developed and evaluated to support different robot application scenarios that each the have their own prioritization of the competing objectives.

This dissertation examines a novel KG representation to support autonomous task execution in everyday environments. KGs that contain semantic knowledge about real-world domains tend to have the properties of being *large*, *sparse*, and *incomplete*. For example, a KG representing semantic knowledge about households, while large, only contains a subset of true facts, which are sparse with respect to the space of many potential facts. Our KG representation uses a Multi-Relational Embedding (MRE), which is a distributed representation that models a KG in vector space [19]. We posit that MREs are well suited to model semantic knowledge about real-world domains because MREs are designed for KGs that are large and sparse [2]. Additionally, MREs excel at learning the underlying structure of KGs to infer new facts beyond those present in a KG, which promotes generalization.

1.1 Dissertation Overview

Everyday environments and users present a challenging problem domain for autonomous robot operation. Semantic reasoning about domain knowledge using KG representations has improved the robustness and usability of end user robots by enabling more fault tolerant execution. This dissertation examines the use of MREs as a KG representation in the context of autonomous task execution. In doing so, we develop a robot architecture that is informed by a MRE during autonomous execution of household cleaning tasks provided as prototypical task plans. Additionally, we introduce methods to explain the inferences of and sequentially train multi-relational embeddings to provide natural language explanations of task plan generalizations and enable updates to represented domain knowledge, respectively. We evaluate how each of these contributions affect MRE performance and ultimately, autonomous task execution success rate.

1.2 Thesis Statement

This thesis **examines the use of multi-relational embeddings as knowledge graph representations within the context of robust task execution and develops methods to explain the inferences of and sequentially train multi-relational embeddings**.

1.3 Contributions

1. **Survey of KG Representations in Robotics**: (Chapter 3) We characterize prior KG representations that model semantic domain knowledge in robotics. We first provide a comprehensive survey of existing work in semantic reasoning for robotics categorizing and comparing prior KG representations. Each referenced work provides examples where semantic reasoning about domain knowledge can improve robot behavior. Second, we extract a set of broad performance characteristics desired of KG representations that support autonomous robots from the surveyed prior works. Third, we compare existing KG representations using the defined performance characteristics, which helps to situate the contributions of this dissertation.

2. **Multi-Relational Embeddings as KG Representations for Robots**: (Chapter 4) We develop and evaluate our KG representation based on MREs. Using the insights from the survey presented in Chapter 3, we develop a novel KG representation for semantic reasoning about domain knowledge on robot systems using MREs. We evaluate our representation, showing that it outperforms description logics and word embeddings in terms of inference performance. Additionally, our experiments show that the proposed KG representation predicts new KG facts beyond the provided KG

facts. Lastly, we show that Bayesian logic networks require an intractable amount of memory to represent the same KG used throughout our experiments.

3. **Robust Task Execution using KG Representations**: (Chapter 5) We integrate our KG representation into the planning level of a robot architecture to improve robust task execution. We define the problem of one-shot task execution wherein a robot must generalize a single demonstrated task plan to a perturbed environment. We integrate the KG representation from Chapter 4 within a robot architecture to perform one-shot task execution informed by our KG representation. Lastly we evaluate our KG representation both in simulation and on a robot, showing the statistically significant improvements in task execution success rate provided by our KG representation. We also identified important avenues for improvement of the KG representation, like making MREs more adaptable and making MRE inference explainable.

4. **Continual Knowledge Graph Embedding**: (Chapter 6) We develop methods to allow the sequential training of our KG representation, which enables robot execution behaviors that adapt to changes in domain knowledge. We develop multiple Continual Knowledge Graph Embedding (CKGE) methods to enable the sequential training of MREs. We provide several datasets for the CKGE problem using different strategies that sample benchmark and household robot KG datasets. We evaluate each CKGE method on multiple datasets with a variety of continual learning metrics to track each method's inference performance, memory usage, and learning speed. Additionally, we provide guidence on which CKGE methods are best suited to different robot applications depending on time or data constraints. Lastly, we analyze each CKGE methods' effects on robot behaviors for the problem of one-shot robust task execution across multiple execution environments.

5. **Explainable Knowledge Graph Embedding**: (Chapter 7) We develop and evaluate an inference reconciliation framework for MREs using Explainable Knowledge

Graph Embedding (XKGE). We use a pedagogical approach to explain the inferences of a learned, black-box KG representation, a MRE. Our student, a graph feature model, uses a decision tree classifier to locally approximate the predictions of the black-box model, and provides natural language explanations interpretable by non-experts. Our explanations can be provided to end users of the robot, enabling a dialogue to reconcile the user's qualms about the robot's knowledge inferences. Results from our algorithmic evaluation affirm our model design choices, and the results of our user studies with non-experts support the need for the proposed inference reconciliation framework. Critically, results from our simulated robot evaluation indicate that our explanations enable non-experts to correct erratic robot behaviors due to nonsensical beliefs within the MRE.

1.4 Outline of Dissertation Document

This thesis is organized as follows. Chapter 2 provides background and related work on KG representations within robotics, MREs, continual learning, and explainable AI. Chapter 3 discusses KG representations for robot systems used in prior work. Chapters 4 and 5 discuss the design and application of the proposed KG representation using MREs. Specifically, Chapter 4 details the MRE domain knowledge representation, and Chapter 5 provides the details of how the proposed KG representation is used to inform a robot's decision making when during robust task execution. Chapter 6 discusses several continual KG embedding approaches that make the KG representation presented in Chapters 4 and 5 more adaptable. Chapter 7 presents a contributed explainable KG embedding method to explain MRE inferences in natural language. We provide concluding remarks and discuss open questions in Chapter 8.

CHAPTER 2
BACKGROUND AND RELATED WORK

In this chapter, we organize related works and background as follows. We first discuss the definitions of semantic common-sense knowledge and KGs we use in section 2.1. In section 2.1 we also mention existing KG representations used in prior robotics works to situate our proposed KG representation. In section 2.2 we provide definitions used in MRE literature. We follow these definitions with recent state-of-the-art methods for the task of link prediction. We follow these definitions with background and related works from continual learning literature in section 2.4 and explainable AI literature in section 2.5, which we use as inspiration to improve the adaptability and explainability, respectively, of MRE. Lastly, in section 2.6 we provide background about the task execution stack that we integrated with our KG representation to inform a robot's decision making.

2.1 Knowledge Graphs

Prior works represent semantic facts for use in robots as symbolic triples $(\mathbf{h}, \mathbf{r}, \mathbf{t})$, which indicate that the directed relationship $\mathbf{r}$ holds between head $\mathbf{h}$ and tail $\mathbf{t}$ of that triple (e.g. (mug, HasState, empty), (knife, CanBe, pickUp)) [3, 4, 5, 6, 7, 8]. Together, these individual facts form a graph $\mathcal{G}$ representing a KG that contains semantic knowledge about a domain (e.g. indoor homes and offices [20], Wikipedia [21], English [22]). This dissertation considers common-sense and situational facts pertaining to objects and their attributes that are structured as directed semantic relationships between two symbolic entities. Examples include actions that can be performed on and with objects, states objects can take, and locations objects can be found in.

KG representations computationally model KGs, providing semantic reasoning abilities to robots. KG representations organize facts and define the possible mathematical computations over facts that provide the core set of reasoning abilities available to a robot (e.g. association, satisfiability, fact-prediction). A wide range of KG representations have been proposed to support different robot application scenarios, prioritizing different qualities in these representations [23]. For example, a KG representation used to support robots working with non-expert end users might require interpretable inferences [9, 24], while one used as a shared cloud-based knowledge repository for fleets of robots must prioritize scalability [5, 25]. These broad qualities of KG representations supporting robotics applications, such as explainability and scalability, are defined and discussed further in chapter 3.

Prior works have leveraged KG representations to endow robots with complex knowledge reasoning abilities that provide solutions in ambiguous or unfamiliar situations. In [14], description logics are used to generate an ontology about the robot, such as what actions it can perform, and the domain it operates in, such as what objects exist. In [15, 4], a Bayesian logic network is used to find objects in the most likely alternate locations during tasks. In [16] a directed graph is used to select alternate tools for actions executed during tasks. In [13], a Markov logic network is used to interpolate ambiguous end user commands into executable robot plans. In [10], a deep transformer-based neural network is used to infer conditional object properties during tasks, like object material given object class and color. In each of these works, the robustness of the robot's autonomy is improved by informing the robot's decision-making of domain knowledge using a KG representation. In chapter 3 we discuss other KG representations used for robotics applications, in addition to those mentioned, to highlight the trade-offs of each KG representation. This dissertation proposes a new KG representation for use in robotics applications based on MREs.

2.2 Multi-Relational Embeddings

We adopt MREs for our KG representation. KGs that model semantic knowledge about real-world domains tend to have the properties of being *large*, *sparse*, and *incomplete*. For example, a KG representing knowledge about households, while large, only contains a subset of true facts, which are sparse with respect to the space of many potential facts. We use MREs for our KG representation because MREs are designed for KGs that are large-scale and sparse [26]. Additionally, MREs excel at learning the underlying structure of graphs to infer new facts beyond known facts in a graph.

MREs are distributed representations that model a KG in vector space [27]. In the MRE literature, a KG is modeled as a graph $\mathcal{G}$ composed of individual facts or triples (h, r, t); h and t are the head and tail entities (respectively) for which the relation r holds, e.g., $($*cup, canBe, fill*$)$. MREs learn a continuous vector representation of $\mathcal{G}$ from a dataset of triples $\mathcal{D} = \big\{(h, r, t)_i, y_i |\, h_i, t_i \in \mathcal{E}, r_i \in \mathcal{R}, y_i \in \{0, 1\}\big\}$, with $i \in \{1...|\mathcal{D}|\}$. Here y_i denotes whether relation $r_i \in \mathcal{R}$ holds between $h_i, t_i \in \mathcal{E}$.

The learning objective of MRE is to find a set of embeddings Θ that minimizes the loss $\mathcal{L}_{\mathcal{D}}$ over $\mathcal{D}$, where $\Theta = \big\{\{\mathbf{v}_e |\, e \in \mathcal{E}\}, \{\mathbf{W}_r |\, r \in \mathcal{R}\}\big\}$. Each entity $e \in \mathcal{E}$ is encoded as a vector $\mathbf{v}_e \in \mathbb{R}^{d_{\mathcal{E}}}$, and each relation $r \in \mathcal{R}$ is encoded as a mapping between vectors $\mathbf{W}_r \in \mathbb{R}^{d_{\mathcal{R}}}$, where $d_{\mathcal{E}}$ and $d_{\mathcal{R}}$ are the dimensions of vectors and mappings respectively [27, 28]. The embeddings for $\mathcal{E}$ and $\mathcal{R}$ are typically learned using a scoring function $f(h, r, t)$ that assigns higher (lower) values to positive (negative) triples [28]. Loss $\mathcal{L}_{\mathcal{D}}$ can take many forms depending on the MRE representation used, e.g., Margin-Ranking Loss [29] or Negative Log-Likelihood Loss [30].

Many MRE methods exist [26, 27, 31], and we use different MRE methods throughout this dissertation because the current state of the art is rapidly advancing from the research efforts of the Knowledge Base [Graph] Completion community. The next section of this chapter discusses in more detail recent state of the art MRE methods. Due to the rapidly developing formulation of the state of the art MRE, the main contributions of this thesis are *MRE agnostic by design*. These contributions include applying MREs as KG representations to robotics in the context of task plan execution, developing methods for the sequential training of MREs under continual learning assumptions, and providing natural language explanations for inferences produced by these black-box models. We have considered three KG embedding representations to show the generality of our methods: TransE, Analogy, and TuckER. We provide more details for each of the MREs used in the paragraphs that follow. By making our KG representation framework *embedding agnostic*, practitioners can use the latest state of the art MRE model.

TransE represents relationships as translations between entities, i.e., $\mathbf{v}_h + \mathbf{W}_r = \mathbf{v}_t$ [29]. It uses the scoring and margin ranking loss functions in Equation 2.1 and Equation 2.2, where $[x]_+ = max(0, x)$, γ is the margin, and (h', r, t') are corrupted triples in a corrupted KG $\mathcal{G}'$. Embeddings are subject to normalization constraints (i.e. $||\mathbf{v}_e||_2 \leq 1 \, \forall \, e \in \mathcal{E}$ and $||\mathbf{W}_r||_2 \leq 1 \, \forall \, r \in \mathcal{R}$) to prevent trivial minimization of $\mathcal{L}$ by increasing entity embedding norms during training.

$$f(h, r, t) = ||\mathbf{v}_h + \mathbf{W}_r - \mathbf{v}_t||_1 \tag{2.1}$$

$$\mathcal{L} = \sum_{\substack{(h,r,t) \in \mathcal{G}, \\ (h',r,t') \in \mathcal{G}'}} [f(h, r, t) + \gamma - f(h', r, t')]_+ \tag{2.2}$$

Analogy represents relationships as (bi)linear mappings between entities, i.e., $\mathbf{v}_h^\top \mathbf{W}_r = \mathbf{v}_t^\top$ [30]. It uses the scoring and negative log loss functions in Equation 2.3 and Equation 2.4 where σ is a sigmoid function, y is a label indicating whether the triple is corrupted, and $\mathcal{G}'$ is the corrupted KG. Additionally, the linear mappings (i.e. relations) are constrained to form a commuting family of normal mappings, i.e., $\mathbf{W}_r \mathbf{W}_r^\top = \mathbf{W}_r^\top \mathbf{W}_r \, \forall \, r \in \mathcal{R}$ and $\mathbf{W}_r \mathbf{W}_{r'} = \mathbf{W}_{r'} \mathbf{W}_r \, \forall \, r, r' \in \mathcal{R}$, to promote analogical structure within the embedding space.

$$f(h, r, t) = \langle \mathbf{v}_h^\top \mathbf{W}_r, \mathbf{v}_t \rangle \tag{2.3}$$

$$\mathcal{L} = \sum_{(h,r,t,y) \in \mathcal{G}, \mathcal{G}'} -\log \sigma(y \cdot f(h, r, t)) \tag{2.4}$$

TuckER represents relationships using the Tucker decomposition, i.e., $\mathcal{W} \times_1 \mathbf{v}_h \times_2 \mathbf{W}_r \times_3 \mathbf{v}_t$ where $\mathcal{W}$ is the core tensor and $\times_n$ is a tensor product along the nth dimension

[32]. It uses the *1-N* scoring and Bernoulli negative log loss functions in Equation 2.5 and Equation 2.6 where $p(h, r, t_i)$ is the predicted probability of tail entity $i \in \{1, ..., |\mathcal{E}|\}$ and y_i is a label indicating whether the relation r holds between h and t_i. Note that $p(h, r, t_i)$ is the predicted probability, not scoring function. The predicted probability p is a function of scoring function f that includes batch norms and dropout before putting output activations through a sigmoid function.

$$f(h, r, t) = \mathcal{W} \times_1 \mathbf{v}_h \times_2 \mathbf{W}_r \times_3 \mathbf{v}_t \tag{2.5}$$

$$\mathcal{L} = -\frac{1}{|\mathcal{E}|} \sum_{i=1}^{|\mathcal{E}|} \Big(y_i \cdot \log\big(p(h, r, t_i)\big) + (1 - y_i) \cdot \log\big(1 - p(h, r, t_i)\big) \Big) \tag{2.6}$$

In practice, KGs are often incomplete; e.g., missing relations between objects and object attributes. Therefore, a common evaluation task is to predict complete triples from incomplete ones by reasoning about a set of seed or training triples, i.e., predict h given (r, t) or t given (h, r) from a split of test triples given a split of training triples. To perform triple prediction, each test triple (h, r, t) is first corrupted by replacing the head (or tail) entity with every other possible entity in the current session $\mathcal{E}^n$. Then, to avoid underestimating the tested method's performance, all corrupted test triples that still represent a valid relationship between the corresponding entities are removed, known as the "filtered" setting in the literature [29]. Last, scores are computed for each test triple and its (remaining) corrupted triples using the scoring function $f(h, r, t)$, then ranked in descending order.

Link prediction in MRE is implemented by completing a transformation in the embedding space using the scoring function. For example, to infer tails $\{t_j | \, t_j \in \mathcal{E} \, \forall j\}$ that might complete $(h, r, \_)$, the scores $f(h, r, t_j)$ of all j triples are computed, and triples with scores meeting some classification threshold are classified true. Each score $f(h, r, t_j)$ is the resultant of a sequence of high-dimensional geometric transformations between the head entity vector $\{\mathbf{v}_h | \, h \in \mathcal{E}\}$, relation mapping $\{\mathbf{W}_r | \, r \in \mathcal{R}\}$, and tail entity vectors $\{\mathbf{v}_{t_j} | \, t \in \mathcal{E}\}$.

2.3 State of the art in Link Prediction

Since the seminal work of [33], many multi-relational embedding methods have been proposed, some of which are presented in prior graph embedding surveys [26, 27, 31]. The main research thrusts for link prediction using MREs has been on designing the representation of entities $\mathcal{E}$, relations $\mathcal{R}$, and scoring. A few recent examples include considering different spaces to represent entities and relations [34, 35], developing more complex scoring functions [32, 36], and incorporating regularization to promote generalization [37].

In addition to MRE, recent works have introduced architectures for link prediction that instead leverage large language models [38, 39, 40]. Applying large language models to the task of link prediction was introduced in COMET to predict missing commonsense knowledge [38]. COMET treated triples as sequences and leveraged the widely applicable transformer backbone [41]. The contributions of COMET allow link prediction to benefit from pretrained architectures of large language models like GPT [42] or BERT [43]. Recent examples after COMET [38] include KG-BERT [39] and LP-BERT [40]. KG-BERT applied a similar methodology as COMET but used a pretrained BERT language model instead of GPT. LP-BERT built on the contributions of COMET and KG-BERT by introducing pretraining tasks to benefit link prediction.

Which of these two approaches will remain as the state of the art representation to use for link prediction is still an open question as results are mixed depending on the benchmark dataset and metric used. For example, if we consider Mean Reciprocal Rank (MRR), which tracks the mean reciprocal rank of the correct response to a query, the results of LP-BERT [40] suggest that the RESCAL-DURA MRE [37] would be considered the state of the art across both benchmark datasets. This result is significant as RESCAL-DURA uses a simpler MRE formulation but includes regularization to promote generalization. The state of the art performance that can be achieved with a relatively outdated MRE [44] by incorporating improve regularization suggests that the higher parameter MREs and large language models may suffer from overfitting on the common benchmark datasets. When considering the broader set of performance metrics besides MRR, both MRE methods and large language models achieve the best reported performance depending on the metric and benchmark dataset. More recently, another direction of research has considered ensemble methods to combine the complementary nature of these two representations [45] in an ad-hoc fashion. The initial results of a MRE and large language model ensemble suggest this merge is a promising direction for future link prediction research [45].

2.4 Continual Learning

Continual learning has evolved as a subarea of life-long machine learning. It focuses on neural networks and seeks to learn new domains, classes, or tasks over time without forgetting previously learned knowledge [46]. Continual learning within computational models is challenging due to a problem known as catastrophic forgetting. Catastrophic forgetting [46] occurs when a neural representation that was optimized for a prior dataset is trained with a new dataset. The neural network's weights are tuned to the new dataset, resulting in a potential loss in performance for classes and tasks not included in the new dataset. Therefore, methods of continual learning seek to sequentially train computational models with samples becoming progressivley available over time while mitigating catastrophic forgetting.

We focus on an *Incremental Class Learning* continual learning scenario because it matches the assumptions of robots systems representing semantic common-sense knowledge. Different categories of continual learning scenarios exist in the literature depending on whether there are shifts in the input or output distributions, and whether the inputs and outputs share the same representation space, namely, incremental domain, incremental class, and incremental task learning [47]. In incremental domain learning the marginal probability distribution of inputs across learning sessions changes. In incremental class learning, in addition to the changes in the marginal probability distributions of inputs, in each learning session only an exclusive subset of classes is present. In incremental task learning, in addition to the two types of previously mentioned changes, the output spaces are disjoint between tasks, hence the output dimensions and their semantic meanings differ. Our work models a scenario where a robot is observing disjoint subsets of a complete KG. We consider incremental class learning because it best matches the assumptions of robot systems representing semantic knowledge with the distribution of input data and target labels changing across learning sessions as the robot incrementally obtains more domain knowledge.

Several classes of continual learning methods have been developed for object recognition, but methods for KG embedding remain unexplored. Regularization methods, such as [48, 49], use different regularization terms to control the changes to weights that were optimized in prior learning sessions. Architecture modification methods, such as [50, 51], add weights for new classes or tasks in future learning sessions by extend various dimensions of neural network architectures and bootstrap the training of new weights using previously learned weights. Generative replay methods, such as [52, 53], learn a generative model each learning session that is trained on examples from the current learning session and examples generated by the generative model from the previous learning session if one exists. Therefore, an ideal generative model supplies all training examples from previous learning sessions so that a discriminative model can train on examples from all learning sessions using batch learning. However, continual learning methods remain largely unexplored for KG embedding.

This dissertation develops and evaluates a suite of continual learning techniques for MREs. Of the range of classes of continual learning methods, only L2-regularization has been applied to KG embeddings [54]; more sophisticated methods that have shown promise in other domains, e.g., generative replay, remain unexplored. We explore and adapt five representative methods for KG embeddings that include all of the previously mentioned classes of methods [49, 50, 51, 52, 55]. Additionally, the implications of any related assumptions for robotics is not well documented because existing evaluations focus on the final inference performance and define different task specific measures [56]. Important measures for robotics, such as learning efficiency and model complexity, are not well documented for representative techniques [57], making it difficult to evaluate the suitability of these methods for modeling semantic knowledge in robotics. The methods and evaluations we develop are designed to fill these gaps, making our proposed KG representation for use on robot systems more adaptable to changes in domain knowledge.

2.5 Explainable AI Planning (XAIP)

Explainable AI Planning (XAIP) is a focus area of Explainable AI (XAI), with the goal of explaining an AI's reasoning to humans in complex decision-making procedures to foster trust, long-term interaction, and collaboration [58]. XAIP methods are designed with a particular audience in mind to receive the explanations, end users that are non-experts interacting with the system and algorithm designers that develop the system. Our research efforts are focused on non-expert end users. Inference reconciliation through dialogue with the AI is one method of explaining an AI's reasoning to end users, motivated by the notion that users have less computational power than sequential decision making systems (e.g., planners). In inference reconciliation through dialogue, user questions about the AI's plan are answered using explanations about the agent's underlying reasoning [58]. These interpretable explanations of the AI's underlying reasoning supporting the inference serve to reconcile the user's qualms about the AI's inference.

There are a growing variety of questions user's might ask about an AI's planning and representations affecting an AI's sequential decision making that need to be mapped into explanations as question responses. Questions from prior work include ”Why is action a in plan π?”, ”Why not this other plan π‘?”, ”Why is this policy (action) optimal?”, ”Why is action a taken in world state s?”, ”Why is the sequential decision making problem Π not solvable?”, ”Why did execution of plan π fail?”. Representations in prior work include plans, policies, rationales, and scene-graphs. In [59, 60, 61], causal link chains formed by action pre- and post-conditions within plans are used to answer "Why is action a in plan π?". In [62, 63], unmet properties of alternative plans (e.g., constraints) are highlighted to answer "Why not this other plan π'?". In [64, 65], the frequencies with which the current action lead to high-value future states or actions are used to answer "Why is this policy (action) optimal?". In [66], a mapping between user ascribed rationales and world states are used to answer "Why is action a taken in world state s?". In [67], a transformation (excuse) that makes the sequential decision making problem Π solvable is used to answer "Why is the sequential decision making problem Π not solvable?". In [68, 69], relevant scene-graph semantic relationships causing plan failure are verbalized to answer "Why did execution of plan π fail?".

This dissertation contributes a method to explain how knowledge inferences over a KG modeled using MRE affect an AI's sequential decision making. To the best of our knowledge, no prior work in XAIP has leveraged KGs as part of the sequential decision making representations that need to be mapped into explanations as question responses. A variety of user questions about an AI's believed facts and how those beliefs affect the AI's sequential decision making can be answered using a KG. We focus on questions of the form "Why is knowledge inference i supporting action a true?". The inference reconciliation framework based on KGs we contribute improves the explainability of our proposed KG representation by providing natural language explanations for knowledge inferences made using MREs. Our approach is inspired by prior works on explainable knowledge base completion.

2.5.1 Explainable Knowledge Base Completion

Knowledge Base [Graph] Completion (KBC) seeks to infer missing facts from a KG using existing facts [26]. In [26] two branches of KBC techniques that have received much attention are surveyed: latent and graph feature models. Latent feature models infer missing facts based on latent features of graph nodes (i.e., MREs) and graph feature models infer missing facts based on features extracted from observed graph edges (e.g., paths). Latent feature models tend to outperform graph feature models [70]. Inferences from latent feature models are not interpretable because all embedding values are learned relative to one another and, therefore, dimensions of latent features have no inherent meaning [70]. However, there exist applications where it is desirable to have accurate inferences that are interpretable (e.g., product recommendation).

Prior works have focused on improving the interpretability of embeddings for latent feature models and explainability of inferences. In [71, 72] embedding interpretability for expert users is provided through attention or importance weights over node features with respect to relation features, respectively. These weights indicate to expert users which interacting dimensions most contributed to an inference. Such explanations are not interpretable to non-experts as dimensions within MREs carry no semantic meaning. In [73, 74], the reasoning behind or reliability of inferences was explained to non-experts in terms of the observed short alternative paths or "crossover interactions" between inferred and given facts, respectively. These methods do not provide explanations for inferences derived from negative correlations because they rely on heuristics, such as support, that only consider positive correlations. In [70], the reasoning supporting inferences was explained by learning an explainable model that provides most highly correlated alternative paths. The explainable model proposed was linear logistic regression, which may not accurately capture the joint contributions of sets of alternative paths. In [75], fully grounded explanations about inferences were provided by using expert labels to learn in a semi-supervised manner which "template" explanation (similar to alternative paths) is best suited to explain an inference. The template selection module learns to select templates mimicking the expert's labels but such labels could diverge from the underlying correlations a MRE, and hence the robot, used to make an inference.

We contribute a novel graph feature model to support our inference reconciliation framework. We compare our graph feature model to each of the mentioned prior works. The developed inference reconciliation framework is built around our graph feature model to provide non-expert end users with natural explanations of inferences made using MREs. Therefore, our methods make the proposed KG representation more explainable to non-expert end users.

2.6 Robot Task Execution Stack

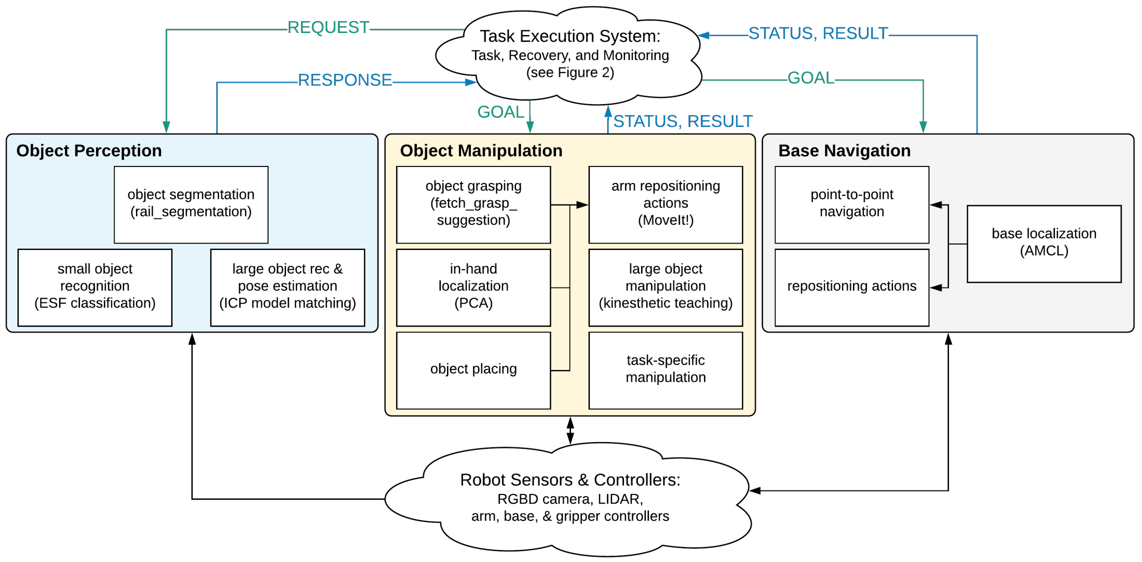
We integrated our KG representation with a mobile manipulator to inform the robot's decision making using the KG representation. We used a Fetch [76] mobile manipulator robot throughout our experiments. In the text that follows we provide some details to the system that sequences primitive robot actions (skills) and monitors their execution, namely the robot's task execution stack [77]. Our planning level module that integrates the KG representation with the task execution stack (see section 5.1) allows the robot to select which sequence of primitive actions a robot should execute to accomplish a task, informed by the semantic knowledge within a KG representation. Below we provide the details first of the modules that implement primitive actions executable by the mobile manipulator, then we briefly overview the task execution stack that sequences and monitors the execution of primitive actions.

The task execution stack consists of a set of independent mobile manipulation modules, implemented using the Robot Operating System (ROS) [78], shown in Figure 2.1. Object perception modules are implemented as ROS service servers, and object manipulation and base navigation modules are implemented as `actionlib`[1] servers. Each independent module can be called by the task executor, and provides feedback to the task executor and task monitor[2]. The modules consist of the following capabilities:

Object Perception. Our perception modules implement a perception pipeline for image

[1]http://wiki.ros.org/actionlib

[2]Each module must necessarily provide feedback on its own faults so that the executive level can make relevant recovery decisions.



REQUEST
RESPONSE
Task Execution System:
Task, Recovery, and Monitoring
(see Figure 2)
STATUS, RESULT
GOAL
GOAL
STATUS, RESULT
Object Perception
object segmentation (rail_segmentation)
small object recognition (ESF classification)
large object rec & pose estimation (ICP model matching)
Object Manipulation
object grasping (fetch_grasp_ suggestion)
arm repositioning actions (MoveIt!)
in-hand localization (PCA)
large object manipulation (kinesthetic teaching)
object placing
task-specific manipulation
Base Navigation
point-to-point navigation
repositioning actions
base localization (AMCL)
Robot Sensors & Controllers:
RGBD camera, LIDAR,
arm, base, & gripper controllers

Figure 2.1: Mobile manipulation system overview. Arrows denote ROS information flow, through publishers, subscribers, services, and actionlib.

and depth sensor data, using the Point Cloud Library (PCL) [79]. The object segmentation module uses the `rail_segmentation`[3] package to identify point cloud clusters-of-interest through table surface detection and Euclidean distance clustering. We divide our object recognition approaches between large and small objects. For large objects, we perform model matching using the `rail_mesh_icp`[4] package that uses an Iterative Closest Point (ICP) PCL pipeline, which also provides object pose detection. For small object recognition, we train an Support Vector Machine (SVM) classifier over Ensemble of Shape Functions (ESF) descriptors [80]. We do not need to perform pose estimation for small objects due to our object grasping approach, described below.

Object Manipulation. Most of our manipulation modules use MoveIt! to perform arm planning to either joint goals or end-effector pose goals using the Open Motion Planning Library's `RRTConnect` motion planner [81]. This includes both general arm repositioning actions, which the task executor can call directly (e.g. to move the arm out of the way of the camera), and execution actions, called by other object manipulation mod-

[3]http://wiki.ros.org/rail_segmentation
[4]http://wiki.ros.org/rail_mesh_icp

ules. Object grasping calculates antipodal grasps over an object point cloud using the `agile_grasp` package [82], which are then ordered and executed using pairwise ranking through `fetch_grasp_suggestion` [83]. As objects can shift during the grasping process, we perform post-grasp pose detection using the in-hand localization module, which identifies the object point cloud by performing background subtraction on the robot's gripper, and calculates the object's pose based on its principal axes determined by Principal Component Analysis (PCA). Given a known object pose and a desired place location, object placing calculates and executes a pose goal for placing an object that ensures the gripper fingers and palm are out of the way of the object's fall trajectory.

We also include some manipulation modules that do not use MoveIt!, due to the limitations of sampling-based motion planning. For large object manipulation, such as lifting and placing kits of objects, we include a kinesthetic teaching module [84]. The kinesthetic teaching module allows system designers to record and play back arm trajectories, either in full or as a set of waypoints. Additionally, we include task-specific manipulation actions to implement specific manipulation skills such as raising and lowering objects, using a Cartesian end-effector controller[5], and peg-in-hole insertion, using a controller with end-effector pose and joint effort as feedback.

Base Navigation. LiDAR-based localization uses adaptive Monte Carlo localization provided by ROS's `nav_stack`[6] to localize the base with respect to a pre-collected 2D occupancy grid of the environment. Navigation is primarily done using point-to-point navigation between waypoints on the map, executed using a Proportional-Integral-Derivative (PID) controller[7]. We also include local repositioning actions, which implement short movement primitives such as backing up from a table. The repositioning actions are implemented using a PID controller with gains tuned for shorter, more precise base goals.

With each module implemented, the navigation, perception, and manipulation actions

[5]Available at https://github.com/GT-RAIL/fetch_simple_linear_controller

[6]http://wiki.ros.org/navigation

[7]In complex environments, `nav_stack`'s global and local planners can be used instead.

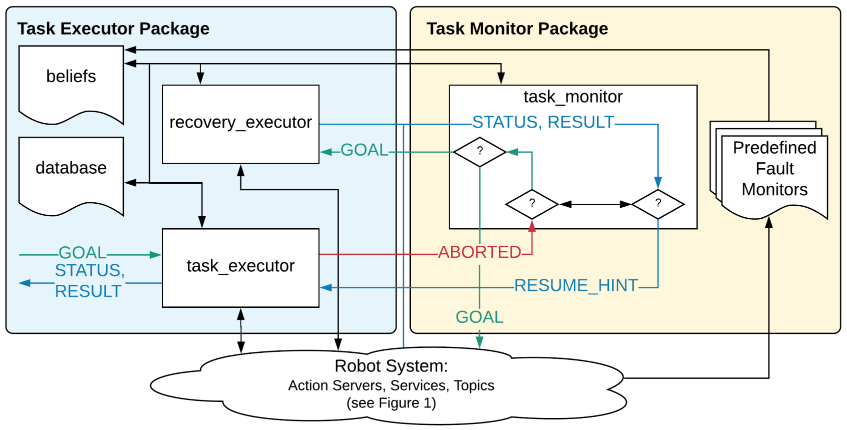


Figure 2.2: Overview of the two packages in our the executive level. Arrows denote ROS information flow, through publishers, subscribers, services, and actionlib.

can be sequenced in a robust manner to complete mobile manipulation tasks by the task execution system described in the next section.

In addition to the modules that implement primitive actions, we briefly describe the modules that sequence and monitor the execution of primitive actions shown in Figure 2.2: the *task executor* and the *task monitor*[8]. The task executor enables a developer to access all implemented primitive actions using a consistent API and specify sequences of actions to execute via manually defined YAML files or programmatically using ROS services. Task relevant information used when executing primitive actions is contained in a database within the task executor for use when calling primitive actions (e.g. locations, gripper poses, arm poses). The task monitor checks the return values of primitive actions to trigger errors when return values diverge from what is expected. In addition to the previously mentioned, the task executor and monitor together implement a recovery system to decide how to resume a sequence of primitive actions when the robot encounters a primitive action failure (e.g. arm motion planning error). These recoveries rely on a variables or beliefs describing task relevant environment states. The details of the recovery modules remain out of scope for the purposes of this dissertation but more details are in [77].

[8]Stand-alone packages at https://github.com/GT-RAIL/assistance_arbitration

CHAPTER 3
SURVEY OF KG REPRESENTATIONS IN ROBOTICS

In this chapter we survey prior KG representations used for robot applications to situate our proposed KG representation. The text in this chapter is selected from a broader survey covering computational frameworks used for semantic reasoning in robot systems [23]. Here, we focus on computational frameworks that are KG representations [23]. From the many works surveyed, broad qualities of KG representations supporting robots become prevalent. We define and discuss these qualities, which together with the referenced works, help to situate our proposed KG representation within the space. Our proposed KG representation is then presented in chapter 4.

3.1 Introduction

KG representations are the organizational structures that enable robots to reason about semantic knowledge. Each KG representation offers different mathematical structures, assumptions, and types of inference (i.e. reasoning). Each KG representation differs in the extent to which it can *model uncertainty*, be *adaptable*, be *explainable*, and *scale* due to each representation's formulation. These broad qualities of KG representations, defined below, are prioritized according to a given robotics application. For example, the scalability requirements of a KG representation for a *single* robot reasoning about affordances [85] are different from a framework designed as a shared cloud-based knowledge repository for *many* robots across different environments [5]. A wide range of KG representations have been applied to reason about semantic knowledge on robot systems and no single approach is applicable across all scenarios.

1. *Modeling uncertainty* - extent to which likelihoods, rankings, confidences, or assign-

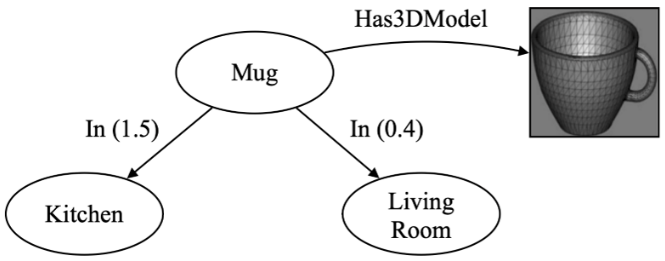


Figure 3.1: An example directed graph that includes multimodal data, different types of relations, and local confidence values defined for edges (shown as values in parentheses). The confidence scale is not bounded.

ments of concepts modeled by a KG representation correspond with their ground truth values;

2. *Adaptability* - extent to which the KG represented can be adapted without corruption of existing KG;

3. *Explainability* - extent to which the KG representation and reasoning is interpretable;

4. *Scalability* - extent to which KG representation and reasoning can be scaled in terms of number of edges and nodes in the KG before being computationally intractable due to time or memory complexity;

Below we present a representative list of KG representations that have been used in prior robotics works. Briefly, we describe the different data and mathematical structures upon which these frameworks are built, how their assumptions lead to various tradeoffs with respect to the mentioned above qualities, and use cases of each KG representation in the context of semantic reasoning. More details about each KG representation and their use in robot systems can be found in the works referenced. We conclude with some discussion comparing KG representations in terms of the above qualities and situate our proposed representation.

3.2 Directed Graphs

A directed graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is a set of vertices **V** connected by directed edges **E**, represented by triples $\{(v_i, e_j, v_k) | v_i, v_k \in \mathbf{V} \wedge e_j \in \mathbf{E}\}$. Vertices **V** store one or multiple types of entity information (e.g., text, real-numbers, images, sounds, trajectories, and algorithm parameters) and each edge $e \in \mathbf{E}$ has a predefined edge type that represents the relation between the connected entities. For example, an edge with relation *HasAppearance* can connect the abstract label *Cup* to vertices containing images of cups. Some prior work has also included confidence values that can be associated with vertices **V** and edges **E** in order to represent the certainty of the encoded knowledge [5]. Figure 3.1 shows an example directed graph.

Tradeoffs in the Context of Robotics: A directed graph is highly adaptable and expandable. New robot observations or facts can be easily added to a directed graph in the form of new vertices or edges. Additionally, confidence values associated with vertices or edges can be updated to reflect new observations. Vertices can also be merged or split, as in [5], when new knowledge is acquired. However, a significant limitation of the graph representation is that it does not support rigorous probabilistic inferences. Furthermore, the belief of an edge or node is not well established, making it difficult to assess the relative certainty of various types of information contained with the directed graph. One approach uses the Katz centrality to assign an ‘importance’ score to nodes corresponding to particular objects or motions [86], while another incorporates beliefs from disparate knowledge sources or algorithms [5].

Uses and Applications: Reasoning over directed graphs can be performed at the node, local subgraph, or global graph levels. At the node level, node similarity can be computed by comparing the locations of nodes in the directed graph. For example, Wu-Palmer similarity [87] is used on WordNet [22] data to generalize manipulation sequences [88] and organizational preferences [89] between similar objects. At the subgraph level, informa-

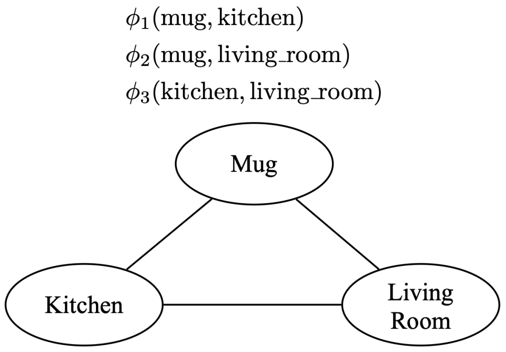


Figure 3.2: An example Markov network with undirected edges between random variables. The joint distribution can be factorized into clique potentials, denoted by ϕ's.

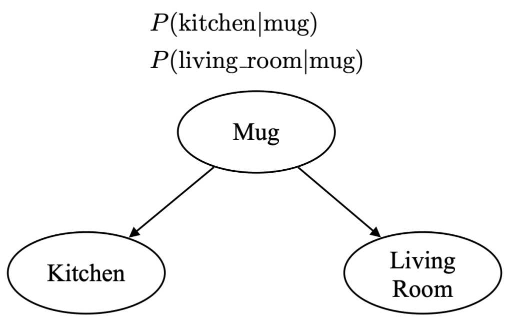
tion retrieval on large-scale directed graphs is performed by matching a query template with the graph [5, 86, 90]. Queries typically take the form (u, e, v), in which the variables u and v are nodes in the directed graph and the variable e is a directed edge from u to v. For example, the query $(u,$ *HasAffordance*$, scoop)$ can be used to retrieve a list of objects that provide the scooping affordance, and the query $(spoon, e, kitchen)$ can be used to identify the relationship between a spoon and a kitchen. More complex reasoning can be achieved by chaining multiple queries together, such as the above examples that can be used to identify that going to the kitchen may allow the robot to find an object for scooping [5]. At the global level, graph matching assesses similarity between different models. For example, graph matching over topological models of human spaces and objects provides a solution for place recognition and place classification [91], and graph matching over object models of constituent parts enables object recognition [92]. A different approach is used in [93], in which a score over an entire object graph is computed based on object properties and neighboring objects. The importance of different features is learned from demonstrations in order to encode trajectory preference.

3.3 Markov Network

A Markov Network (MN), or Markov Random Field, is a probabilistic graphical model represented by the pair $(\mathcal{H}, \mathcal{P})$. The joint probability distribution $\mathcal{P}$ factorizes over the undirected graph $\mathcal{H}$, whose nodes represent a set of propositional random variables and edges represent the correlations between random variables, as illustrated in Figure 3.2. A commonly used type of MN is a Conditional Random Field (CRF), in which random variables are divided into a set of target variables Y and a set of observed variables X. Rather than encoding the joint distribution $P(Y, X)$, a CRF represents the conditional distribution $P(Y|X)$.

Tradeoffs in the Context of Robotics: As a probabilistic model, a MN provides a KG representation for representing a complete probability distribution - the probability of every possible event as defined by the values of all the random variables. Additionally, the independence assertions encoded in the graphical structure allow a distribution to be compactly represented as products of factors, or clique potentials. Since the factorization is over cliques, which are fully connected subsets of the random variables in the graph (e.g., pairs of variables), a Markov network is especially suitable for modeling symmetric or associative relations between variables. When relations between certain variables are hard to elicit due to overlapping information or implicit correlations, a CRF can be used to avoid representing a probabilistic model over these variables. However, the flexibility in defining MNs and CRFs results in a lack of clear semantics, which has several disadvantages. First, each clique contributing to the overall inference result does not help to reveal which random variables affect the result the most. Second, the use of cliques to define a joint distribution makes parameterizing the model by hand more difficult. The second limitation leads to the convention of learning clique potentials from training examples, which requires apriori data to converge to reasonable parameters [94, 95, 96].

Uses and Applications: MNs have been used to model spatial and contextual relations



$P(\text{kitchen}|\text{mug})$
$P(\text{living\_room}|\text{mug})$
Mug
Kitchen
Living Room

Figure 3.3: An example Bayesian network with directed edges between random variables. The joint distribution can be factorized into conditional probabilities, denoted by P's.

between objects. Jointly reasoning about these relations helps to improve the robustness of object classification algorithms over those that are based solely on visual features. For example, Relational Markov Networks, an extension of CRFs, have been used to represent the spatial relations between walls and doors in 2D laser scans [95]. Modeling the spatial relations allows the approach in [95] to infer labels for line segments that are not confidently classified from the 2D map features alone. Similarly in [96], CRFs are used to exploit contextual and spatial relations between objects in a scene to improve object classification.

3.4 Bayesian Networks

A Bayesian Network (BN) is a probabilistic graphical model represented by the pair $(\mathcal{G}, \mathcal{P})$, where the probability distribution $\mathcal{P}$ factorizes over a directed acyclic graph $\mathcal{G}$. The nodes in $\mathcal{G}$ represent propositional random variables, and edges represent informational or casual dependencies between the variables. A BN asserts that each variable is independent of its nondescendants in $\mathcal{G}$ given its parents (see Figure Figure 3.3). The collection of independence assertions allows a BN to compactly represent a joint probability distribution by factoring it into local, conditional distributions for each variable.

Tradeoffs in Context of Robotics: The main advantages of a Bayesian network as a KG representation are its precise probabilistic interpretation and its adaptability. The Markov assumption and directed-acyclic constraints that define the scope of the network, allow for simple interpretations of conditionally independent variables and causal relations between variables. The structure of BNs allows for direct inference over variables and learning of parameters/structures via efficient approximations, such as importance sampling or Gibbs sampling. Factorizing the full joint distribution over $\mathcal{G}$ via conditional probability tables (or distributions) also makes determining the influences of inference results more accessible than MNs. However, managing large-scale conditional probability tables leads to drawbacks in terms of scalability. While the richness of the resulting probabilistic representation is useful to robots reasoning about specific problems, inference and learning in BNs is often intractable for real-world problem sizes involving many random variables and edges in a densely connected network. Thus, due to limited scalability, BNs are rarely used in complex robot environments, but instead are commonly utilized as a foundation for more complex KG representations.

Uses and Applications: BNs have been used for semantic grasping by encoding relations between grasps, object features, and task constraints [97]. In [97], Gaussian Mixture Models are used to discretize continuous data for BNs in order to learn the network structures for factors such as object convexity and grasp location. In [98], BNs are extended to incorporate context and temporal relations into action selection using Dynamic Bayesian Networks, a representation that presents a compromise between state and space complexity.

3.5 Partially Directed Acyclic Graphs

A Partially Directed Acyclic Graph (PDAG), or Chain Graph, is a graphical model represented by the pair $(\mathcal{I}, \mathcal{P})$, where the probability distribution $\mathcal{P}$ factorizes over the hybrid graph $\mathcal{I}$, which consists of both directed and undirected edges that represent influences between the propositional random variables encoded in the nodes of $\mathcal{I}$ [94]. Figure Figure 3.4 shows an example PDAG.

Tradeoffs in Context of Robotics: PDAGs, which can be thought of as a combination of MNs and BNs, allow for modeling causal as well as associative relationships. However,

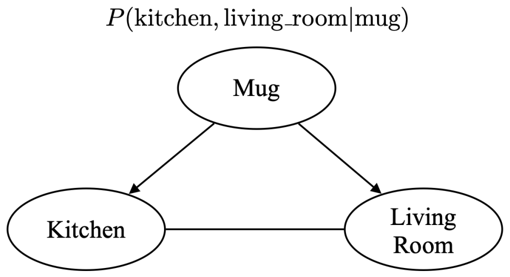


Figure 3.4: An example partially directed acyclic graph with both directed and undirected edges between random variables. The edge between the two nodes in the same chain component is undirected, while the edges between two nodes in different chain components are directed. The joint distribution can be factorized into conditional probabilities of chain components given their respective parents.

similar to MNs, the PDAG joint distributions are defined over chain components of cliques in the moralized graph of $\mathcal{I}$ instead of individual conditional probabilities, as in BNs. Factorization over cliques leads to confounding inference results for reasons similar to that of MNs, namely it is difficult to distinguish the variables that contribute to an inference result. Additionally, PDAGs lack clear semantics for model parameters, which makes model parameters difficult to elicit from experts. As a result, the convention, much like for MNs, is to estimate model parameters from training data [17].

Uses and Applications: PDAGs have been used to perform causal and associative reasoning of spatial common-sense knowledge in order to build spatial models of indoor environments. In [17, 24], each room instance is connected to one another by undirected edges according to a topological map. The potentials on undirected edges are used to describe typical connectivity between room categories. Within each room, the variable representing the room's category is linked via directed edges to the room shape, size, appearance, and objects in it, capturing the causal relations between these attributes and the room type. Similarly in [99], undirected edges are used to model connectivity; however in [99], they connect nodes representing waypoints in the map. For each waypoint, causal relations between viewing angles, expected objects in a view, and observations from a robot are modeled by directed edges.

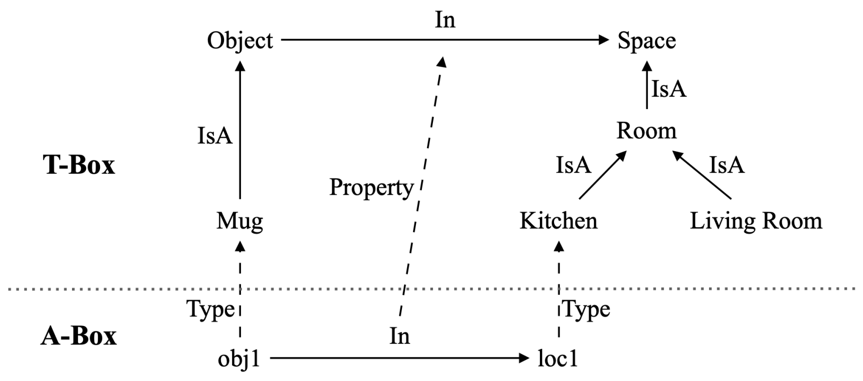


Figure 3.5: A graphical representation of an example description logic axioms. The T-Box contains axioms defining relations between class-level concepts. The A-Box contains a single fact, which is governed by the constraints defined in T-Box.

3.6 Logics

Logics are a family of programming languages used to describe the relations between problem domain objects, attributes, actions, and axioms through a series of true or false clauses [100]. Examples include Description Logics (DL) [101], Prolog [102], and Answer Set Programming (ASP) [103]. Figure Figure 3.5 shows an example of a description logic ontology. Description logics represent a domain knowledge using the pair $(\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$, the T-Box, contains terminological axioms describing relationships between concepts, and $\mathcal{A}$, the A-Box, contains assertional axioms capturing knowledge about named individuals, i.e., the concepts to which they belong and how they are related to each other [101]. Prolog encodes domain knowledge as a set of clauses that include relations between concepts and rules structured as "Rule Head true if Rule Body true" [102]. In addition to modeling relations between concepts, ASP allows for reasoning with incomplete knowledge by using default negation to denote the unknown value of concepts in addition to true or false values [100].

Tradeoffs in Context of Robotics: The predefined structure of clauses in logics enable transparent and interpretable reasoning. As rules in logics define a causal relationship between the head and body of a rule, all inferred relations must be implied by provided rules and relations. Additionally, by only modeling true or false clauses, logics-based semantic reasoning frameworks can scale to a greater number of instances, types, and relations than frameworks that model full joint probability distributions [15]. Although logics typically only contain true or false clauses, many extensions have been made to logics to reason about the uncertainty of clauses by associating probabilities with them [104], such as Problog [105], a probabilistic Prolog extension, and P-log [106], a probabilistic ASP extension.

Uses and Applications: Large-scale description logic ontologies have been used to encode contexts, spaces, objects, actions, and features, along with axioms that express inter- and intra-group relations [107, 108, 109, 110, 111, 112]. The KnowRob ontology is constructed by combining a manually designed ontology with the public OpenCyc ontology, therefore bootstrapping available declarative knowledge with general knowledge that could be leveraged during tasks [113]. The ORO ontology is created in a similar fashion by integrating with the OpenCyc ontology [9]. However, the ORO ontology focuses on human-robot interaction, therefore adding new concepts designed to facilitate interaction and modeling of human agents. Problog, the probabilistic extension of Prolog, was used in [114] to extend probabilistic programming techniques to deal with dynamic relational domains involving both discrete and continuous random variables. Each state of the environment is represented as a set of ground facts that define a possible world and a particle filter is developed to allow one to recursively estimate the state over time given some observations. ASP and P-log, a probabilistic extension of ASP, have been used in many robotics applications for planning in open and uncertain worlds [115]. ASP is used in [116] to enable mobile robots to collaboratively tidy up a house by modeling commonsense knowledge about households and using the ASP solver to guide motion planning by providing a discrete task plan that satisfies complex temporal goals. The work of [117] introduced richer representation in ASP with incomplete domain knowledge and the use of ASP-based inference to heuristically generate a multinomial prior for partially observable Markov decision process state estimation.

$\forall x \forall y \ \text{IsMug}(x) \land \text{IsKitchen}(y) \Rightarrow \text{AtLocation}(x, y)$	1.5
$\forall x \forall y \ \text{IsMug}(x) \land \text{IsLivingRoom}(y) \Rightarrow \text{AtLocation}(x, y)$	0.7
$\forall x \forall y \ \text{IsKitchen}(x) \land \text{IsLivingRoom}(y) \Rightarrow \text{Near}(x, y)$	0.4

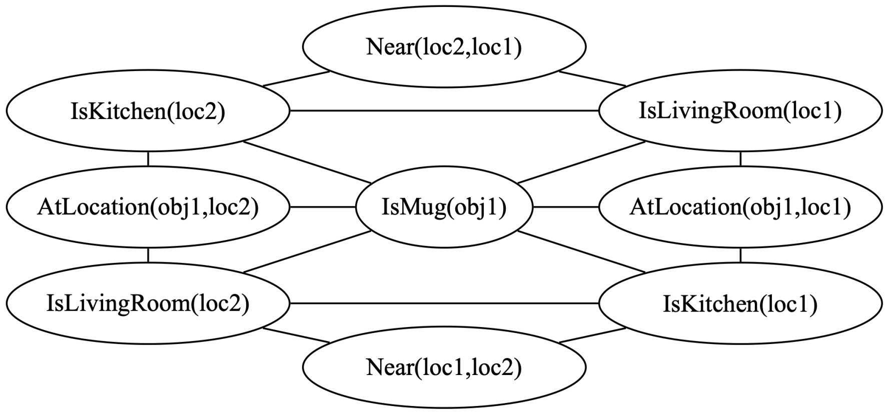


Figure 3.6: An example Markov logic network and its grounded Markov network. Each formula defining the Markov logic network has an associated weight that reflects how strong a constraint it is. The Markov network has 3 grounded atom: obj1, loc1, and loc2.

3.7 First-order Probabilistic Models

First-order Probabilistic Models are formalisms widely used in Statistical Relational Learning (SRL) that combine graphical models with first order relational representations [118], such as Markov Logic Networks (MLN) [119] and Bayesian Logic Networks (BLN) [120]. A first-order probabilistic model is typically represented as a collection of first-order logic formulas with confidence scores, as illustrated in Figure Figure 3.6. For example, a MLN is formally defined as a set L of pairs (F, w), where F is a first-order logic formula and w is a real number denoting the belief about F. More details about SRL can be found in [121] and [122].

Tradeoffs in Context of Robotics: The most obvious benefit that results from combining probabilistic reasoning with first-order logic is that relational inferences can be used to model uncertainty, becoming more flexible to noisy or contradictory evidence. Another advantage of First-order probabilistic models is that its world definitions are more compact because variables act as placeholders for entities, which allows them to make relational rules interchangeable among entities. In contrast, languages based on propositional logic or propositional probabilistic graphical models (e.g.,MNs, BNs, and PDAGs) assume each symbol represents a concrete fact or entity. However, while first-order probabilistic models allow the compact writing of rules, their representation rapidly expands when performing probabilistic inference or learning because first-order probabilistic models still need to be grounded to their constituent probabilistic graphical models for computations. For example, a MLN needs to be grounded to a MN, which contains every possible assignment to the variables in the MLN, as illustrated in Figure Figure 3.6. The scalability of inference and learning can be partially addressed by leveraging structures in the models or exploiting the context of an inference. For example, in [13] the context from input language related to the inference and the semantic description of the observed world state are used to reduce the large number of variables in a grounded knowledge state. Due to poor scalability, applications of first-order probabilistic models for large-scale semantic reasoning remain limited.

Uses and Applications: First-order probabilistic models have been used to construct multi-relational probabilistic knowledge bases. In [6], a MLN is used to store knowledge between object properties and affordances by using probabilistic relations such as *isA*, *hasAffordance*, and *hasVisualAttribute*. The MLN allows for a variety of queries, for example, predicting affordance based on properties extracted from object images and inferring typical features of objects with a specific affordance. Similarly in [123], a MLN encoding relations between object properties (e.g., shape, size, and logo) is used to fuse information from different perception routines for collective classification. In [85], a probabilistic programming language, ProbLog, is used to construct a multi-objects affordance model. The probabilistic logical rules can deal with uncertainty in perception and action outcomes. Through the use of placeholder variables in place of individual objects, the relational representation is able to generalize the affordance model learned from 2 objects to any arbitrary numbers of objects. In [4, 15], BLNs are used to generate situated probabilistic models of environments. The BLNs model relationships between objects, object locations, rooms, and object affordances. The BLN can be used to make different queries, like enabling the robot to provide likely locations of objects. In [124], Distributional Clauses, which is a first-order probabilistic model that supports modeling continuous probability distributions, enables occluded object search by encoding different spatial relations between objects such as co-occurrence and stacking in addition to affordance related relations. A dynamic version of Distributional Clauses is used for object tracking during human activities [125]. The physics laws and common sense knowledge (e.g., if an object is on top of another object, it cannot fall down) that are encoded in probabilistic and continuous first-order rules help robots robustly track objects even with occlusion.

3.8 Neural Networks

A Neural Network (NN) is a parameterized function approximator $\mathcal{F}$ that defines a mapping from input data $\mathcal{X}$ to output results $\mathcal{Y}$ (i.e. $\mathcal{Y} = \mathcal{F}(\mathcal{X}, \Theta)$, where Θ represents parameters of $\mathcal{F}$) [126]. The NNs used in semantic reasoning usually have two distinctive features. First, these NNs often make use of data containing semantic features, such as natural language descriptions of manipulation actions [127], or subgraphs from a KG [128]. Second, the objective of these NNs is to learn not only the mapping from input to output, but also the structure and semantic relations that underlie the data. Embedding methods [31], in particular, represent a family of NNs that focus on encoding the structure of data by projecting input data into vector spaces in which spatial relations reflect the semantic relations of the input data.

Tradeoffs in Context of Robotics: NNs are a flexible computational framework capable of learning highly complex relations from data that are often difficult to encode manually. NNs can also take data of almost any form (e.g., task label, natural language instruction, image, point cloud, grasp pose, and trajectory), thereby allowing multimodal data to be combined and reasoned about collectively in a principled way. However, the adaptability of NNs comes at a cost to semantic reasoning for robotic applications. First, training NNs

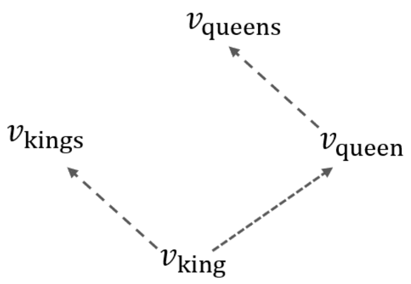


Figure 3.7: An example neural network word embedding where proximity of semantic meaning can be determined by vector space computations. Entities are vectors and relations can be embedded as vector offsets.

requires large amounts of data, which is often challenging to obtain, particularly in physical environments. Second, NNs are not as transparent as logic or probabilistic models, therefore reducing the explainability of the decision-making process. Additionally because the learned semantics within the NNs are challenging to extract, a NN trained in one domain may not transfer well to another (e.g., simulation vs real-world). While progress continues to be made on each of these fronts, these are considerable limitations for practical applications involving physical robots operating autonomously in real-world environments.

Uses and Applications: NNs have been used in a number of semantic reasoning frameworks to learn mappings from semantic features to desired robot behaviors. In [127, 129], NNs are used to learn a multi-modal embedding that fused trajectory, language instruction, and object point cloud data. The NNs are used transfer manipulation trajectories to previously unseen objects or provide a semantic label for a manipulation being performed on an object. In [130], contextual information, such as object affordances, materials, and intended manipulation tasks, represented in the form of text labels, are used as inputs to a NN to predict the compatibility of grasps with given tasks.

As noted above, embedding methods also have been widely used in semantic reasoning to capture proximity of meaning within the data. Word embeddings capturing similar word meanings [131] are used in [12] and [18] to perform multi-modal language grounding and learn common sense navigational knowledge, respectively. Graph NNs learn graph embeddings and have been applied to the problems of recognition of novel objects [132] and visual semantic navigation in novel environments [133].

3.9 Summary of KG Representations

A wide range of KG representations have been used for semantic reasoning in robotics, and no single representation is applicable across all scenarios. We can compare KG representations by the previously defined broad qualities that result from each representation's underlying assumptions and formulation. These qualities include modeling uncertainty, adaptability, explainability, and scalability. Similarly, one can prioritize these qualities for an application by considering the reasoning problem domain, available data, end user of the robot, and other considerations. Therefore, the selection or design of a KG representation for an application can be informed by matching the qualities between the representation and application. Trade-offs exist between each of these qualities due to their opposing nature, and existing KG representations have succeeded in excelling at subsets of these qualities. Below we summarize how the KG representations in the previous sections of this chapter excel in subsets of these qualities.

Modeling uncertainty: Probabilistic models, including MNs, BNs, PDAGs, and first-order probabilistic models, are inherently effective at modeling uncertainty. When using other frameworks in situations with nondeterministic information, probabilistic variants of these frameworks, such as Bayesian Neural Networks [134] and Probabilistic Description Logics [135], can be selected.

Adaptability: Incorporating new information into probabilistic models is hard as it often entails learning new parameters or structures. In contrast, new knowledge can be more easily added as new assertions in logics-based frameworks and as new nodes or edges in directed graphs. Since NNs are typically capable of generalizing learned models to new data, new information can also be reasoned without retraining. However, in cases where retraining is necessary, NNs can be challenging to update without corrupting learned semantic knowledge due to the previously mentioned problem of catastrophic forgetting (see section 2.4).

Explainability: Symbolic KG representations, including all approaches previously introduced except NNs, are typically more transparent than non-symbolic approaches. The opaqueness of NNs is due to the unclear semantics within the learned weights of NNs. Therefore, it can be more difficult to incorporate methods that provide explanations to end users non-symbolic KG representations than symbolic ones.

Scalability: NNs, directed graphs, and logics have all been used with large-scale datasets modeling large problem domains [132, 5, 3]. In contrast, probabilistic models are less efficient at handling a large problem domains. As for first-order probabilistic models specifically, scalable inference and learning are still open research problems.

3.10 Discussion & Conclusion

We covered seven different classes of KG representations and referenced prior work leveraging each in robot systems: (1) Directed Graph, (2) Markov Network, (3) Bayesian Network, (4) Partially Directed Acyclic Graph, (5) Logics, (6) First-order Probabilistic Models, and (7) Neural Networks. For each of these KG representations we discussed tradeoffs in terms of broad features desired of robot systems operating with non-expert end users. From the many works surveyed, we identified four broad qualities of KG representations prevalent in prior work: (1) Modeling uncertainty, (2) Adaptability, (3) Explainability, and (4) Scalability. Last we highlighted the KG representations that excel in each of these categories and discussed how to inform the selection of a KG representation for a robotics application.

3.11 Findings & Contributions

This work comprises our contributions to **a survey of KG representations that model semantic domain knowledge in robotics**. We did this by surveying robotics works that used KG representations to perform semantic reasoning. Additionally, we identified four broad qualities of KG representations. Our discussion of these qualities served to compare KG representations inform the matching of a representation to a robot application. The next two chapters of this thesis formulate and evaluate the use of MRE as a KG representation to inform a robot's decision making during autonomous task execution. Later chapters of this thesis serve to improve upon the inherent limitations of using MRE as a KG representation within the context of robot autonomy.

CHAPTER 4
MULTI-RELATIONAL EMBEDDINGS AS KG REPRESENTATIONS FOR ROBOTS

In this chapter we present our proposed KG representation based on MREs. The work in this chapter presents the foundation of our KG representation. As discussed in chapter 3, the term computational frameworks is used interchangeably here with KG representations, which are a subclass of computational frameworks. Contributions of later chapters build upon our framework to demonstrate its use in robot systems and improve its limitations with respect to adaptability and explainability.

4.1 Introduction

Robots operating in human environments benefit from encoding world information in a semantically-meaningful representation in order to facilitate generalization and domain transfer. Our work focuses on the problem of semantically representing a robot's world in a robust, generalizable, and scalable fashion. Semantic knowledge is typically modeled by a set of entities $\mathcal{E}$ representing concepts known to the robot (e.g. apple, fabric, kitchen), and a set of possible relations $\mathcal{R}$ (e.g. atLocation, hasMaterial, hasAffordance) between them [3, 4, 5, 6, 7, 8].

While some semantic information can be hard-coded, large-scale and long-term deployments of autonomous systems require the development of computational frameworks that i) enable abstract concepts to be learned and generalized from observations, ii) effectively model the uncertain nature of complex real-world environment, and iii) are scalable, incorporating data from a wide range of environments (e.g., hundreds of households). Previous work in semantic reasoning for robot systems has addressed subsets of the above challenges. Directed graphs [94] used in [5] allowed individual observations to adapt gen-

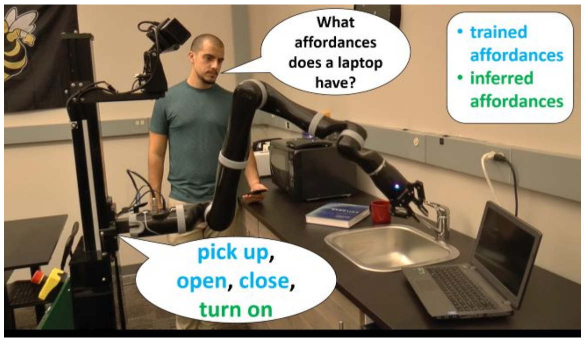

What affordances does a laptop have?
• trained affordances
• inferred affordances
pick up, open, close, turn on

Figure 4.1: RoboCSE can be queried by a robot to infer knowledge and make decisions.

eralized concepts at large scale, integrating multiple projects. BLN [120] in [4] allowed for precise probabilistic inference and learning assuming KGs have manageable sizes. DL [101] used in [15] allowed for large-scale deterministic reasoning about many concepts. In summary, each of these representations have limitations with respect to at least one of the three characteristics above.

We contribute Robot Common Sense Embedding (RoboCSE), a novel KG representation for semantic reasoning that is highly scalable, robust to uncertainty, and generalizes learned semantics. Given a KG $\mathcal{G}$, formalized in section 4.2, RoboCSE encodes semantic knowledge using *MREs* [28], embedding $\mathcal{G}$ into a high-dimensional vector space that preserves graphical structure between nodes and edges, while also facilitating generalization (see Figure 4.2a and Figure 4.2e). We show that RoboCSE can be trained on simulated environments (AI2Thor [136]), and that the resulting learned model effectively transfers to data from real-world domains, including both pre-recorded household scenes (MatterPort3D [137]) and real-time execution on a robot[1] (Figure 4.1).

We compare our work to three baselines: BLNs, Google's pre-trained Word2Vec embeddings [138], and a theoretical upper bound on the performance of logic-based methods [101]. Our results show that RoboCSE uses orders of magnitude less memory than BLNs[2] and outperforms the Word2Vec and logic-based baselines across all accuracy metrics. RoboCSE also successfully generalizes beyond the training data, inferring triples

[1]https://youtu.be/ynHwNotCkDA
[2]implemented using ProbCog [139]

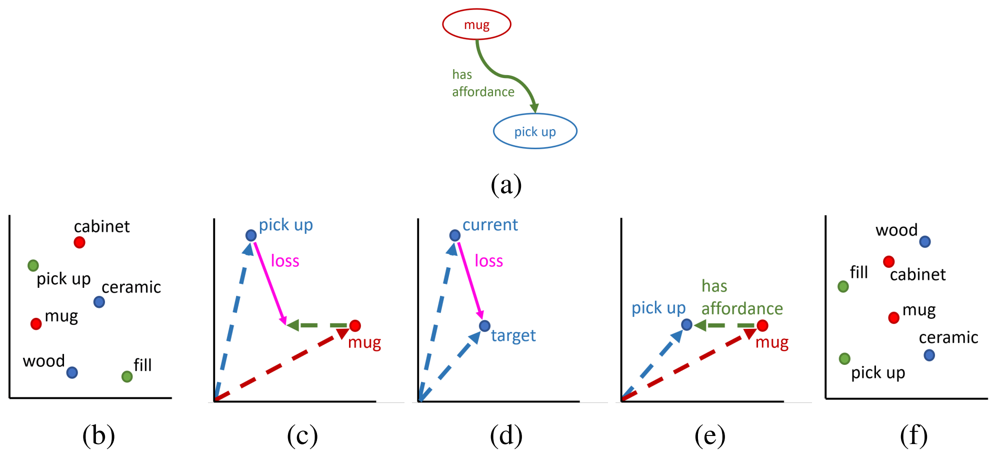


Figure 4.2: From a directed graph (Figure 4.2a) we can learn a vector embedding containing the same nodes and edges. MREs begin randomly initialized (Figure 4.2b) and are updated by calculating the losses between target transformations and actual transformations (Figure 4.2c-Figure 4.2d) until they converge on a semantically meaningful structure (Figure 4.2e-Figure 4.2f).

held-out from the training set, by leveraging latent interactions between multiple relations for a given entity. Furthermore, results returned by RoboCSE are ranked by confidence score, enabling robot behavior architectures to be constructed that effectively reason about the level of uncertainty in the robot's knowledge. Combined, the memory efficiency and learned generalizations of RoboCSE allow a robot to semantically model a larger world while accounting for uncertainty.

4.2 Background: Multi-Relational Embeddings

The objective of the multi-relational (i.e. KG) embedding problem is to learn a continuous vector representation of a KG $\mathcal{G}$, encoding vertices that represent entities $\mathcal{E}$ as a set of vectors $v_{\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}| \times d_{\mathcal{E}}}$ and edges that represent relations $\mathcal{R}$ as mappings between vectors $W_{\mathcal{R}} \in \mathbb{R}^{|\mathcal{R}| \times d_{\mathcal{R}}}$, where $d_{\mathcal{E}}$ and $d_{\mathcal{R}}$ are the dimensions of $\mathcal{E}$ vectors and $\mathcal{R}$ mappings, respectively [28, 27]. The KG $\mathcal{G}$ is composed from individual knowledge triples (h, r, t) such that $h, t \in \mathcal{E}$ are identified as head and tail entities of the triple, respectively, for which the relation $r \in \mathcal{R}$ holds (e.g. $($*cup, hasAffordance, fill*$)$). Collectively, the set of all triples from a dataset $\mathcal{D}$ form a directed graph $\mathcal{G}$ expressing the knowledge for that domain (note

$\mathcal{G}$ is considered incomplete because some set of triples may be missing).

Generically, a MRE is learned by minimizing the loss $\mathcal{L}$ using a scoring function $f(h, r, t)$ over the set of knowledge triples from $\mathcal{G}$. In addition to knowledge triples from $\mathcal{G}$, embedding performance substantially improves when negative triples are sampled from a negative triple KG $\hat{\mathcal{G}}$ [28]. Therefore, $\mathcal{L}$ is defined as $\mathcal{L}\big(f(h, r, t), y\big)$ where y is the positive or negative label for the triple.

4.3 RoboCSE

RoboCSE is a KG representation for semantic reasoning that uses MREs to encode abstract knowledge obtained by the robot from its sensors, simulation, or even external KGs (Figure 4.3). The robot can use the resulting KG representation as a queriable database to obtain information about its environment, such as likely object locations, material properties of objects, object affordances, and any other relation-based semantic information the robot is able to mine.

Figure 4.2 simplifies and summarizes the embedding process, conceptually. Training instances are provided in the form of knowledge triples (h, r, t, y) (Figure 4.2a). The embedding space containing all entity vectors has no structure before training (Figure 4.2b) because the relational embeddings must be learned. Therefore, all entities $\mathcal{E}$ and relations $\mathcal{R}$ are initialized as random vectors and mappings, respectively.

Each (h, r, t) training instance provided is used to perform stochastic-gradient-descent. Given a particular MRE, its loss function is used to compute a loss between a current vector and a target vector (Figure 4.2d). The current vector is calculated using a subset of the knowledge triple (e.g. *pick up* in Figure 4.2c) and the target vector is calculated using the remaining subset (e.g. *mug hasAffordance* in Figure 4.2c). The loss is used to update the appropriate $\mathcal{E}$ vectors and $\mathcal{R}$ mappings to better approximate the correct representation (Figure 4.2e).

The vectors and mappings of $\mathcal{E}$ and $\mathcal{R}$, respectively, converge to semantically meaningful values after repeating the training process with different subsets of knowledge instances (Figure 4.2e), which can be used to perform inference (see [28] for a survey on MREs). In Figure 4.2f we see that similar entities are grouped horizontally, cabinets are more likely to be filled than picked up, and mugs are equally likely to have either affordance.

We use the Analogy [30] MRE. Analogy represents relationships as (bi)linear mappings between entities, i.e., $\mathbf{v}_h^\top \mathbf{W}_r = \mathbf{v}_t^\top$ [30]. It uses the scoring and negative log loss functions in Equation 4.1 and Equation 4.2 where σ is a sigmoid function, y is a label indicating whether the triple is corrupted, and $\mathcal{G}'$ is the corrupted KG. Additionally, the linear mappings (i.e. relations) are constrained to form a commuting family of normal mappings, i.e., $\mathbf{W}_r \mathbf{W}_r^\top = \mathbf{W}_r^\top \mathbf{W}_r \,\forall\, r \in \mathcal{R}$ and $\mathbf{W}_r \mathbf{W}_{r'} = \mathbf{W}_{r'} \mathbf{W}_r \,\forall\, r, r' \in \mathcal{R}$, to promote analogical structure within the embedding space.

$$f(h, r, t) = \langle \mathbf{v}_h^\top \mathbf{W}_r, \mathbf{v}_t \rangle \tag{4.1}$$

$$\mathcal{L} = \sum_{(h,r,t,y) \in \mathcal{G}, \mathcal{G}'} -\log \sigma(y \cdot f(h, r, t)) \tag{4.2}$$

Inference in RoboCSE is done by completing a knowledge triple given only partial information. For example given $(h, r\_)$, RoboCSE returns a list of the most likely tails t_i to complete the knowledge instance. Mathematically, given $(h, r,\_)$, r maps an h by a high-dimensional geometric transformation, then the vectors with the highest f scores to the resultant are selected as results, which represent the most likely tails t_i. In the case of RoboCSE, which uses [30], r maps an h via $\mathbf{v}_h^T \mathbf{W}_r$. Result tails t_i are ordered using the bilinear scoring function f, in which higher scores be more likely (i.e. more closely aligned vectors). RoboCSE can make inferences about knowledge triples it has never seen before because these transformations can be done over any entities in the embedding space, allowing for generalization.

An assumption made widely across prior MRE work [33, 30, 27] is that query responses are deterministic (i.e. either always true with rank 1 or false with lower ranks), and only factual relations are provided in the training data. However, the semantic data we are

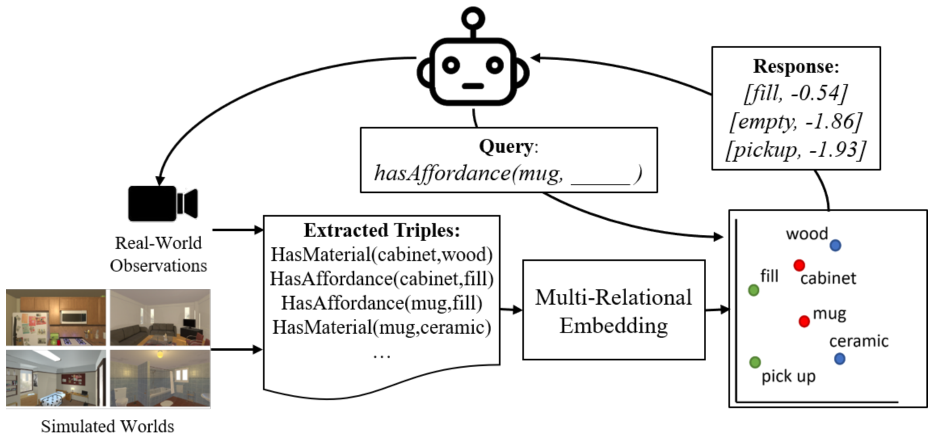


Figure 4.3: Overview of RoboCSE framework integrated with mobile robot.

modeling is highly stochastic; for example, multiple potential locations are likely for a given object. As a result, the ground truth rank of responses is often not 1. Instead, ground truth ranks reflect the number of observations in the data. To support ranks higher than 1 in our evaluation, we extend the standard performance metrics of MRR and Hits at top K (Hits@K), which assume a ground truth rank of 1, and for the experiments in section 4.5, we instead report:

$$\begin{gathered}\text{Hits@5*} = \frac{1}{N}\sum_{n=1}^{N} Hits_5(\mid G_r - I_r \mid) \\ s.t. \quad Hits_5 = \begin{cases} 1 \text{ if} \mid G_r - I_r \mid < 5 \\ 0 \text{ otherwise} \end{cases}\end{gathered} \tag{4.3}$$

$$\text{MRR*} = \frac{1}{N}\sum_{n=1}^{N} \frac{1}{\mid G_r - I_r \mid + 1} \tag{4.4}$$

where N is the number of triples tested, G_R is the ground truth rank, and I_r is the inferred rank. For both these metrics, scores range from 0 to 1 with 1 being the best performance. MRR* is a more complete ranking metric for which the inferred and ground truth ranks must match exactly to get a MRR* of 1. Hits@K* gives a more granular look at rankings and is informative of how often the correct response is ranked within the top K results. We discuss how the ground truth set and ranks are generated for each experiment in section 4.4.

Table 4.1: RoboCSE Knowledge Source

AI2Thor: 3 Relation Types, 117 Entities					
Median Count per Environment					
Env. Type	Loc. Rel.	Mat. Rel.	Aff. Rel.	Entities	Num. Rooms
Bathroom	28	21	46	18	30
Bedroom	28.5	16	54.5	20	30
Kitchen	59.5	51	109	27	30
Livingroom	22.5	8	37	20	30
All	29.5	18.5	50	20	120

4.4 Experimental Procedure

We evaluated RoboCSE's generalization capability on two scenarios: inferring the ranks of unseen triples (triple generalization) and accurately predicting the properties and locations of objects in previously unseen environments (environment generalization).

4.4.1 RoboCSE Knowledge Source: AI2Thor

Our semantic reasoning framework targets common sense knowledge for residential service robots. Knowledge embedded in RoboCSE was mined from a highly realistic simulation environment of household domains, AI2Thor [136]. AI2Thor offers realistic environments from which instances of semantic triples about affordances and locations of objects can be mined (see Table 4.1). Entities include 83 household items (e.g. microwave, toilet, kitchen) and 17 affordances (e.g. pick up, open, turn on). Additionally, we manually extended objects within AI2Thor to model 17 material properties (e.g. wood, fabric, glass), which were assigned probabilistically based on materials encountered in the SUNCG dataset [140]. The addition of material properties brought the total number of triples available for training, validation, and testing to over 15K. Note that while only the default environments (and therefore, true triples) were mined from AI2Thor, because object position and properties in the open-source simulator can be manipulated, many more environments and unique triples

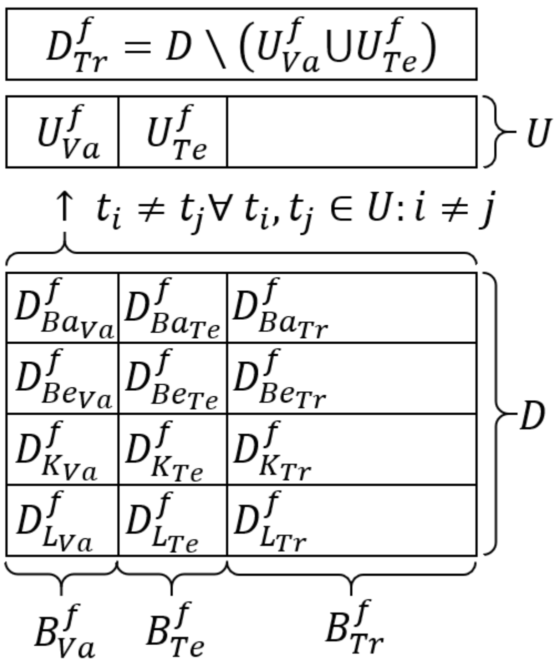


Figure 4.4: Diagram of triples contained in train, validation, and test sets for each fold (not to scale) during triple generalization (above) and environment generalization (below).

can be artificially generated for representations requiring more training data.

Prior work on MRE has shown that inclusion of negative examples in the training data, defined as triples known to be false, leads to improved training performance [28]. Therefore, we additionally trained on $(9 \times \text{number\_of\_true\_triples})$ negative examples for our domain to improve training performance. Similar to prior work, we used the closed world assumption to sample negative triples. However, we did not find that using the perturbing method suggested in [30] gave the best results. Instead, we observed empirically that better results were achieved after filtering perturbed triples to verify the sample was not in the training set.

4.4.2 Inferring Unseen Triples: Triple Generalization

Inevitably, an autonomous robot operating in a real-world environment will encounter problems that require answers to queries it was not trained on (e.g. *can mugs be filled?*). To probe how well RoboCSE can correctly generalize to do triple prediction, triple generalization performance was tested for each algorithm as follows.

Five-fold cross-validation was performed to estimate each evaluated algorithm's performance. To generate each fold, the set of all unique triples U was generated from the set of all triples in our test case dataset $\mathcal{D}$ where $U \subset \mathcal{D}$ by filtering out repeated triples (i.e. each triple $t \in U$ has the property $t_i \neq t_j \forall t_i, t_j \in U : i \neq j$) (see Figure 4.4). U was split into five equally sized sets of triples for folds U^f where $f \in \{1, 2, 3, 4, 5\}$. U^f was divided in half to create a validation portion U^f_{Va} and test portion U^f_{Te}. The training set for each fold D^f_{Tr} was generated from $\mathcal{D}$ by ensuring $\mathcal{D}^f_{Tr} \subset \mathcal{D}$ such that $\mathcal{D}^f_{Tr} \cap (U^f_{Va} \cup U^f_{Te}) = \emptyset$. For each fold, $\mathcal{D}^f_{Tr}$ was trained on while validating on U^f_{Va}, and the learned embedding was then tested on U^f_{Te}.

The training process follows the same procedure as in [30]. Testing was done by generating three ranks with each triple (i.e. rank h given $(h, r, \_)$, rank r given $(h, \_, t)$, and rank t given $(h, r, \_)$) then comparing them to their respective ground truth ranks. Each triple in the test set was a held-out triple ranked using the full-distribution of triples $\mathcal{D}$. Ground truth ranking was calculated according to the number of observations (i.e. more observations give higher rank). Error metrics similar to those from the relational embedding community (MRR* and Hits@K*) were calculated using the ground truth rank for comparison[3].

4.4.3 Applying Common Sense: Environment Generalization

Our second test targets the scenario of deploying a robot equipped with a semantic knowledge base in a new environment, with the goal of evaluating how well the embedded knowledge generalizes to new rooms and the degree to which a robot can use its knowledge to predict object properties or locations in the new setting (e.g. in a new house, where would I likely find a towel?). Environment generalization was tested as follows.

Five-fold cross-validation was performed to estimate each algorithm's performance

[3]Note that the * over both of our error metrics are to distinguish them from the standard error metrics used in relational embeddings which assume 1 as the reference rank.

over a test case dataset balanced across environment types (i.e. bathroom, bedroom, kitchen, livingroom). To generate each fold, $\mathcal{D}$ was separated into four sets for each environment type maintaining resolution at environment level (i.e. a single environment with all triples contained is an atomic unit for splitting purposes), $\mathcal{D}_{Ba}$ for bathrooms, $\mathcal{D}_{Be}$ for bedrooms, $\mathcal{D}_{K}$ for kitchens, and $\mathcal{D}_{L}$ for living rooms (see Figure 4.4). Then each environment type set $\mathcal{D}_E$ for $E \in \{Ba, Be, K, L\}$ was split at environment resolution into five equally sized sets of environments for folds $\mathcal{D}^f_E$ where $f \in \{1, 2, 3, 4, 5\}$. The smaller fraction of each fold of $\mathcal{D}^f_E$ was then divided in half to create a validation portion $\mathcal{D}^f_{E_{Va}}$ and test portion $\mathcal{D}^f_{E_{Te}}$, while the larger fraction served as a training set $\mathcal{D}^f_{E_{Tr}}$. Finally, the balanced train B^f_{Tr}, validation B^f_{Va}, and test B^f_{Te} sets were generated according to:

$$B^f_{Tr} = \bigcup_{e \in E} \mathcal{D}^f_{e_{Tr}} \quad B^f_{Va} = \bigcup_{e \in E} \mathcal{D}^f_{e_{Va}} \quad B^f_{Te} = \bigcup_{e \in E} \mathcal{D}^f_{e_{Te}} \tag{4.5}$$

The training process followed the same procedure as in [30]. Testing was done by querying the tested algorithm for triples that come from new environments, which have not been trained on found in each B^f_{Te} for folds $f \in \{1, 2, 3, 4, 5\}$. The standard MRR and Hits@K were used to measure the algorithm's performance, allowing us to assess how frequently the robot was correct on the first attempt.

4.5 Experimental Results

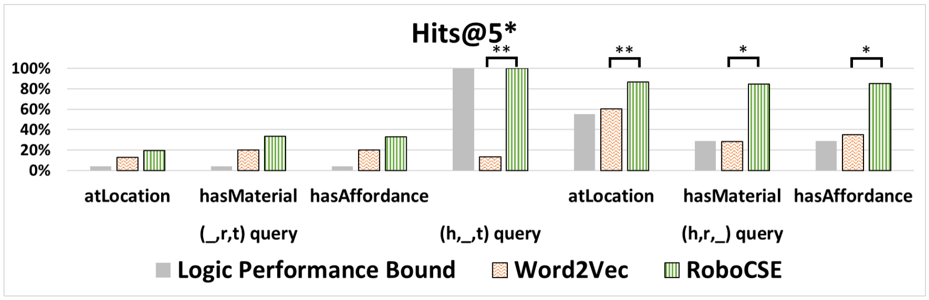
In this section, we report results characterizing the performance of various models trained on AI2Thor data to understand the advantages and limitations of RoboCSE. Pre-trained Word2Vec embeddings were used in Triple Generalization as a comparable baseline not within the class of MREs. An upper-bound on the performance of logic-based systems was also included in the Triple Generalization experiment to compare with more historically prevalent approaches [15, 9, 108]. For Environment Generalization and Domain Transfer, the KG $\mathcal{G}$ formed from the training set was used as a baseline baseline that memorizes the frequencies of triples in the training set. The KG is a controlled baseline that gives a clear indicator of how well RoboCSE generalized knowledge beyond what was available in the training set. Lastly, the memory requirements of RoboCSE were compared to a BLN because both account for uncertainty while modeling a graph of knowledge triples unlike Word2Vec or logic-based approaches. BLNs and MLNs were widely used in previous works but suffer from similar intractability problems [4, 15, 6]. Due to memory requirements, the BLN baseline could not be included in all experiments.

4.5.1 Testing Triple Generalization

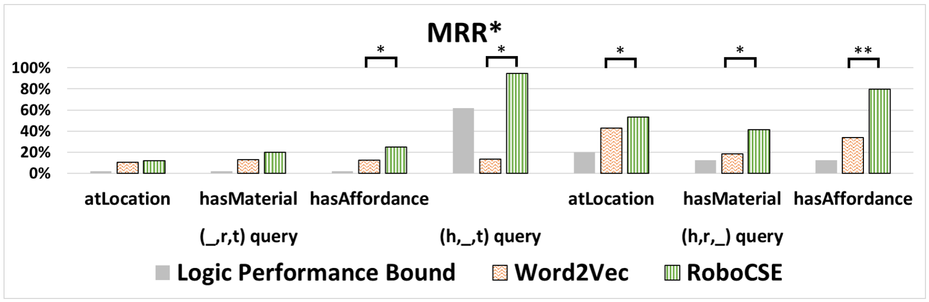
Triple Generalization was tested to quantify how well RoboCSE could infer missing triples using the learned KG representation (i.e. infer rank for *fork atLocaion kitchen*, not in the train set).

Two baselines were used to compare with RoboCSE, Word2Vec and DL performance upper-bound. The Word2Vec baseline first forms a 'comparison' group C of responses from all triples in the training set matching a test query (i.e. group heads from all training triples matching the query *(__,atLocation,cabinet)*). With the Word2Vec embeddings of C, the Word2Vec embedding of all candidate responses (i.e. all entities $\notin C$) are ranked using the cosine distance. We estimated the upper-bound of a DL based system at best be able to perform at type-specific chance (e.g. for a total of 17 affordances, guessing the correct affordance to *(mug,hasAffordance,__)* in the top five hits has a chance $\frac{5}{17}$). Upper-bounds of DL performance can be estimated because DL can determine the type of result that should be returned by a query but cannot infer which entity within a type would be most likely. Therefore the performance could be estimated for each query assuming type-specific chance (see Figure 4.5a). The bar graphs in Figure 4.5 show the performance of each algorithm w.r.t. Hits@5* and MRR* metrics for each relation and query type on the x-axis.

RoboCSE outperformed all baselines across all metrics at predicting unseen triples,



(a)

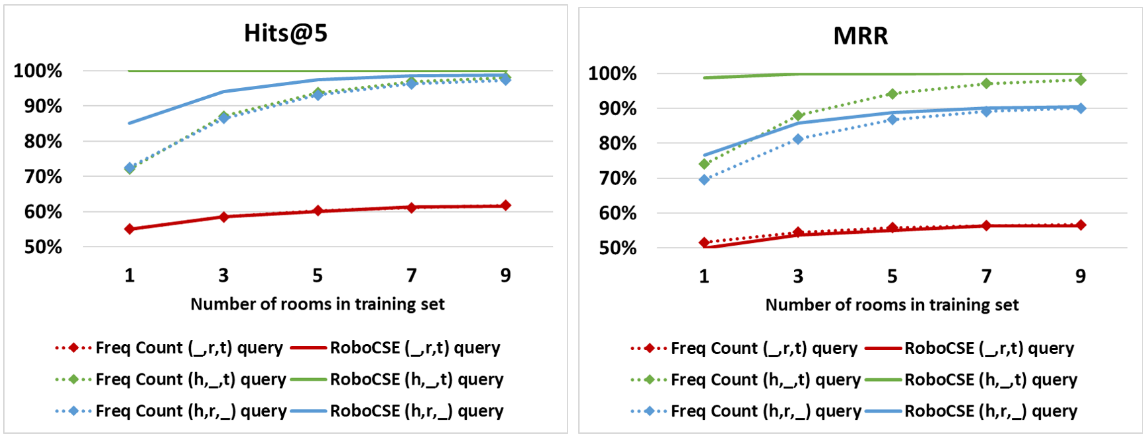


(b)

Figure 4.5: Performance w.r.t. Hits@5* and MRR* metrics for Triple Generalization in AI2Thor

which were statistically significant improvements on $(h,_-,t)$ and $(h,r,_-)$ queries compared using non-parametric 2 group Mann-Whitney U tests. The DL bound performs well for $(h,_-,t)$ queries because DL has explicitly defined types for all entities in a T-Box [101], allowing the framework to select the correct relation given a head and tail. The overall implication of these performance improvements is that a robot using RoboCSE to reason about a task could not only infer new knowledge it might not have been trained on to complete a task, but also reason about the confidence in the inferences to return the best result.

All algorithms performed worse at $(_-,r,t)$ queries than other queries, which is prevalent across our experiments. The drop in performance for $(_-,r,t)$ queries is because selecting the right entity as a head to complete a triple is a more difficult learning problem (a chance of $\frac{1}{74}$) versus selecting the correct affordance, material, or location, which are much fewer in number.



(a) (b)

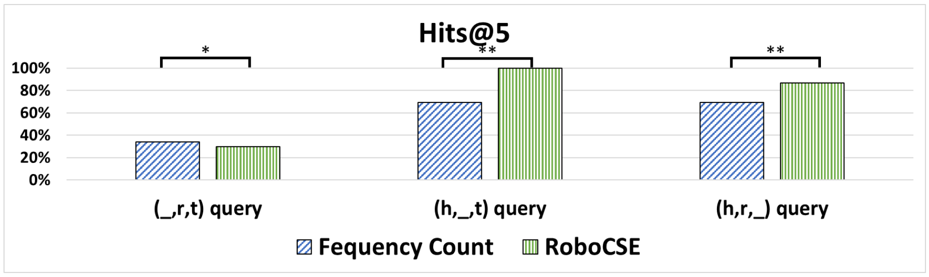
Figure 4.6: Performance Trends w.r.t. Hits@5* and MRR* Metrics for Environment Generalization in AI2Thor

4.5.2 Testing Environment Generalization

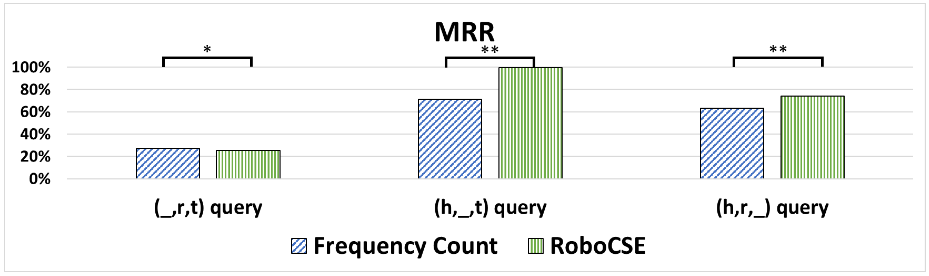
Environment Generalization was tested to measure how well RoboCSE could accurately complete triples in new rooms, motivated by real-world application of RoboCSE and the way training/deployment would proceed when a service robot encounters a new environment.

We compared RoboCSE to an instance-based learning baseline that memorizes the training set (i.e. frequency count) and the initial results showed these two methods were comparable. The baseline completed triples by selecting the most observed matching candidate (i.e. given query $(\_, r, t)$ it returned the head most often observed with the matching relation r and tail t). We trained each algorithm on 24 rooms of each type available and the results showed the baseline and RoboCSE had closely matching strong performances (whereby performance of each was within 1% of each other, >90% for $(h,\_, t)$ and $(h, r,\_)$ queries and >40% for $(\_, r, t)$). The similarity in performance between our approach and the baseline was because the default rooms of AI2Thor do not have enough variety between rooms (i.e. algorithms rarely have to generalize to unseen triples).

However, reducing the number of rooms reveals RoboCSE's ability to learn from the interactions of triples and generalize to the best performance faster than the baseline (see



(a)



(b)

Figure 4.7: Performance w.r.t. Hits@5 and MRR metrics for Domain Transfer to MatterPort3D

Figure 4.6). Lines in Figure 4.6 were generated by averaging across relations for each query type at varying numbers of rooms in the training set. The trend of RoboCSE generalizing to new rooms faster than the baseline was most pronounced with the fewest number of rooms in the training set (i.e. 1) but continued up to about 9 rooms as shown in the line plots. RoboCSE's generalization across rooms was most pronounced for the $(h,_{-},t)$ and $(h,r,_{-})$ queries on both metrics. Therefore, a robot bootstrapped with RoboCSE can learn general structures from individual instances to perform better in new environments and require less training data.

4.5.3 Domain Transfer: Testing on MatterPort3D

The learned embeddings from AI2Thor were tested on MatterPort3D (MP3D) to measure how well RoboCSE transfers to *envrionments from* real-world domains. While MP3D does not contain all the object properties we included in AI2Thor (no affordances or materials), it does contain triples about object locations for over 500 real-world environments.

The results from domain transfer showed that RoboCSE generalized to MP3D better than our instance-based learning baseline that memorizes the training set (i.e. frequency count), effectively inferring new triples not present in the training data. Training and validation for domain transfer closely followed the Environment Generalization procedure (see subsection 4.5.2) but only for *atLocation* relations. During testing, the models learned from all rooms in AI2Thor were used to answer queries about all rooms in MP3D. The bar graphs in Figure 4.7 show that the semantics learned in AI2Thor can be directly applied to MP3D, evident in the high performance of both algorithms. Furthermore, inference in RoboCSE successfully generalized beyond training data to accurately infer more queries indicated by the statistically significant higher scores RoboCSE gets on $(h,_{-},t)$ and $(h,r,_{-})$ queries compared using non-parametric 2 group Mann-Whitney U tests. In short, bootstrapping a robot with semantics learned in simulation using RoboCSE can be applied to data from real world environments.

4.5.4 Analyzing Memory Requirements

We analyzed the memory requirements of RoboCSE and BLNs [120] to compare the scalability of each.

To analyze memory requirements, all unique triples from AI2Thor were extracted (352) and modeled in a BLN using a standard package (ProbCog [139]). The resulting BLN required 9 orders of magnitude more memory than RoboCSE (i.e. 100 TB vs. 96 KB). Although BLNs have been used to model semantic knowledge within robot systems to do accurate probabilistic reasoning [15, 4], maintaining conditional-probability tables in BLNs can be intractable due to the rapid increase of node in-degree (i.e. number of parents) and therefore table size, for densely connected networks. For example, let the network structure G of a BLN be $G = (V, E)$ where V represents all the nodes of the network, E are all edges between nodes, and v_i is any node in the network such that $v_i \in V$. Noting the in-degree of v_i as $deg^{-}(v_i)$ and n_V as the number of nodes in V, then the number of entries in all conditional-probability tables within a network G will be $\sum_{i=1}^{n_V} 2^{deg^{-}(v_i)+1}$.

In-degree of nodes in BLNs increases rapidly because edges between nodes are often connected with specific relation types *atLocation(apple, fridge)* but BLNs cannot store relation type information on edges. Therefore, all relations need to be modeled as predicates such as *atLocation(x, fridge)*, which are stored in nodes. As a result, there are nodes with very large in-degree counts even for simple directed graphs. The issue holds for Markov Logic Networks [119], which have also been leveraged to model KGs [6].

RoboCSE's drastic memory reduction was possible because its space complexity scales linearly with the number of entity or relation types and RoboCSE's space complexity does not directly depend on node in-degree. RoboCSE requires $(|\mathcal{E}| + |\mathcal{R}|) \times d \times 8$ bytes of memory, where d is the vector space dimensionality. Although RoboCSE provides a considerable improvement in space complexity, RoboCSE cannot represent the joint distribution or true probabilities as a BLN can. Instead, the distances measured using a scoring function between the queried transformation and results are interpreted as confidence (see section 4.3). Furthermore, only the subset of the triple in the query can be used as 'evidence' to condition on (e.g. the best h_i are selected conditioned on an $(\_, r, t)$ query).

4.5.5 Implementation on a Mobile Manipulator

As a final demonstration, we deployed our KG representation to a mobile manipulator, shown in Figure 4.1. We used an object detector [141] to get the semantic names of objects in the mobile manipulator's camera view. The semantic names were then used to query RoboCSE for all likely relationships about the detected objects witin some belief threshold (e.g. affordances, likely locations). The ranked responses to each query were then uttered by the robot through speakers. Our robot implementation[4] demonstrates a simple proof of concept on how the KG representation could be deployed to a robot.

[4]https://youtu.be/ynHwNotCkDA

4.6 Discussion & Conclusion

We approached the problem of semantically representing a robot's world via (h, r, t) triples in a manner that supports generalization, accounts for uncertainty, and is memory-efficient. We presented RoboCSE, a novel framework for robot semantic knowledge that leverages mutli-relatonal embeddings.

From our experiments two benefits have emerged from the use of MREs in RoboCSE: (1) the generalizations learned outperformed Word2Vec at prediction, being robust to significant reductions in training data and domain transfer and (2) RoboCSE used orders of magnitude less memory to represent projections of graphs than representations of the same graph with BLNs. The collectively distinct set of benefits MREs have to offer could be taken advantage of to further progress for robots performing semantic reasoning robustly in semantically rich environments.

However, leveraging MRE has its limitations. As previously mentioned, answering $(\_, r, t)$ queries is particularly difficult. Answering $(\_, r, t)$ queries is useful for robots reasoning to plan tasks (i.e. which head satisfies *(_,hasAffordance,fill)*). Secondly, conditioning is very limited compared to a BLN. The limited conditioning of MREs leads to the same responses in different environments. Third, realistic systems in long-term deployments need the ability of incremental learning, enabling online adaptations as new knowledge arrives, which is not possible in our formulation. Last, robots working with end users should have explainable inferences that support its decision-making, yet the inference mechanism of MRE is not interpretable to end users.

4.7 Findings & Contributions

This work comprises our contributions to **the development and evaluation our KG representation based on multi-relational embeddings**. We did this by taking individual semantic common-sense knowledge facts from simulation forming a KG about the household environments. The gathered facts were used to train a continuous vector representation of the KG that could be queried to infer new facts not present in the given KG. Our experiments showed that our KG representation outperformed word embeddings while being robust to large reductions in training data and domain transfer. Additionally, our KG representation used orders of magnitude less memory than BLNs to represent the same KG. While MREs do have limitations detailed in section 4.6, we believe the collectively distinct set of advantages of MREs from prior KG representations could further progress of semantic reasoning on robot systems. Although our robot implementation provided a proof of concept, a more complete set of modules is needed to allow the robot to perform task planning and execution informed by our KG representation, as this thesis details in the next chapter.

CHAPTER 5
ROBUST TASK EXECUTION USING KG REPRESENTATIONS

In this chapter we present a method to integrate our KG representation with a robot to enable task planning and execution informed by the representation. We build upon the KG framework in chapter 4 to integrate our KG representation with a task execution stack from section 2.6. Our simulation and robot experiments in this chapter highlight the advantages of informing the planning level of a robot architecture with domain knowledge. Additionally, these grounded experiments make clear some limitations of applying MRE to robotics. These limitations are addressed in later chapters of this thesis, namely whether MREs can be sequentially trained to update the represented KG and whether MRE inference can be explainable to end users.

5.1 Introduction

Robust one-shot task execution is an ongoing challenge in robotics. Learning from Demonstration (LfD) provides the means for end users to program new task plans (structured sequences of abstract primitive actions [142]). Typically, multiple varying demonstrations are required in order to learn task plans that are resilient to failures in execution of primitive actions. Requiring multiple demonstrations can present an additional burden to the end user. However, task plans that are learned in domains rich with semantic knowledge can benefit from incorporating such domain knowledge into the task plan. Task plan constituents from a prototypical or demonstrated task plan can be effectively generalized to new execution environments by leveraging statistical correlations or heuristic features learned from the task domain [16, 13].

We address the problem of one-shot task execution, in which a robot must generalize a *single* demonstration or prototypical example of a task plan to a new execution environ-

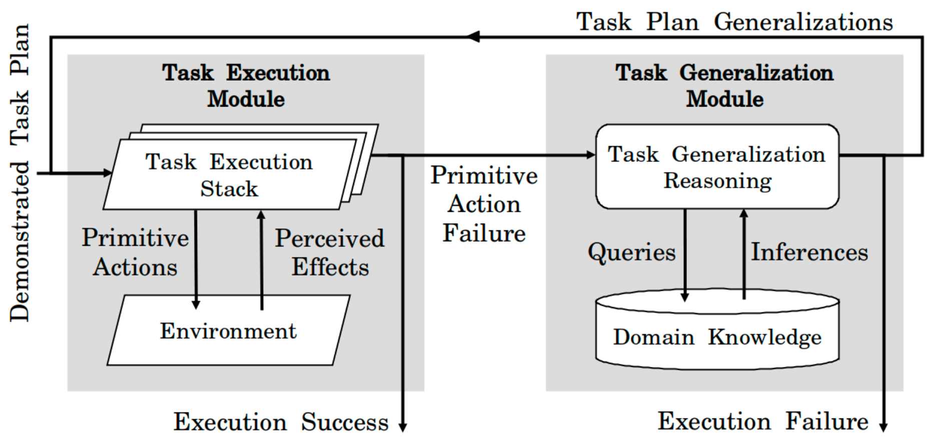


Figure 5.1: The task generalization module incrementally generalizes failed task plans by leveraging the learned KG to infer plan constituents (see section 5.3). The task execution module sequentially executes primitive actions to complete the task plan.

ment. For example, the robot could be shown a demonstration of cleaning a kitchen counter using a napkin found in the kitchen cabinet. The robot may have to repeat the task plan in a new environment where napkins may be kept in another location, or are unavailable altogether. Such differences between environments typically halt execution of the task plan due to unsatisfied preconditions when searching for the missing object, causing a primitive action failure.

Semantic knowledge about real-world domains tends to have the properties of being *large*, *sparse*, and *incomplete*. For example, a KG representing semantic knowledge about households, while large, only contains a subset of true facts, which are sparse with respect to the space of many potential facts. For this reason, our KG representation uses a MRE, which is a distributed representation that models a KG in vector space [19]. We posit that MREs are well suited to model semantic knowledge about real-world domains because MREs are designed for KGs that are large and sparse [2]. Additionally, MREs excel at learning the underlying structure of KGs to infer new facts beyond those present in a KG. We integrate MREs as a KG representation along with the task plan, in a *task generalization module*. Our developed task generalization module infers appropriate task plan constituents by reasoning about the learned domain knowledge. Task plan generalizations are then used by a task execution module when the demonstrated task plan fails. The task execution module sequentially executes primitive actions on an environment to complete the task plan. Our implemented architecture (Figure 5.1), enables one-shot task execution both in simulation and on a physical robot.

We compare our domain knowledge representation against four representative baselines in the context of robust one-shot task execution, including prior MREs [143], Markov Logic Networks [144, 3], Plan Networks [145], and word embeddings [18]. In simulation, we evaluate each method's ability to generalize an initial demonstrated task plan to 300 execution environments, averaged across 40 different initial demonstrations for a total of 12,000 executions. Our experiments in simulation indicate that our knowledge representation provides improvements in success rate over the baselines with strong statistical significance. We follow these experiments with an ablation study that characterizes the variances of our performance metrics with respect to components of the task generalization module. Lastly, we validate the implemented architecture on a physical robot that must autonomously execute an initial demonstrated task plan in 10 new execution environments, for 5 different initial demonstrations. Our system successfully generalizes the initial task plans to 38 of 50 sampled execution environments, where failures stem solely from robot errors, such as precarious object grasps. Ignoring these robot errors, our system successfully generalized the initial task plan in every execution environment. Our contributions are summarized as follows.

- We design a new robot KG that adds entity and relation types not present in prior work [143];
- We design a task generalization approach that integrates the MRE KG representation with multiple task reasoning levels to generalize task plans;
- We demonstrate our approach's advantages against representative baselines in 12,000 simulations and validate the implemented architecture on a physical robot performing one-shot task execution in 50 execution environments.

5.2 Problem Definition

Our work is motivated by applications in which a robot operates in dynamic environments, such as households, with naive users who provide a demonstration of a task plan T_d. The task plan is defined as a sequence of primitive actions $\{a_d^1, a_d^2, ..., a_d^k\}$, where each primitive action may or may not be parameterized by objects. The execution environment can be different from the demonstration environment in terms of the environment state, such as differing object types available for the task. As a result, the demonstrated task plan fails due to unsatisfied pre-conditions of primitive actions. Therefore, the robot must generalize the demonstrated task plan to formulate an *executable* task plan T_x. We then formulate our problem as:

Given an execution environment E_x and a task plan T_d recorded from a demonstration in a demonstration environment E_d, can a modified task plan T_x be formulated for E_x such that the robot is able to accomplish the task?

We assume that unsatisfied pre-conditions stem solely from environment state changes (perturbations) that prevent the completion of a primitive action. Perturbations are limited to varying the type or location of the object available to perform the task. We note that this assumption commonly applies to household settings where objects do not have static locations or objects available for tasks vary between households. We consider other sources of primitive action failures, such as hardware failure or manipulation errors, out of scope.

5.3 Approach

Our approach generalizes a failed demonstrated task plan by reasoning about the primitive actions and the learned domain knowledge, as shown in Figure 5.1. When the execution of a demonstrated task plan T_d is halted due to a failed primitive action, the task generalization module is called to infer a task plan T_x for the execution environment E_x. The task generalization module generalizes the demonstrated task plan by iteratively querying the learned domain knowledge representation and making incremental updates to the task plan. The updated task plan is then sent back to the task execution module to resume the task from the failure point. In particular, we integrate our approach with a task execution module developed in prior work for mobile manipulator robots [77]. We provide further details on the components of the task generalization module below.

5.3.1 Knowledge Representation

Our domain knowledge representation uses an explicit model of world semantics in the form of a KG $\mathcal{G}$ composed of individual facts or triples (h, r, t) with h and t being the head and tail entities (respectively) for which the relation r holds, e.g. (*cup, hasAction, fill*) [8, 4, 5, 6]. We model $\mathcal{G}$ using graph embeddings because of their ability to learn the underlying structure of graphs and infer new facts beyond known facts in a graph [18, 12, 143, 146, 147, 11]. We build upon the framework in RoboCSE [143], which uses a MRE to represent $\mathcal{G}$.

MREs model $\mathcal{G}$ in vector space [27], learning a continuous vector representation from a dataset of triples $\mathcal{D} = \big\{(h, r, t)_i, y_i |\, h_i, t_i \in \mathcal{E}, r_i \in \mathcal{R}, y_i \in \{0, 1\}\big\}$, with $i \in \{1...|\mathcal{D}|\}$. Here y_i denotes whether relation $r_i \in \mathcal{R}$ holds between $h_i, t_i \in \mathcal{E}$. Each entity $e \in \mathcal{E}$ is encoded as a vector $\mathbf{v}_e \in \mathbb{R}^{d_\mathcal{E}}$, and each relation $r \in \mathcal{R}$ is encoded as a mapping between vectors $\mathbf{W}_r \in \mathbb{R}^{d_\mathcal{R}}$, where $d_\mathcal{E}$ and $d_\mathcal{R}$ are the dimensions of vectors and mappings respectively [27, 28]. The embeddings for $\mathcal{E}$ and $\mathcal{R}$ are typically learned using a scoring function $f(h, r, t)$ that assigns higher (lower) values to positive (negative) triples [28]. The learning objective is thus to find a set of embeddings $\Theta = \big\{\{\mathbf{v}_e |\, e \in \mathcal{E}\}, \{\mathbf{W}_r |\, r \in \mathcal{R}\}\big\}$ that minimize the loss $\mathcal{L}$ over the training split $\mathcal{D}_{Tr}$ of the dataset $\mathcal{D}$. Loss $\mathcal{L}_\mathcal{D}$ can take many forms depending on the KGE representation used, e.g., Margin-Ranking Loss [29] or Negative Log-Likelihood Loss [30]. The learned embeddings Θ are used to infer the likelihoods of facts in the held out splits of the dataset (i.e. $\mathcal{D}_{Va}$ and $\mathcal{D}_{Tr}$), which are not present in the training split.

We use the Analogy embedding representation as in [143]. Analogy represents relationships as (bi)linear mappings between entities, i.e., $\mathbf{v}_h^\top \mathbf{W}_r = \mathbf{v}_t^\top$ [30]. It uses the scoring and negative log loss functions in Equation 5.1 and Equation 5.2 where σ is a sigmoid function, y is a label indicating whether the triple is corrupted, and $\mathcal{G}'$ is the corrupted KG containing only false triples. Additionally, the linear mappings (i.e. relations) are constrained to form a commuting family of normal mappings, i.e., $\mathbf{W}_r \mathbf{W}_r^\top = \mathbf{W}_r^\top \mathbf{W}_r \, \forall \, r \in \mathcal{R}$ and $\mathbf{W}_r \mathbf{W}_{r'} = \mathbf{W}_{r'} \mathbf{W}_r \, \forall \, r, r' \in \mathcal{R}$, to promote analogical structure within the embedding space.

$$f(h, r, t) = \langle \mathbf{v}_h^\top \mathbf{W}_r, \mathbf{v}_t \rangle \tag{5.1}$$

$$\mathcal{L} = \sum_{(h,r,t,y) \in \mathcal{G}, \mathcal{G}'} -\log \sigma(y \cdot f(h, r, t)) \tag{5.2}$$

We extend the original KG presented in [143] by adding new relation types and entity types. We added 9 new relation types and a new entity type that represents action effects. Triples were extracted from VirtualHome simulations as in [143], using the expanded set of relations and entities. Table 5.1 presents examples and statistics about the newly extracted KG from VirtualHome. Our experimental evaluations show that these additions to the KG significantly improve the generalization capabilities of the task generalization module against the prior KG.

5.3.2 Task Generalizations

We focus on generalizing object-oriented primitive actions, and denote an action $a_d^i \in T_d$ associated with an object o_d present at a location l_d, as $a_d^i(o_d, l_d)$. In our notation, any action $a_d^i(o_d, l_d)$, e.g., "Scrub(scrubber, shelf)", is *not* the parameterization of the action itself. Instead, it indicates that the action "scrub" uses the object "scrubber", which is found on the "shelf". We assume object oriented primitive actions fail for 3 types of reasons, o_d not being found in the demonstrated location l_d, o_d not being present in the environment to perform a_d^i, or the environment lacking any object that can be used to perform a_d^i. These three failure types lead to three levels of reasoning about the task plan constituents, that

Table 5.1: Dataset extracted from VirtualHome to learn $\mathcal{G}$

Relation	$\lvert E_{head}\rvert^{\dagger}$	$\lvert E_{tail}\rvert^{\dagger}$	$\lvert\mathcal{D}_{Tr}\rvert^{\ddagger}$	$\lvert\mathcal{D}_{Va}\rvert/\lvert\mathcal{D}_{Te}\rvert^{\dagger}$	$\lvert\mathcal{D}\rvert^{\dagger}$
HasEffect	28	10	39,263 (24)	2	28
InverseActionOf	10	10	29,956 (10)	1	12
InverseStateOf	15	15	23,763 (13)	1	15
LocInRoom	41	4	3,972 (78)	9	96
ObjCanBe	159	33	45,075 (886)	110	1106
ObjInLoc	164	28	9,461 (409)	51	511
ObjInRoom	164	4	8,276 (289)	36	361
ObjOnLoc	149	32	2,346 (269)	33	335
ObjUsedTo	71	21	6,224 (76)	9	94
ObjHasState	153	15	28,306 (431)	53	537
OperatesOn	68	125	84,286 (1124)	140	1404

Example entities (281 total entities)	
Rooms (4)	kitchen, bedroom, bathroom, livingroom
Locations (45)	fridge, table, sink, garbage, bed, desk, cabinet, drawer
Objects (182)	chair, towel, bleach, tomato, rug, plant, fork, laptop
Actions (34)	wipe, open, pick up, turn off, bake, unplug, disinfect
States (16)	dirty, clean, on, off, cooked, broken, open, plugged in

†|unique instances|,‡ |instances|(|unique instances|)

can be combined and used to generalize the demonstrated task plan. We categorize these generalizations based on the specific variable that each is associated with, namely, l_d, o_d and/or a_d. With each reasoning level, the robot generalizes up to 1, 2 or 3 types of variables:

1. **Spatial Reasoning**: This reasoning level only requires spatial generalizations to generate an executable plan.
    - Location (L): These generalizations arise when o_d is in a different location than l_d in the execution environment. Other likely locations for o_d are inferred.
2. **Object Reasoning**: This reasoning level requires object reasoning and potentially, spatial generalizations.
    - Object (O): These generalizations arise when o_d is not available in the execution environment, but an appropriate substitute object that can be used to perform a_d is available at the location l_d. Other appropriate objects to perform a_d are inferred.
    - Object-Location (OL): These generalizations like O lack the demonstrated object o_d,

but the appropriate substitute is not in the demonstrated location l_d. Other appropriate objects to perform a_d are inferred as well as their corresponding likely locations.

3. **Action Reasoning**: This reasoning level requires both action and object generalizations, and potentially, spatial generalizations as well.
    - Action-Object (AO): These generalizations arise when all objects associated with a_d are unavailable in the execution environment. An appropriate action that achieves the same effect as a_d, and a corresponding object that is a valid parameterization of the action must be inferred.
    - Action-Object-Location (AOL): These generalizations are similar to AO, but additionally, the substitute object is not available at the demonstrated location. Therefore, an appropriate action, object, and location must be inferred to generalize the task plan.

We make a simplifying assumption that actions with the same effects have the same preconditions. The implications of our assumption are lessened by using auxiliary queries to the KG $\mathcal{G}$ that infer which demonstrated objects are applicable to the task. Our future work will relax this assumption to learn and infer the full parameterizations of actions. In the following section, we describe the task generalization module's implementation of the three generalizations above that integrates the knowledge representation in subsection 5.3.1 with the task plan.

5.3.3 Task Generalization Module

In algorithm 1 we provide our approach to one-shot task execution. The task generalizations in subsection 5.3.2 are implemented in algorithm 1 using the computational framework in subsection 5.3.1. The pseudocode for algorithm 1 is written as a callable Python object with three queues as class members that maintain state across function calls. In algorithm 1 we incrementally update the failed task plan passed as the input parameter T, which is initially the demonstrated task plan T_d. The incremental task plan updates and ex-

Algorithm 1: Task Generalization Algorithm

```
Function generalize_task(T):
    $l_f$, $o_f$, $a_f$ = extract_failure(T)
    if $l_{subs}$ is $\emptyset$ then
        $l_{subs}$ = infer $l_{subs}$ for $o_f$ from $\mathcal{G}$
    $l$ = next($l_{subs}$)
    if $l \neq$ None then // Spatial Gen.
        if $\exists l$ then
            $T_x$ = replace($l_f, l, T$)
            return $T_x$
        else
            goto line 5
    if $o_{subs}$ is $\emptyset$ then
        $o_{subs}$ = infer $o_{subs}$ used to $a_f$ from $\mathcal{G}$
    $o$ = next($o_{subs}$)
    $l_{subs} = \emptyset$
    if $o \neq$ None then // Object Gen.
        $T$ = replace($o_f, o, T$)
        $o_f = o$
        goto line 3
    if $a_{subs}$ is $\emptyset$ then
        $a_{subs}$ = infer $a_{subs}$ with effect($a_f$) from $\mathcal{G}$
    $a$ = next($a_{subs}$)
    $o_{subs} = \emptyset$
    if $a \neq$ None then // Action Gen.
        $T$ = replace($a_f, a, T$)
        $a_f = a$
        goto line 12
    return Task Failure
```

ecutions continue until an executable task plan T_x is found or the algorithm cannot find an executable task generalization for the environment (i.e. returns Task Failure). Each incremental update to the task plan T is achieved by replacing all mentions of the variable that is being generalized in the task plan T, denoted by the *replace* function. We implemented algorithm 1 to prioritize generalizations involving fewer variables (i.e. generalizing one type of variable is preferable to two). Our approach leads to simpler generalizations that more closely match the demonstrated task plan.

Our approach in algorithm 1 uses the three levels of generalization discussed in subsection 5.3.2 depending on the variable being reasoned about (location, object, or action). Each level of reasoning uses different sets of queries to infer generalizations from the knowledge representation. The MRE infers location generalizations by completing the triples $(o_f, \text{ObjAtLoc}, \_)$ and $(o_f, \text{ObjInRoom}, \_)$. The inference results of these incomplete triples are combined to produce a prioritized queue of each likely location instance in the environment for o_f. Similarly, the MRE infers object generalizations by completing the triples $(o_f, \text{ObjUsedTo}, \_)$ and $(o_f, \text{OperatesOn}, \_)$. The inference results of these incomplete triples are combined to produce a prioritized queue of each viable object class that can perform the desired action on the target object. Lastly, when inferring action generalizations, the MRE completes the triple $(\_, \text{ActionHasEffect}, \text{effect}(a_f))$. The inference results of this incomplete triple produce a prioritized queue of each viable action class that achieves the desired effect. The queue for each reasoning level has a fixed size, which is tuned and discussed further in subsection 5.4.2.

5.4 Experimental Evaluation

Our experimental evaluations model the task domain of cleaning since it is a commonly desired task [1] and involves a variety of objects and actions. We perform evaluations both in simulation using VirtualHome [148], and on a physical robot (Fetch [76]). Our *simulation* experiments included 7 cleaning actions (“wipe”, “dust”, “sweep”, “wash”, “rinse”, “disinfect”, and “scrub”), 28 cleaning objects, and 45 possible object locations leading to 8,820 possible generalizations for a primitive action $a_d^i(o_d, l_d)$. Our *robot* experiments included 4 cleaning actions (“wipe”, “dust”, “sweep”, and “scrub”), 5 cleaning objects, and 12 possible object locations leading to 240 possible generalizations for a primitive action $a_d^i(o_d, l_d)$. The large number of possible task generalizations in both cases highlights the potential of incorporating learned domain knowledge into the task demonstration, as well as the challenge when reasoning about the most likely generalizations given the scale and complexity of probabilistic distributions within the domain.

We performed 3 evaluations; first, to broadly compare existing knowledge representations against ours, second to further analyze components of algorithm 1 in ablation studies, and last to validate our approach on a physical robot. We tested the statistical significance of results using repeated-measures ANOVA and a post-hoc Tukey's test.

5.4.1 Comparison of Knowledge Representations

We focus on comparing our knowledge representation to baselines representative of prior work in this experiment, in the context of robust task execution. The baselines were:

1. **Single Demo (SD)** serves as a reference for how often the original demonstration is randomly spawned.

2. **Plan Network (PN)** answers queries by building a repository of valid location, object, and action substitutes observed across consecutive executions, similar to [145]. The repository is built from demonstrations obtained from an oracle (i.e. solutions of test execution environments).

3. **Word Embedding (WE)** answers queries using cosine similarity over ConceptNet Numberbatch v19.08 embeddings [149]. Word embeddings have been used in robotics works as semantic representations [18, 12, 146].

4. **Markov Logic Network (MLN)** answers queries by learning statistical correlations relating objects and object attributes as in [13, 6] trained on the dataset in Table 5.1.

5. **Original RoboCSE (RCSE)** answers queries using the MRE framework in [143] that lacks the entity and relation types added in this work. Trained on the dataset in Table 5.1.

6. **Training Set Memorization (TM)** answers queries using the KG $\mathcal{G}$ formed from training triples in Table 5.1 without an MRE (i.e. lacks inference of valid/test triples).

Each simulation experiment in VirtualHome begins by spawning an agent in an initial environment E_d. A hierarchical task network provides the agent with a task plan T_d in the initial environment E_d that serves as the single task demonstration. The demonstrated task

Table 5.2: Metrics for 12,000 Environment Executions

Metric	SD	PN	WE	RCSE	TM	Ours
Success Rate (%)	1.2±1.3	54.6±2.6	9.8±7.8	19.9±6.1	68.8±3.3	**73.7±2.6**
Num. Attempts	1±0	17±14	30±33	62±34	32±14	67±28

plan T_d defines an initial cleaning task by performing a cleaning action a_d using a demonstration object o_d found in a demonstrated location l_d. Then, the agent must execute the task plan T_d in an execution environment E_x, which is a perturbation of the demonstration environment E_d. If a primitive action fails when executing T_d, the agent iteratively invokes the task generalization module until it finds an executable task plan T_x or fails.

We track the *number of attempts* the agent makes at completing the task plan as well as whether it *succeeds*. The agent must attempt to generalize the demonstrated task plan to 300 other randomly generated execution environments. Execution environments are generated by perturbing the object used to demonstrate the task; changing the object's location, type, or both. The perturbations are made in accordance with the action-object-location distributions present in VirtualHome, ensuring that objects are not placed at implausible locations (e.g., broom inside the toilet) and that the intended generalization is not unreasonable (e.g. cleaning a table with a washing-machine). Each execution environment has exactly one valid solution.

We repeat the experiment 40 times with different initial demonstrations to avoid overfitting reported results to an initial demonstration that is an outlier, totalling 12,000 sampled execution environments. Note that the domain knowledge representation was the only variable across the experiments, while other components of the system architecture in Figure 5.1, initial demonstrations, and perturbations were controlled. For all methods, we tuned the sizes of prioritized queues in algorithm 1 to 12 for locations, 8 for objects, and 4 for actions using an analysis described in subsection 5.4.2.

Our major results are summarized by Table 5.2 and Figure 5.2. In Table 5.2 we show the success rate and number of attempts for each approach averaged across the 40 experiment trials. As shown in Table 5.2, Ours provides improvements in success rate compared against

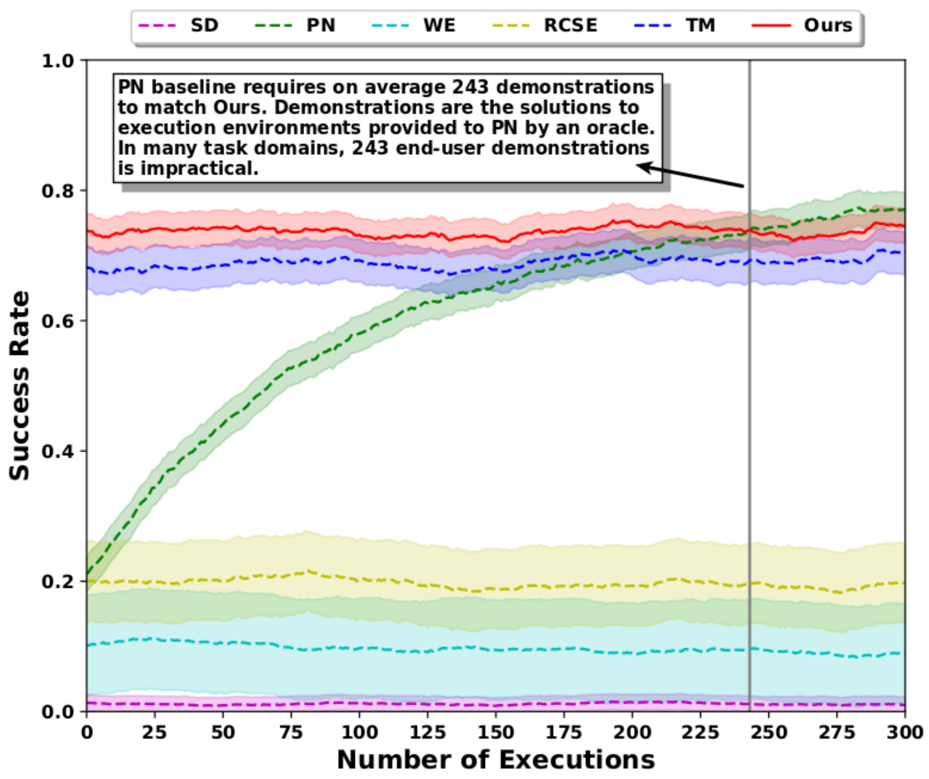


Figure 5.2: Moving average success rate of 60 executions for all tested task generalization approaches.

all the baselines with *strong statistical significance* (i.e. $p < 0.001$). Without the capacity to directly model relations, WE produces less relevant generalizations using word similarities to the initial demonstration. RCSE's disadvantage compared with Ours stems from the lack of modeling action effects as well as other relationships that can help to rank more relevant generalizations higher. Crucially, Ours also outperforms TM, signaling that the embedding is inferring generalizations not possible with the training set alone. Lastly, we can consider the PN baseline; however, analyzing Table 5.2 hides the transient behavior of PN as it accumulates demonstrations. We provide the moving average over a window of 60 executions in Figure 5.2 to show how PN eventually approaches the performance of Ours, but only after PN accumulates on average 243 additional demonstrations from the oracle. PN highlights a limitation of Ours, the lack of adaptation to execution outcomes leading to a flat success rate. However, despite PN having the additional 243 demonstrations, Ours outperforms PN on average with a single demonstration because Ours learns priors over

Table 5.3: Ablation Success Rates in 6 E_x Types

E_x Type	Abl. L	Abl. O	Abl. OL	Abl. AO	Abl. AOL
L Gen. Only	86.4%	0.0%	86.4%	0.0%	86.4%
O Gen. Only	0.0%	71.6%	71.6%	82.2%	82.2%
OL Gen. Only	0.0%	0.0%	68.2%	0.0%	74.6%
AO Gen. Only	0.0%	0.0%	0.0%	80.2%	80.2%
AOL Gen. Only	0.0%	0.0%	0.0%	0.0%	67.4%
Random Gen.	6.2%	4.6%	24.4%	14.8%	71.8%

the semantic domain knowledge in VirtualHome.

Unlike our MRE, the MLN baseline implemented with pracmln [144] is not included in the above results because the scale of the learned KG (see Table 5.1) caused intractability issues. The complexity for the number of ground variables for an MLN is $\mathcal{O}(|L| * |C|^N)$ where $|L|$ is the number of predicates (relations - 11), $|L|$ is the number of constants (entities - 281), and N is the maximum arity of a predicate in symbol space (2) [13]. Despite simplifying the problem from 868,571 ground variables to 67,656, MC-SAT required $\sim$10 minutes to perform full-posterior inference for a single ground atom. These inferences occur many times during a single sample of the 12,000 sampled execution environments making the MLN baseline impractical for our use case.

5.4.2 Ablation Studies of Task Generalization Algorithm

We now provide multiple ablation studies of algorithm 1.

Characterizing algorithm 1 reasoning levels: We implemented 5 ablations that allowed combinations of generalizations at the spatial, object, or action reasoning levels. These 5 ablations implement at least one of the 5 types of generalizations presented in subsection 5.3.2 by toggling portions of algorithm 1 (i.e. O, L, OL, AO, AOL). In addition, we perturbed environments in a controlled manner to observe which components of our algorithm were suited to which environment perturbation types. The environment perturbation types corresponded to one of the 5 generalization cases in subsection 5.3.2 (e.g. for L, only perturb location of o_d) or random perturbations as in the prior evaluation of subsection 5.4.1. We reduced the number of initial demonstrations to 10 and execution

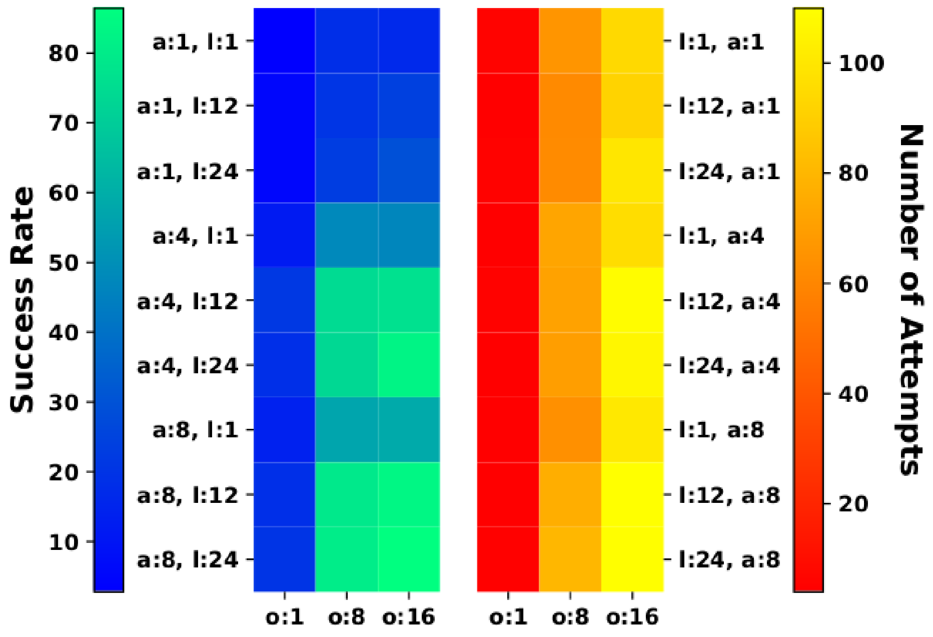


Figure 5.3: Ablation queue size tuning heatmap showing success rate vs number of attempts.

environments to 50 due to the large number of cases for each ablation and test environment combination. Our results, shown in Table 5.3, highlight the importance of combined reasoning at multiple levels to generalize the demonstrated task plan to larger number of execution environments.

Characterizing algorithm 1 queue sizes: We defined ablations by varying the queue sizes for each reasoning level in proportion to the total number of valid generalizations (i.e. $|l_{subs}| = \{1, 12, 24\}$, $|o_{subs}| = \{1, 8, 16\}$, $|a_{subs}| = \{1, 4, 8\}$), leading to a total of 27 different ablations. Random environment perturbations were used for testing, as in the prior evaluation of subsection 5.4.1. Due to the large number of ablations, we again reduced the number of initial demonstrations to 10 and execution environments to 50. We selected $\{|l_{subs}|, |o_{subs}|, |a_{subs}|\} = \{12, 8, 4\}$ from the results of our grid search to have moderate success rates while keeping the number of generalization attempts below 1% of the total possible (8,820). The results of our ablation study are shown in Figure 5.3. Each cell in Figure 5.3 has a unique combination of ($\mathbf{l}$, $\mathbf{o}$, $\mathbf{a}$) values corresponding to values of $(|l_{subs}|, |o_{subs}|, |a_{subs}|)$, respectively. For example, the top left cell corresponds to the ablation where $(|l_{subs}|, |o_{subs}|, |a_{subs}|) = (1, 1, 1)$. Overall, the different ablations indicated a trade-off for priority queue size; larger queue sizes lead to higher success rates but also higher numbers of generalization attempts.

5.4.3 Validation on a Robot Platform

Our last experiment was to validate our approach on a physical robot platform, Fetch [76], using a task execution stack that facilitates execution of primitive actions [77]. The task execution stack was used to make the Fetch execute task plans, receive environment percepts indicating failed primitive actions, and accept generalizations from the task generalization module. Some errors outside the scope of our work are handled by the task execution stack (e.g. base path-planning obstacles, precarious grasps); however, any errors not handled by the task execution stack or our approach are considered failures in our test cases, as all executions are autonomous without human intervention[1].

In our robot experiments, we used five different objects, namely “towel rolled”, “washing sponge”, “scrubber”, “feather duster” and “duster” (Shown in Figure 5.4), with four different actions “wipe”, “scrub”, “dust” and “sweep”. Possible locations for objects include, kitchen counters, coffee tables, sofa, desk, sink, drawers, shelf and kitchen table. The testing environment that emulates a small studio apartment is shown in Figure 5.4. As shown in the figure, the robot has to find an appropriate object in the environment to clean the designated location. We evaluated our approach on the robot by generating 10 random perturbations of one cleaning task demonstrated for each object, as described in subsection 5.4.1.

In our robot evaluations, the robot successfully generalized demonstrated tasks to 38 of 50 total execution environments (76% success rate). We note that none of the failures were due to our approach. The failure cases were due to manipulation (2), object detection (7), plane-segmentation (1), and grasping (2) robot errors. It is worth noting the MRE inferred three key triples not within the training set used to learn the embedding (i.e. (featherduster,

[1]Example executions: https://youtu.be/epRjleYDTCw

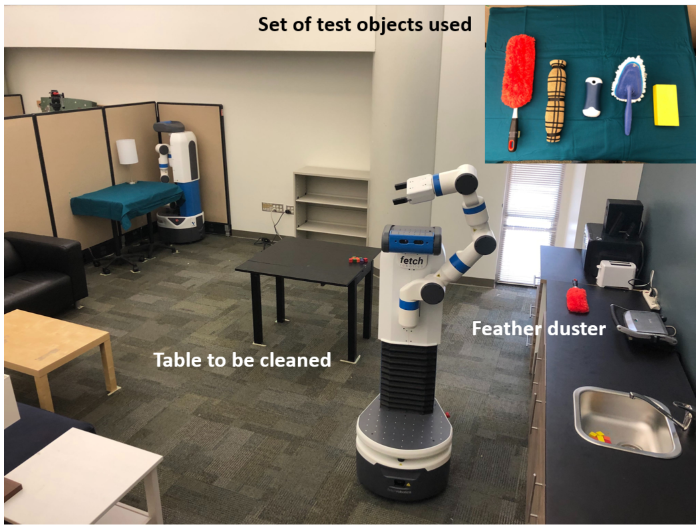


Figure 5.4: Sample test environment set up similar to a household setting. Test objects shown in top right, from left to right: feather duster, towel rolled, scrubber, duster, and washing sponge.

ObjOnLoc, shelf), (duster, ObjUsedTo, dust), (towelrolled, ObjUsedTo, wipe)) to generate 26.3% of the successful generalizations (10 of the 38). The robot's successful executions from inferred triples highlights the benefit of using a KG representation that can infer unseen facts given existing facts.

5.5 Discussion & Conclusion

Robust one-shot task execution continues to be a challenging problem for learning from demonstration. We introduced the task generalization module for generalization of task plans to new execution environments by integrating task plans with domain knowledge modeled by a MRE. We showed how the MRE can be learned from individual observations in simulation. We compared our approach to representative baselines in the context of one-shot task execution in simulation. Our experiments demonstrated that our knowledge representation infers more relevant generalizations on average, leading to higher success rates than the baselines. Lastly, we validated our work on a physical platform, showing generalization of the demonstrated tasks to 38/50 execution environments, including technical failures.

Our future work will explore how the domain knowledge can adapt to outcomes observed during execution, instead of solely relying on the prior distributions as is common in existing work. The lack of adaptability results in default robot behaviors that do not adapt despite receiving many new observations during execution failures. Incorporating adaptability into a MRE is challenging due catastrophic forgetting, as discussed in section 2.4. Therefore, an approach that seeks to make MREs more adaptable must include mechanisms to mitigate catastrophic forgetting.

In addition, our task generalization module presented allows a non-expert end user to program a robot task that generalizes to many execution environments but does not explain the reasoning behind the robot's decision making. We designed the robot's decision making to be informed by the MRE. The inferences provided by the MRE enable the robot to infer facts not present within the observed knowledge, enabling robust execution in more execution environments. However, the MRE has no mechanism to explain the reasoning that led to an erroneous inference causing strange robot generalization behavior. For example, wiping with a vacuum. The opaque nature of MREs limits transparency of the robot's decision making which can reduce the end users trust in the system's autonomous execution. Our future work will seek to address the lack of explainability of MREs by using XAI techniques discussed in section 2.5.

5.6 Findings & Contributions

This work comprises our contributions to **the integration of our KG representation into a robot architecture to improve robust task execution**. We did this by generalizing task plan constituents from a demonstrated task plan using domain knowledge modeled in our proposed KG representation. Our experiments showed that our KG representation generalized learned domain knowledge beyond the set of observed facts and outperformed word embedding and plan network baselines. Additionally, our experiments showed that MLNs had intractable inference time when representing the same domain knowledge used in our experiments. Our task generalization module demonstrates one way in which our KG representation can be used on a robot to inform task planning and execution. Our experiments highlighted two branches of future work to improve the proposed KG representation detailed in section 5.5. The next two chapters of this thesis will address whether MREs can be sequentially trained to update the represented KG and whether MRE inference can be explainable to end users.

CHAPTER 6
CONTINUAL KNOWLEDGE GRAPH EMBEDDING

In this chapter we present several methods to sequentially train a MRE under continual learning assumptions. We relax the assumptions from the KG representation presented in chapter 4 to enable the sequential training of the MRE on disjoint subsets of a KG. We posit that a robot using a MRE to represent domain knowledge can incrementally update the representation with new domain knowledge the robot obtains while not forgetting previous concepts. We validate our assumption by incorporating our contributed methods into the system from chapter 5 and evaluate the robot's task execution success rate in continual learning scenarios.

6.1 Introduction

Representing and reasoning about semantic knowledge is a key task in robotics. In recent years, there has been a resurgence in methods that use distributed (neural) representations, e.g., word and KG embeddings, for reason about semantic knowledge in the context of navigation [18], grounding [12], affordance modeling [143], success detection [146], manipulation [147], and instruction following [11]. While robots frequently observe previously unknown concepts, these embedding algorithms typically assume that all embedding concepts are known a priori, and incorporating new information requires all concepts to be learned afresh. In addition, in robotics applications, the limited availability of computational resources and storage, and concerns regarding storing sensitive information, can make batch learning with all observed data infeasible. We seek to relax this static assumption in KG embedding and enable adaptive revision of distributed representation of semantic knowledge for robots.

Towards achieving our objective, we draw on *Continual Learning*, the research area which focuses on the challenging problem of incrementally revising learned neural representations [46]. Existing continual learning methods have predominantly been applied to object recognition and include regularization [48, 49], architecture modification [50, 51], generative replay [52, 53], and a reformulation of regularization for KG embedding [54]. However, continual learning methods remain largely unexplored for KG embedding. Furthermore, the implications of any related assumptions for robotics is not well documented because existing methods focus on the final inference performance and define different task specific measures [56].

Our work makes three contributions. First, we reformulate and extend the underlying principles of five representative continual learning methods: (i) Progressive Neural Networks [50]; (ii) Copy Weight Re-Init [51]; (iii) L2 regularization [49]; (iv) Synaptic Intelligence [55]; and (v) Deep Generative Replay [52], and apply them to the *continual KG embedding* (CKGE) problem. Second, we introduce an empirically evaluated heuristic sampling strategy to generate CKGE datasets from KGs, since benchmark datasets do not exist for the CKGE problem. Third, we build on existing continual learning measures [57] to characterize the use of each reformulated method for robot tasks that leverage semantic knowledge.

We performed two sets of evaluations to characterize the effect of each CKGE method on MRE link-prediction and robust task execution success rate in different continual learning scenarios. For the MRE link-prediction evaluations, we consider two KG embedding representations with different assumptions and loss functions: TransE [29] and Analogy [30]; and three benchmark KGs (WN18RR, FB15K237 [150], and AI2Thor [143]). We evaluated each adapted method under unconstrained, data-constrained, and time-constrained settings by sampling disjoint subsets of a KG used in prior robotics work [143], containing actions, locations, objects, and other concepts. For the robot task execution evaluations, we use the TuckER embedding [32] and perform task execution in a household simulator [148]. We evaluated each adapted method under unconstrained, data-constrained, and time-constrained settings by sampling disjoints subsets of the unique environments available in the household simulator. Experimental results indicate that: (i) our generative replay approach outperforms other methods; (ii) there are interesting trade-offs between inference capability, learning speed, and memory usage that should be considered when choosing a CKGE method; and (iii) insights gained from exploring these trade-offs enable us to select a CKGE method that best matches the constraints of a given robotics application that models semantic knowledge.

6.2 Background

We begin by detailing background information about MRE and continual KG embedding. Modeling semantic knowledge in robotics is often achieved using an explicit model of world semantics in the form of a KG $\mathcal{G}$ composed of individual facts or triples (h, r, t); h and t are the head and tail entities (respectively) for which the relation r holds, e.g., (*cup, hasAction, fill*) [8, 4, 5, 6]. Recent work has modeled $\mathcal{G}$ using distributed representations because of their ability to approximate proximity of meaning from vector computations [18, 12, 143, 146, 147, 11].

Multi-relational (KG) embeddings are distributed representations that model $\mathcal{G}$ in vector space [27], learning a continuous vector representation from a dataset of triples $\mathcal{D} = \big\{(h,r,t)_i, y_i |\, h_i, t_i \in \mathcal{E}, r_i \in \mathcal{R}, y_i \in \{0,1\}\big\}$, with $i \in \{1...|\mathcal{D}|\}$. Here y_i denotes whether relation $r_i \in \mathcal{R}$ holds between $h_i, t_i \in \mathcal{E}$. Each entity $e \in \mathcal{E}$ is encoded as a vector $\mathbf{v}_e \in \mathbb{R}^{d_\mathcal{E}}$, and each relation $r \in \mathcal{R}$ is encoded as a mapping between vectors $\mathbf{W}_r \in \mathbb{R}^{d_\mathcal{R}}$, where $d_\mathcal{E}$ and $d_\mathcal{R}$ are the dimensions of vectors and mappings respectively [27, 28]. The embeddings for $\mathcal{E}$ and $\mathcal{R}$ are typically learned using a scoring function $f(h, r, t)$ that assigns higher (lower) values to positive (negative) triples [28]. The learning objective is thus to find a set of embeddings $\Theta = \big\{\{\mathbf{v}_e |\, e \in \mathcal{E}\}, \{\mathbf{W}_r |\, r \in \mathcal{R}\}\big\}$ that minimizes the loss $\mathcal{L}_\mathcal{D}$ over $\mathcal{D}$. Loss $\mathcal{L}_\mathcal{D}$ can take many forms depending on the MRE representation used, e.g., Margin-Ranking Loss [29] or Negative Log-Likelihood Loss [30]. However, all entities and relations are assumed to be known before training [27, 28], which may be infeasible for robots observing new concepts or new facts about existing concepts.

Of the continual learning scenarios that exist, we chose *Incremental Class Learning* (ICL) because it best matches the assumptions of robot systems representing semantic knowledge, with the distribution of input data and target labels changing across learning sessions as the robot incrementally observes disjoint sets of new facts about new and existing concepts. In CKGE, the dataset $\mathcal{D}$ of a KG $\mathcal{G}$ is split into multiple datasets $\mathcal{D}^n$ where n indicates the learning session [54]. Each $\mathcal{D}^n$ contains a disjoint set of all triples of a subset of entities and relations. For a robot observing new facts, the size of the set of observed entities, relations, and triples grows, (e.g., $|\mathcal{E}^n| \leq |\mathcal{E}^{n+1}|$), and the embedding must consider new facts and concepts in each learning session. In such a learning scenario, the objective is to find a set of embeddings $\Theta^n = \big\{\{\mathbf{v}_e^n | \, e \in \mathcal{E}^n\}, \{\mathbf{W}_r^n | \, r \in \mathcal{R}^n\}\big\}$ that minimize the loss $\mathcal{L}_{\mathcal{D}^n}$ over the dataset for all time steps. Of the range of continual learning methods, only L2-regularization has been applied to CKGE [54]; more sophisticated methods that have shown promise in other domains, e.g., generative replay, remain unexplored. Also, important measures for robotics, such as learning efficiency and model complexity, are not well documented for representative techniques [57], making it difficult to evaluate the suitability of these methods for modeling semantic knowledge in robotics. Our work is designed to fill these gaps.

6.3 Approaches to Continual Knowledge Graph Embedding

We seek to characterize the use of continual learning methods for KG embedding in robotics by exploring the associated assumptions and trade-offs. In this section, we describe how we reformulate and extend the principles of five carefully selected representative continual learning techniques to develop *continual KG embedding* (CKGE) methods. These methods were designed for traditional neural networks and required varying levels of innovation to support KG embeddings. In each case, we carefully considered the suitability of its principles to support the desired capabilities and assumptions of KG embeddings.

6.3.1 Architectural Modification Methods

Among the methods that modify the architecture of a neural network to accommodate new training data while minimizing performance losses over older data, we adapted two methods for KG embeddings.

Progressive Neural Networks (PNN) [50] add copies of existing layers of a multi-layered neural network for each new learning session. When a new learning session begins, existing weights are frozen so that back-propagated gradients do not affect the performance over data from previous sessions. Also, lateral connections are made between successive layer copies to enable the forward transfer of previously learned weights. To make PNNs applicable to KG embedding, we first expand the embedding matrices $\mathbf{v}^n \in \mathbb{R}^{|\mathcal{E}^n| \times d_{\mathcal{E}}}$ and $\mathbf{W}^n \in \mathbb{R}^{|\mathcal{R}^n| \times d_{\mathcal{R}}}$ to include new entities and relations in the learning session n. Second, we freeze embeddings for entities and relations encountered in prior learning sessions to prevent their corruption in the current learning session. Instead of creating separate copies of these embedding matrices for each learning session, we only expand the existing matrices to promote forward transfer of prior embeddings in new learning sessions.

Copy Weight with ReInit (CWR) [51] maintains the weights of the final layer of the network during a new learning session (i.e. temporary weights, TW), separate from the corresponding weights trained in prior learning sessions (i.e. consolidated weights, CW) to avoid corruption. Other than the two sets of final layer weights considered during (continual) learning, the weights of other layers are frozen and shared across learning sessions. TW are re-sized and re-initialized in each learning session according to the number of classes being trained. After each learning session, the TW for new classes are copied over to CW, which acts as a memory buffer separate from the network. If a previously trained class is encountered, relevant entries in TW are averaged with those in CW. Training for the subsequent session begins by re-sizing and re-initializing TW.

To apply the principles of CWR to KG embedding, we first introduce two sets of embeddings: consolidated embeddings (CE) $\{\mathbf{v}^n_{ce},\mathbf{W}^n_{ce}\}$ and temporary embeddings (TE) $\{\mathbf{v}^n_{te},\mathbf{W}^n_{te}\}$. Second, for each learning session, we resize and re-initialize the TE for entities $\mathbf{v}^n_{te}$ and relations $\mathbf{W}^n_{te}$ based on the number of entities and relations (respectively) in the session. After the session, we move TE into CE by copying new embeddings or averaging existing ones. As a result, the number of CE increases monotonically in each learning session with the number of observed entities $\mathcal{E}^n$ and relations $\mathcal{R}^n$ so that $\mathbf{v}^n_{ce} \in \mathbb{R}^{|\mathcal{E}^n| \times d_{\mathcal{E}}}$ and $\mathbf{W}^n_{ce} \in \mathbb{R}^{|\mathcal{R}^n| \times d_{\mathcal{R}}}$) (respectively); the number of TE changes in each learning session according to the number of entities and relations in that learning session's dataset $\mathcal{D}^n$.

6.3.2 Regularization Methods

Freezing previously learned weights prevents their corruption in subsequent sessions, but also prevents shared weights from being revised to better accommodate new concepts. Some continual learning methods allow adjustments to shared weights that perform well for prior and new sessions; they do so by enforcing some regularization terms in new learning sessions. We reformulate two such approaches for KG embeddings.

L2 Regularization (L2R) [49, 54, 47] is adapted in our approach by adding a regularization term to the learning session loss $\mathcal{L}_{\mathcal{D}^n}$, encouraging the trained weights to not deviate from their previous values:

$$\mathcal{L}_{\mathcal{D}^n} + \lambda \cdot \Big(||\mathbf{v}^n_e - \mathbf{v}^{n-1}||^2_2 + ||\mathbf{W}^n_r - \mathbf{W}^{n-1}||^2_2 \Big) \tag{6.1}$$

where $e \in \mathcal{E}^{n-1}$, $r \in \mathcal{R}^{n-1}$, and λ is a regularization scaling term tuned as a hyper-parameter. L2R can be rather strict because it penalizes all dimensions of an embedding equally, whereas a subset of the embedding dimensions often contribute more to loss or predictive abilities than others.

Synaptic Intelligence (SI) [55] extends L2R by considering the weight-specific contributions to the reduction in loss over a learning session. These contributions are quantified by summing the gradients that each weight adjustment contributes to the loss and using the total loss reduction as a normalizer. SI is generic enough to apply to KG embeddings with minimal changes because it is formulated in terms of the weight and loss trajectories. Equation 6.2 defines our implementation of SI for KG embedding, re-using terms from [55]:

$$\mathcal{L}_{\mathcal{D}^n} + \lambda \cdot \left(||\Omega_e(\mathbf{v}_e^n - \mathbf{v}^{n-1})||_2^2 + ||\Omega_r(\mathbf{W}_r^n - \mathbf{W}^{n-1})||_2^2 \right) \tag{6.2}$$

where $e \in \mathcal{E}^{n-1}$, $r \in \mathcal{R}^{n-1}$, Ω is the parameter regularization strength [55], and λ is a regularization scaling term tuned as a hyper-parameter for a particular representation.

We also considered but did not adapt Elastic Weight Consolidation (EWC) [49] for KG embedding. EWC performs regularization using the Fisher Information matrix to estimate how predictive particular weights are. However, an assumption of the Fisher Information matrix is that the scoring function which describes the relationship between weights and the data they are being trained on has a probabilistic interpretation. The assumption of a probability density function for the Fisher Information matrix does not allow EWC to be directly applied to MRE representations as they often use loss functions that do not have probabilistic interpretations, e.g., those using Margin-Ranking Loss.

6.3.3 Generative Replay Methods

Instead of maintaining model weights across learning sessions, generative replay methods learn generative models of the distribution of training data from previous learning sessions. Then, batch learning is approximated by sampling from the learned distribution and the training data from the current learning session. We reformulate one such method for KG embeddings.

Deep Generative Replay (DGR) [52, 53] is a continual learning method that uses a generative model $\mathbf{G}$ to approximate the distribution of all observed training examples (i.e. $\mathcal{D}$), and trains a discriminative model (i.e., solver) to perform a task. In the initial learning session, generator $\mathbf{G}^0$ and solver are trained using examples in $\mathcal{D}^0$. In any subsequent learning session i, a new generator $\mathbf{G}^i$ and solver are trained using examples in $\mathcal{D}^i$ and

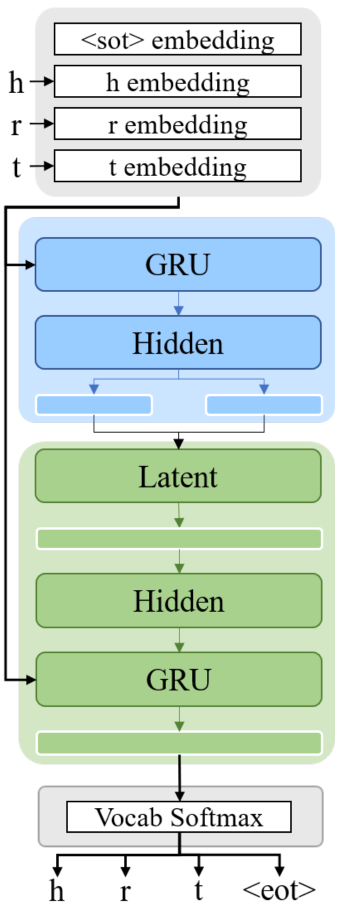


Figure 6.1: DGR architecture. Layers with white outline are linear.

samples from $\mathbf{G}^{i-1}$ that approximate $\mathcal{D}^{i-1}$, thus approximating training with $\mathcal{D}^{i-1} \cup \mathcal{D}^{i}$.

The challenge in applying the principles of DGR to KG embeddings is designing an effective generator, as the solver is determined by the representation used, i.e., $\Theta = \big\{\{\mathbf{v}_e | \, e \in \mathcal{E}\}, \{\mathbf{W}_r | \, r \in \mathcal{R}\}\big\})$. Sampling training examples is a known problem in KG embedding. Prior work has used Generative Adversarial Neural Networks to generate negative examples [151, 152, 153], but we cannot use these methods because their generators require positive examples as input. Prior work has shown that a Variational Auto-Encoder (VAE) can be used to sample sequences of discrete tokens [154]. We treat each triple as a sequence of discrete tokens to design our VAE-based generator.

Figure 6.1 shows our VAE architecture that uses Gated-Recurrent Units to encode and decode the triples to and from the latent space z. Input triples (h, r, t) to the encoder are first transformed into token embedding sequences $x = (\nu_h, \nu_r, \nu_t)$, where $\nu \in \mathbb{R}^{|\mathcal{E}^n| + |\mathcal{R}^n| \times d_{\mathcal{V}}}$

is a token embedding learned by the encoder with dimensionality $d_{\mathcal{V}}$. The encoder, shown in blue in Figure 6.1, is a learned posterior recognition model $q(z|x)$ that approximates the posterior distribution over z, conditioned on the input triple sequences x. Unlike a standard auto-encoder, the encoder is encouraged to keep the learned posterior $q(z|x)$ close to the prior over the latent space $p(z)$, which is a standard Gaussian. A similarity constraint based on the KL divergence measure in the objective function allows samples to be generated from the latent space. These samples are decoded using the decoder, shown in green in Figure 6.1, to maximize $p(x|z)$, the likelihood of a triple sequence x conditioned on its encoded latent space vector z, as in a standard auto-encoder. The output sequences of the VAE are transformed back into a triples using a Softmax function over all tokens (i.e., $e \in \mathcal{E}^n$ and $r \in \mathcal{R}^n$). The objective function for the VAE architecture is:

$$-\text{KL}\big(q(z|x)||p(z)\big) \cdot \alpha(\text{epoch}) + \mathbb{E}_{q(z|x)}\big[\log p(x|z)\big] \tag{6.3}$$

where an additional term $\alpha(\cdot)$ is included to anneal the Kullback–Leibler divergence loss, preventing issues such as vanishing gradients caused by posterior sampling and Kullback–Leibler divergence loss terms being driven to zero [154]. $\alpha(\cdot)$ is a function of the number of epochs trained for the learning session:

$$\alpha(\text{epoch}) = \frac{\lambda_{am}}{1 + e^{-\lambda_{as}\big(\text{epoch} - \lambda_{ap}\big)}} \tag{6.4}$$

where λ_{am}, λ_{as}, and λ_{ap} are hyper-parameters tuned during training to control the maximum value, slope, and position of the annealing function, respectively.

6.4 Experimental Setup for Link-Prediction Evaluations

We evaluate our CKGE methods in regards to link-prediction on two MRE representations: TransE [29] and Analogy [30]; and three benchmark KGs: AI2Thor [143], FB15K237 [150], and WN18RR [150]. The last two KGs are challenging and have been widely used in the graph embedding literature [150, 152, 155]. AI2Thor contains relations and entities related to service robotics, e.g., locations of objects, actions that can be performed on objects, and the outcomes that result from these actions [143]. We report the accuracy and complexity of each method based on seven performance measures chosen from prior continual learning work in robotics [57]. In each trial, the evaluation task is triplet prediction, a fundamental KG embedding task [18, 143] with a well-defined experimental setup [29, 27] as described later in this section.

CKGE datasets: Since there is no established benchmark dataset for CKGE, we introduce three standard evaluation datasets that we obtain by sampling. Our heuristic sampling strategy emulates the New Instances and Concepts scenario presented in [57] under the categorization of the nature of data samples within training sets. Therefore, our sampling strategy models the scenario where a robot explores a world and discovers new triple instances that contain new concepts (i.e. entities or relations), new triple instances that contain previously observed concepts, and triple instances that have been previously observed. Consider a KG $\mathcal{G}$ whose triples $\mathcal{D}$ have been split into a training set $\mathcal{D}_{Tr}$, validation set $\mathcal{D}_{Va}$, and test set $\mathcal{D}_{Te}$. Our approach for generating datasets for $n = \{1, ..., N\}$ learning sessions is:

1. *Sample training triples*: uniformly sample without replacement $\frac{|\mathcal{D}_{Tr}|}{N}$ triples from training set $\mathcal{D}_{Tr}$ of $\mathcal{G}$. These triples form training dataset $\mathcal{D}^n_{Tr}$.

2. *Extract entities and relations*: create a set of entities E^n and a set of relations R^n for this session from the triples in $\mathcal{D}^n_{Tr}$. The set of all observed entities (relations), i.e., $\mathcal{E}^n$ ($\mathcal{R}^n$) is the union of current and prior E^n (R^n).

3. *Construct n^{th} validation and test sets*: extract from $\mathcal{D}_{Va}$ and $\mathcal{D}_{Te}$ the triples whose head, relation, and tail belong to E^n and R^n (respectively). These triples form validation set $\mathcal{D}^n_{Va}$ and test set $\mathcal{D}^n_{Te}$ of the n^{th} session.

4. *Remove sampled training triples*: remove $\mathcal{D}^n_{Tr}$ from $\mathcal{D}_{Tr}$ of $\mathcal{G}$.

Table 6.1: CKGE Datasets; Benchmarks

	WN18RR-5-LS				
	LS-1	LS-2	LS-3	LS-4	LS-5
$\lvert E^n \rvert$	20,368/(50%)	20,389/(73%)	20,249/(87%)	20,463/(95%)	20,437/(99%)
$\lvert R^n \rvert$	11/(100%)	11/(100%)	11/(100%)	11/(100%)	11/(100%)
$\lvert \mathcal{D}^n_{Tr} \rvert$	17,367/(20%)	17,367/(40%)	17,367/(60%)	17,367/(80%)	17,367/(100%)
$\lvert \mathcal{D}^n_{Va} \rvert$	1,117/(37%)	1,141/(57%)	1,187/(71%)	1,190/(80%)	1,184/(86%)
$\lvert \mathcal{D}^n_{Te} \rvert$	1,168/(37%)	1,159/(57%)	1,218/(72%)	1,173/(81%)	1,175/(87%)
	FB15K237-5-LS				
	LS-1	LS-2	LS-3	LS-4	LS-5
$\lvert E^n \rvert$	13,143/(90%)	13,106/(96%)	13,115/(98%)	13,089/(99%)	13,163/(100%)
$\lvert R^n \rvert$	237/(100%)	237/(100%)	237/(100%)	237/(100%)	237/(100%)
$\lvert \mathcal{D}^n_{Tr} \rvert$	54,423/(20%)	54,423/(40%)	54,423/(60%)	54,423/(80%)	54,423/(100%)
$\lvert \mathcal{D}^n_{Va} \rvert$	17,013/(97%)	16,929/(99%)	16,917/(100%)	16,882/(100%)	16,905/(100%)
$\lvert \mathcal{D}^n_{Te} \rvert$	19,776/(97%)	19,727/(99%)	19,734/(99%)	19,758/(100%)	19,801/(100%)

Table 6.2: CKGE Datasets; Robotics

	LS-1	LS-2	LS-3	LS-4	LS-5
$\lvert E^n \rvert$	199/(87%)	202/(96%)	191/(97%)	193/(98%)	198/(100%)
$\lvert R^n \rvert$	15/(100%)	14/(100%)	15/(100%)	15/(100%)	15/(100%)
$\lvert \mathcal{D}^n_{Tr} \rvert$	405/(20%)	405/(40%)	405/(60%)	405/(80%)	405/(100%)
$\lvert \mathcal{D}^n_{Va} \rvert$	224/(91%)	231/(98%)	229/(99%)	223/(99%)	231/(100%)
$\lvert \mathcal{D}^n_{Te} \rvert$	224/(91%)	225/(98%)	214/(99%)	213/(99%)	226/(100%)

5. Repeat steps 1-4 until no training triples exist in $\mathcal{G}$ or a predefined number of iterations are completed.

We generated three CKGE datasets with $n = 5$ sessions using our approach on two established benchmark KGs in the graph embedding community (WN18RR and FB15K237 [150]) and a KG used in robotics (AI2Thor [143]). Table 6.1 and Table 6.2 report statistics of each dataset. The columns of the tables denote the learning session (LS-X, X$\in$ $[1, 5]$), while rows correspond to the statistics, e.g., $|E^n|$ is the size of the entity set. Individual cells indicate the value, with coverage with respect to the original KG shown in parentheses. For instance, in LS-2 of WN18RR-5-LS, there are $20,389$ entities and 73% of all entities in WN18RR have been observed. Note that our sampling strategy empirically produces datasets with better coverage and higher percentages of new training triples each learning session, i.e., more challenging datasets for CKGE, than previous methods such as entity sampling [54]. Furthermore, our sampling strategy makes the distribution of the n training sets more closely match the original $\mathcal{D}_{Tr}$ than entity sampling by ensuring sampling without replacement $\left(\mathcal{D}^n_{Tr} \bigcap \mathcal{D}^{n+1}_{Tr} = \emptyset \,\forall\, n\right)$. In addition to our chosen sampling strategy, we repeated all experiments with two other sampling strategies, entity and relation sampling, detailed in Appendix A.

Evaluation procedure: The evaluation task is to predict complete triplets from incomplete ones in test splits $\mathcal{D}^n_{Te}$, i.e., predict h given (r, t) or t given (h, r). To perform triplet prediction, each test triplet (h, r, t) is first corrupted by replacing the head (or tail) entity with every other possible entity in the current session $\mathcal{E}^n$. Then, to avoid underestimating the embedding performance, we remove all corrupted test triplets that still represent a valid relationship between the corresponding entities, known as the "filtered" setting in the literature [29]. Last, scores are computed for each test triplet and its (remaining) corrupted triplets using the scoring function $f(h, r, t)$ (defined below), then ranked in descending order.

Recall that we consider two KG embedding representations to show the generality of our methods: TransE and Analogy. TransE represents relationships as translations between entities, i.e., $\mathbf{v}_h + \mathbf{W}_r = \mathbf{v}_t$ [29]. It uses the scoring and margin ranking loss functions in Equation 6.5 and Equation 6.6, where $[x]_+ = max(0, x)$, γ is the margin, and (h', r, t') are corrupted triples in a corrupted KG $\mathcal{G}'$. Embeddings are subject to normalization constraints (i.e. $||\mathbf{v}_e||_2 \leq 1 \,\forall\, e \in \mathcal{E}$ and $||\mathbf{W}_r||_2 \leq 1 \,\forall\, r \in \mathcal{R}$) to prevent trivial minimization of $\mathcal{L}$ by increasing entity embedding norms during training.

$$f(h, r, t) = ||\mathbf{v}_h + \mathbf{W}_r - \mathbf{v}_t||_1 \tag{6.5}$$

$$\mathcal{L} = \sum_{\substack{(h,r,t)\in\mathcal{G},\\ (h',r,t')\in\mathcal{G}'}} [f(h, r, t) + \gamma - f(h', r, t')]_+ \tag{6.6}$$

Analogy, on the other hand, represents relationships as (bi)linear mappings between entities, i.e., $\mathbf{v}_h^\top \mathbf{W}_r = \mathbf{v}_t^\top$ [30]. It uses the scoring and negative log loss functions in Equation 6.7 and Equation 6.8 where σ is a sigmoid function, y is a label indicating whether the triple is corrupted, and $\mathcal{G}'$ is the corrupted KG. Additionally, the linear mappings (i.e. relations) are constrained to form a commuting family of normal mappings, i.e., $\mathbf{W}_r\mathbf{W}_r^\top = \mathbf{W}_r^\top\mathbf{W}_r \,\forall\, r \in \mathcal{R}$ and $\mathbf{W}_r\mathbf{W}_{r'} = \mathbf{W}_{r'}\mathbf{W}_r \,\forall\, r, r' \in \mathcal{R}$, to promote analogical structure within the embedding space.

$$f(h, r, t) = \langle \mathbf{v}_h^\top \mathbf{W}_r, \mathbf{v}_t \rangle \tag{6.7}$$

$$\mathcal{L} = \sum_{(h,r,t,y)\in\mathcal{G},\mathcal{G}'} -\log\sigma(y \cdot f(h, r, t)) \tag{6.8}$$

Evaluation measures: We build on existing measures to characterize each of our CKGE methods. We consider different factors important for robotics applications modeling semantic knowledge, e.g., inference, memory usage, and learning efficiency. In addition to the only measure provided in prior CKGE work [54] (i.e. inference performance), we report seven other robotics-oriented metrics cataloged in [57] that measure unique aspects of continual learning algorithms. Specifically, for inference performance, we consider the *mean reciprocal rank* of correct triplets (MRR) and the proportion of the correct triplets ranked in the top 10 (Hits@10). During each learning session, we compute the evaluation measures for the test sets of all learning sessions to characterize the effect of learning on prior, current, and future learning sessions. During the n^{th} learning session of N total sessions, the two training-test inference performance matrices $\mathbf{M} \in \mathbb{R}^{N\times N}$ (for MRR and Hits@10) are used to compute four measures that summarize accuracy and forgetting across learning sessions: (i) Average accuracy (ACC)Average accuracy (ACC) measures the average accuracy across learning sessions—Equation 6.9; (ii) Forward Transfer (FWT)Forward Transfer (FWT) measures zero-shot learning in future sessions by transferring weights learned in prior session(s)—Equation 6.10; (iii) Backwards Transfer (BWT)Backwards Transfer (+BWT) measures the improvement over expected performance of a prior learning session as a result of learning in future sessions—Equation 6.12; and (iv) Remembering (REM) measures how performance in a learning session degrades as a result of learning in subsequent sessions—Equation 6.13.

$$\text{ACC} = \frac{\sum_{i \geq j}^{N} \mathbf{M}_{i,j}}{\frac{N(N+1)}{2}} \qquad (6.9) \qquad\qquad \text{FWT} = \frac{\sum_{i<j}^{N} \mathbf{M}_{i,j}}{\frac{N(N-1)}{2}} \qquad (6.10)$$

$$\text{BWT} = \frac{\sum_{i=2}^{N} \sum_{j=1}^{i-1} (\mathbf{M}_{i,j} - \mathbf{M}_{j,j})}{\frac{N(N-1)}{2}} \qquad (6.11)$$

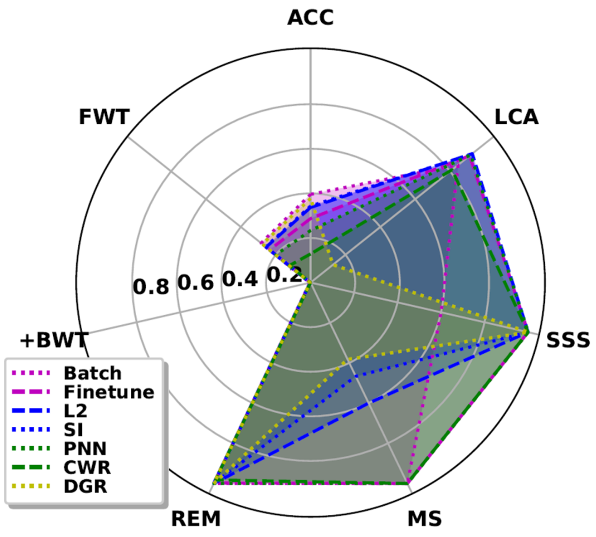
$$\text{+BWT} = \max(0, \text{BWT}) \qquad (6.12) \qquad\qquad \text{REM} = 1 - |\min(0, \text{BWT})| \qquad (6.13)$$

Other measures important for robotics applications that leverage semantics are space complexity and learning speed [57]. We capture space complexity for each CKGE method using Model Size (MS) and Samples Storage Size (SSS) measures [57]. MS measures the growth in memory usage $\mathcal{U}$ for model parameters θ across learning sessions for a particular method—Equation 6.14. SSS measures the growth in memory usage $\mathcal{U}$ for stored samples SS across learning sessions as a proportion of the total number of training samples for the task, i.e., $\mathcal{D}_{Tr}$, in Equation 6.15. For learning speed, we use the Learning Curve Area (LCA) measure [57], which we modify to range between zero and one (like other measures). For a performance measure m, it computes the area covered by the learning curve of the learning method up to the best measured performance m^* at time t as a proportion of the area achieved by perfect zero-shot learning (Equation 6.16).

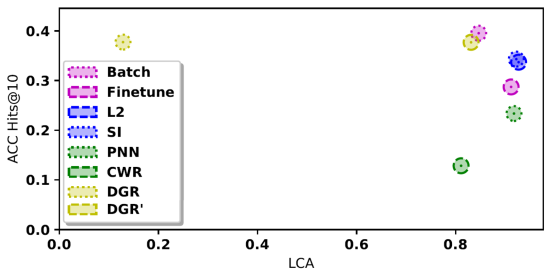
$$\text{MS} = \min(1, \frac{\sum_{i=1}^{N} \frac{\mathcal{U}(\theta_1)}{\mathcal{U}(\theta_i)}}{N}) \qquad (6.14)$$

$$\text{SSS} = 1 - \min(1, \frac{\sum_{i=1}^{N} \frac{\mathcal{U}(SS_i)}{\mathcal{U}(\mathcal{D}_{Tr})}}{N}) \qquad (6.15)$$

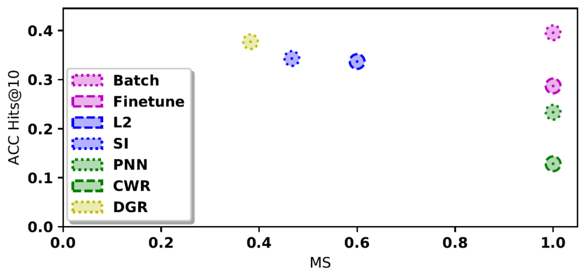
$$\text{LCA} = \frac{\int_0^t m \, dm}{m^* \times t} \qquad (6.16)$$



(a)



(b)



(c)

Figure 6.2: Measures averaged for all datasets in Table 6.1 and graph embedding representations in section 6.4. Hits@10 used for ACC, FWT, +BWT, and REM. Best viewed in color.

Software implementation: Please see supplementary material[1] for details about the tuning of hyper-parameters of CKGE methods, each KG embedding representation used for evaluation, and evaluation datasets, experiments, and results that are omitted here for brevity.

6.5 Experimental Results for Link-Prediction Evaluations

Results reported in this section are the average of five test runs in each experimental scenario; statistical significance is tested using repeated-measures ANOVA and a post-hoc Tukey's test. Any mention of 'significance' implies statistical significance at 95% significance level (i.e. $p < 0.05$).

[1]https://github.com/adaruna3/continual-kge

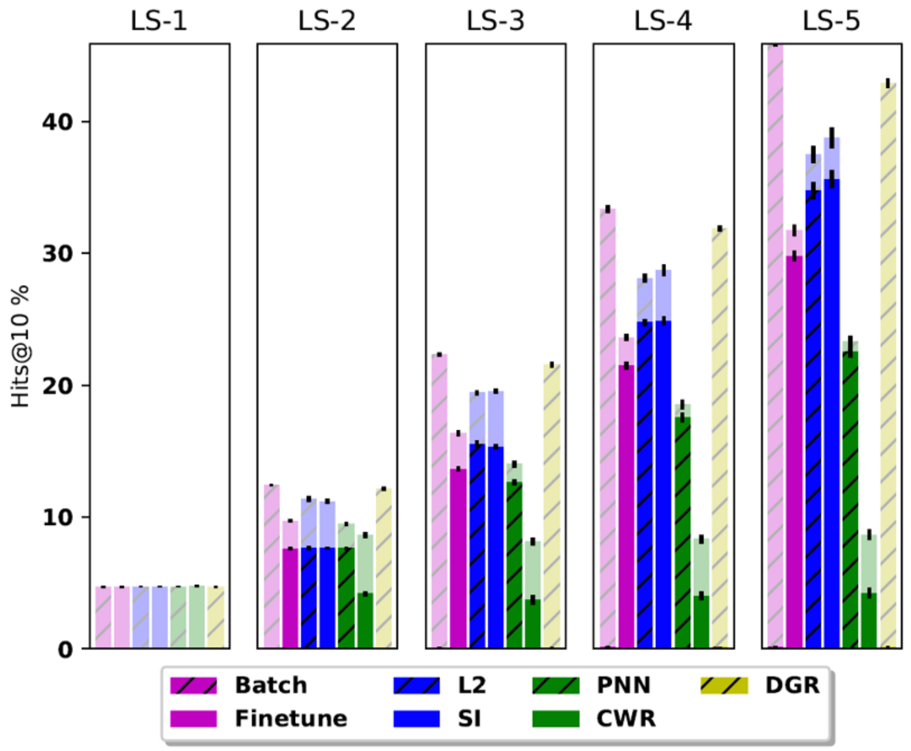
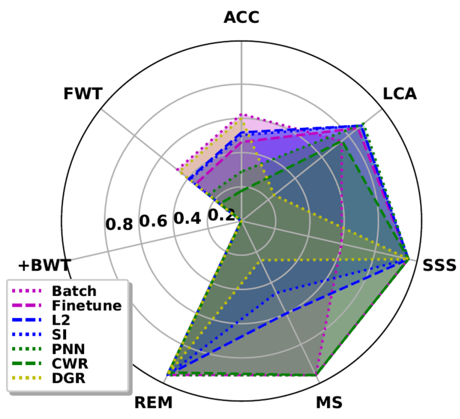


Figure 6.3: Hits@10 from initial (bright) to final (transparent) epoch. Black errors bars indicate standard deviation. L2R and SI perform better than DGR in the initial epoch, but DGR outperforms in the final epoch after the first learning session. Best viewed in color.

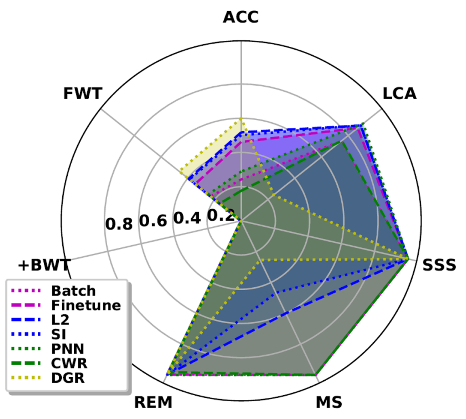
In addition to the CKGE methods, we considered two additional methods that served as upper and lower bounds (i.e., baselines) for the expected inference performance of the CKGE methods. *Batch* represents the inference upper bound because it can store all prior examples to train a new embedding in each learning session. *Finetune* represents the lower bound because it fine-tunes the embedding with examples only from the current learning session and has no means to prevent catastrophic forgetting.

6.5.1 Benchmark Datasets:

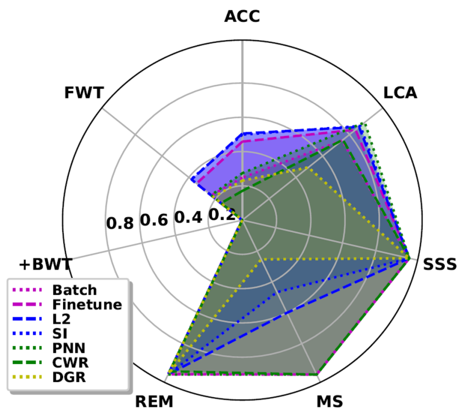
Figure 6.2a summarizes the results of experiments using benchmark KG datasets of Table 6.1 (WN18RR, FB15K237), where the range of each measure is $[0, 1]$ and larger values are better. Although DGR significantly outperforms other methods in terms of inference (i.e., using ACC and FWT), there are insights and trade-offs to consider based on other factors.



(a)



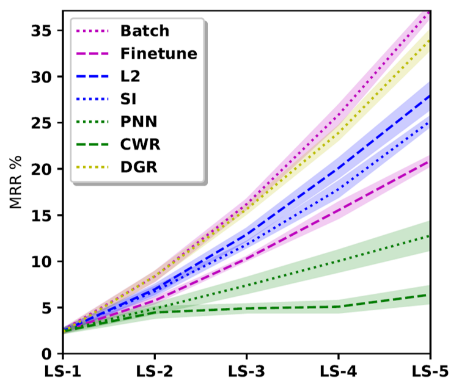
(b)



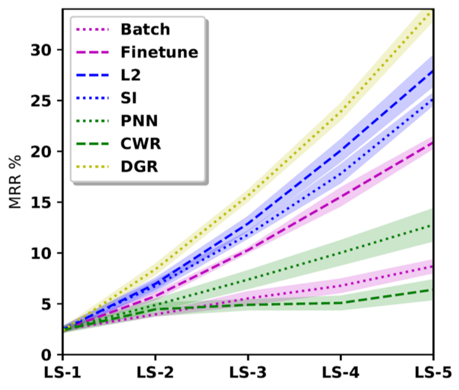
(c)

Figure 6.4: Semantics-driven robotics application scenarios: (a) Unconstrained; (b) Data Constrained; and (c) Time and Data Constrained. Best viewed in color.

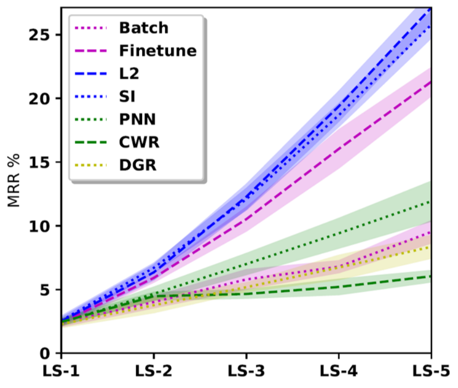
- Figure 6.2b shows that DGR has a significantly lower learning speed (based on LCA) than the other methods since a new generative model must be trained in each learning session. If the number of epochs to train the generative model are ignored, DGR's LCA is comparable to Batch (DGR′ in Figure 6.2b) but still significantly lower than the regularization techniques (L2R and SI).
- Figure 6.2c indicates that methods with good inference performance also tend to have higher model memory growth (i.e., MS measure); among the methods with significantly better inference performance than Finetune, L2R has the smallest MS followed by SI and DGR.
- Since they regularize prior embeddings, L2R and SI initially perform better than DGR, as does the Finetune baseline, as seen in the Hits@10 plots for each method at the start



(a)



(b)



(c)

Figure 6.5: Semantics-driven robotics application scenarios: (a) Unconstrained; (b) Data Constrained; and (c) Time and Data Constrained. In each line plot, shading indicates standard deviation. Best viewed in color.

and end of each learning session (Figure 6.3).

- Figure 6.2 and Figure 6.3 indicate that the CKGE methods based on architecture modification (i.e., PNN and CWR) have significantly lower inference performance than Finetune in all experiments. The difference in performance between PNN and the regularization-based methods shows the importance of flexibility over prior concepts for CKGE. Also, CWR's poor inference performance highlights the challenges of directly manipulating the embedding space because, although CWR can learn TE well in isolation, CE is quickly corrupted by the averaging performed to merge embeddings.

6.5.2 Robotics Datasets:

We constructed three evaluation scenarios using the AI2Thor KG dataset in Table 6.2. Each scenario corresponds to a different class of semantics-driven robotics applications. The first scenario, *Unconstrained* in Figure 6.4a and Figure 6.5a, corresponds to a robot that has access to all prior training examples at training time. More generally, the unconstrained scenario could represent robots with ready access to cloud services for data storage and processing. However, such a scenario may be unfeasible in some applications due to hardware constraints or security concerns. Our second scenario, *Data Constrained* in Figure 6.4b and Figure 6.5b, represents robots with access to limited training examples, e.g., only from the current learning session ($\mathcal{D}^n_{Tr}$); data constraints could be due to storage constraints or dynamic domain changes. The final scenario, *Time and Data Constrained* in Figure 6.4c and Figure 6.5c, mimics a mobile robot (or drone) operating under resource constraints; the robot only has access to training examples for the current learning session and has limited time to update the KG embedding. For simplicity, we limited the number of training epochs in each learning session to 100. The ranges of each measure are in $[0, 1]$ and larger values are better. The results from each scenario provide key insights about the choice of the CKGE method:

- In an *unconstrained scenario* (Figure 6.4a and Figure 6.5a), such as one in which a robot might have access to a cloud compute service, Batch learning is the best choice despite its significantly lower sample efficiency (SSS) and learning speed (LCA) because it provides significantly higher ACC and FWT compared with other methods.
- In a *data constrained scenario* (Figure 6.4b and Figure 6.5b), e.g., the robot can only update its semantic representation intermittently using limited on-board hardware. Batch's inference performance collapses because prior observations are unavailable. Given these constraints, DGR is the best choice, with much better ACC and FWT than other methods because it approximates Batch in the unconstrained scenario. However, DGR incurs a significant computational cost to train the generative model, resulting in a significantly

lower LCA value.

- In a *data and time constrained scenario* (Figure 6.4c and Figure 6.5c), e.g., the robot is updating its own semantic model on-board *during* a task, DGR is a poor choice because there is not enough time to sufficiently train the generative model. L2R and SI are better choices; SI with Analogy and L2R with either graph embedding offer significantly better inference performance than Finetune and significantly better LCA than Batch. Compared with SI, L2R's memory growth (MS) is significantly lower.

6.6 Experimental Setup for Robust Task Execution

For task execution, we evaluate our CKGE methods with the TuckER embedding [32] in the VirtualHome simulator using the robust task execution approach presented in chapter 5. The VirtualHome simulator offers multiple home simulation environments in which we can simulate task execution under continual learning assumptions with a virtual agent as in chapter 5. In these evaluations, we simulate a continual learning scenario in which a robot is operating in different environments (i.e. homes). We model a continual learning scenario by sequentially training the robot on datasets extracted from each unique environment in the simulator and evaluating the robot's task execution success rate on all previously encountered environments. We report the accuracy and complexity of our best performing classes of CKGE methods from the prior experiments (i.e. architecture modification methods are not considered). As in the previous experiments, our seven performance measures were chosen from prior continual learning work in robotics [57]. In each trial, the evaluation task is one-shot task execution, a real world robot task defined previously in chapter 5 and briefly summarized in this section for convenience.

VirtualHome environments: VirtualHome contains six household environments, each containing a kitchen, bedroom, bathroom, and livingroom. We extracted a KG from each unique environment that is used to train the robot's MRE before it performs task execution in the environment. Table 6.3 reports statistics of the KG extracted from each household in

Table 6.3: VirtualHome Dataset; 6 Households

	VirtualHome-6-H					
	H-1	H-2	H-3	H-4	H-5	H-6
$\lvert E^n\rvert$	209/(80%)	210/(88%)	194/(91%)	189/(92%)	206/(95%)	191/(97%)
$\lvert R^n\rvert$	11/(100%)	11/(100%)	11/(100%)	11/(100%)	11/(100%)	11/(100%)
$\lvert \mathcal{D}^n_{Tr}\rvert$	1,112/(40%)	1,077/(60%)	1,003/(78%)	929/(82%)	1,053/(93%)	974/(100%)
$\lvert \mathcal{D}^n_{Va}\rvert$	25/(21%)	22/(38%)	26/(59%)	23/(73%)	22/(90%)	19/(100%)
$\lvert \mathcal{D}^n_{Te}\rvert$	25/(20%)	22/(57%)	26/(56%)	23/(71%)	22/(87%)	19/(100%)

Table 6.4: VirtualHome Household Triple Overlap

	% of column household in row household					
	H-1	H-2	H-3	H-4	H-5	H-6
H-1	100%	48%	34%	49%	50%	37%
H-2	49%	100%	34%	46%	35%	35%
H-3	38%	36%	100%	39%	31%	27%
H-4	59%	54%	43%	100%	33%	33%
H-5	53%	36%	29%	29%	100%	40%
H-6	42%	39%	28%	32%	43%	100%

VirtualHome. The columns of the table denote the household (H-X, X$\in [1, 6]$), while rows correspond to the statistics as in Table 6.1, e.g., $|E^n|$ is the size of the entity set. Individual cells indicate the value, with coverage over the global KG that is extracted from all households combined shown in parentheses. In addition to Table 6.3, we provide Table 6.4 which shows the overlap in triples between each environment in virtual home. Sequentially evaluating the robot in each unique environment presents a continual learning scenario where the robot trains sequentially on subsets of a KG because the overlap of observed triples between environments is $\leq 60\%$ and the coverage of training triples increases between each household.

Evaluation procedure: In these experiments we measured how the robot's task execution success rate in each household was affected by the CKGE method used. In our experiments we sequentially trained the robot on datasets extracted from each unique environment in the simulator and evaluated the robot's task execution success rate on all previously encountered environments. In these experiments the robot performs robust execution of household cleaning tasks, using the system and definitions in chapter 5. In robust task execution, the robot is provided an execution environment E_x and a demonstrated task plan T_d that is recorded in a demonstration environment E_d, and asked to find a modified task plan T_x, executable in E_x, that accomplishes the goal(s) of T_d. Task plans are defined as a sequence of primitive actions, where each primitive action may or may not be parameterized by objects. Task plans are incrementally modified using knowledge inferences from a MRE with the system defined in chapter 5 to find an executable task plan. When the robot begins task execution in each household, it has the opportunity to update its MRE with the triples extracted from that household. Execution environments are sampled by perturbing the object used to demonstrate the task; changing the object's location, type, or both. The perturbations are made in accordance with the action-object-location distributions present in each VirtualHome environment (H-X, X$\in$ $[1, 6]$), ensuring that objects are not placed at implausible locations (e.g., broom inside the toilet) and that the intended generalization is not unreasonable (e.g. cleaning a table with a washing-machine). As a result, the demonstrated task plan often fails due to unsatisfied pre-conditions of primitive actions, and the robot must generalize the demonstrated task plan to formulate the *executable* task plan. Additionally, the prior distributions for which objects are likely to be spawned and their likely locations is unique to each household.

As our CKGE methods are all embedding agnostic, we used recent a state of the art MRE for this experiment, TuckER [32]. TuckER represents relationships using the Tucker decomposition, i.e., $\mathcal{W} \times_1 \mathbf{v}_h \times_2 \mathbf{W}_r \times_3 \mathbf{v}_t$ where $\mathcal{W}$ is the core tensor and $\times_n$ is a tensor product along the nth dimension [32]. It uses the *1-N* scoring and Bernoulli negative log loss functions in Equation 6.17 and Equation 6.18 where $p(h, r, t_i)$ is the predicted probability of tail entity $i \in \{1, ..., |\mathcal{E}|\}$ and y_i is a label indicating whether the relation r holds between h and t_i. Note that $p(h, r, t_i)$ is the predicted probability, not scoring function. The predicted probability p is a function of scoring function f that includes batch norms and dropout before putting output activations through a sigmoid function.

$$f(h, r, t) = \mathcal{W} \times_1 \mathbf{v}_h \times_2 \mathbf{W}_r \times_3 \mathbf{v}_t \tag{6.17}$$

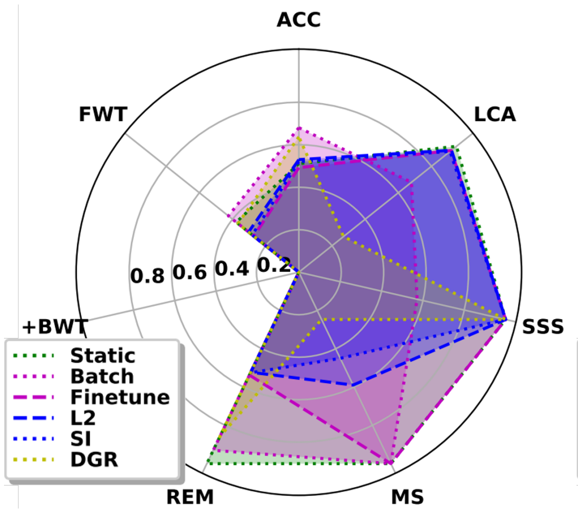
$$\mathcal{L} = -\frac{1}{|\mathcal{E}|}\sum_{i=1}^{|\mathcal{E}|}\Big(y_i \cdot \log\big(p(h,r,t_i)\big) + (1-y_i)\cdot \log\big(1-p(h,r,t_i)\big)\Big) \tag{6.18}$$

Evaluation measures: We used the same measures from the link-prediction experiments that summarize accuracy and forgetting across learning sessions: ACC, FWT, BWT, and REM. During each learning session, we compute the evaluation measures for the test sets of all learning sessions to characterize the effect of learning on prior, current, and future learning sessions. However, in these experiments only a single performance matrix $\mathbf{M} \in \mathbb{R}^{N\times N}$ is used to compute these measures with a different underlying performance metric. Specifically, for robust task execution performance, we consider the *success rate* of the robot to execute the demonstrated task T_d in the execution environment E_x. We capture space complexity for each CKGE method using MS and SSS measures as in the previous experiments [57]. For learning speed, we use the LCA measure as in the previous experiments [57].

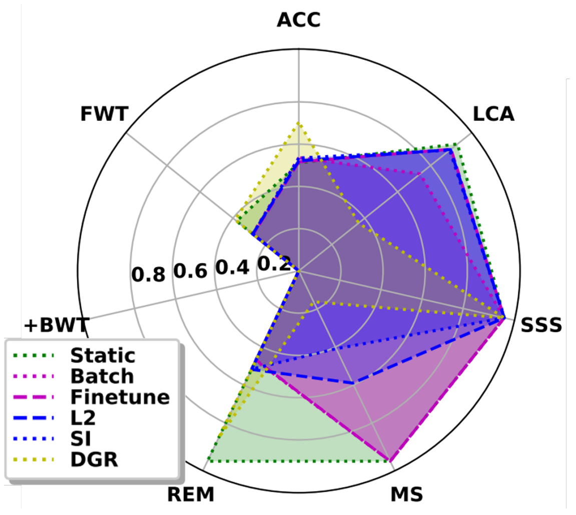
6.7 Experimental Results for Robust Task Execution

Results reported in this section are the average of five test runs in each experimental scenario; statistical significance is tested using repeated-measures ANOVA and a post-hoc Tukey's test. Any mention of 'significance' implies statistical significance at 95% significance level (i.e. $p < 0.05$).

As in the link-prediction experiments, we included Batch and Finetune methods as upper and lower bounds, respectively, for the expected task execution performance when using CKGE methods. In addition, to highlight the importance of adapting the robot's MRE as the robot encounters new environments, we also include results for a static MRE baseline. *Static* represents performance that can be achieved by learning a MRE in the initial environment that remains the same in each future environment encountered. The *Static* baseline represents the common approach of many prior KG representation works that assume static prior distribution across environments [13, 16, 6, 4].



(a)



(b)

Figure 6.6: Robot Task Execution Performance in (a) Unconstrained and (b) Data Constrained scenarios. Best viewed in color.

We sampled 40 initial demonstrations with 10 executions environments for a total of 400 robot executions per household, test run, and evaluated method combination. Before evaluating the robot's task execution performance in each household, we updated the MRE used by the robot according to the method being evaluated. We constructed three evaluation scenarios by subjecting the MRE update procedure to different constraints as in the link-prediction experiments. Unconstrained corresponds to a robot that has access to all prior training examples at training time. Data Constrained represents robots with access to limited training examples, e.g., only from the current learning session ($\mathcal{D}^{n}_{Tr}$). Last, in Time and Data Constrained the robot only has access to training examples for the current learning session and has limited time to update the MRE. We varied the number of training epochs in each learning session from 15 to 480 to characterize how each method is affected by the varying time constraint. Here the ranges of each measure are in $[0, 1]$ and larger values are better. Overall, the results from each scenario further substantiate our results from the link-prediction experiments.

- In an *unconstrained scenario* (Figure 6.6a), such as one in which a robot might have access to a cloud compute service where it uploads all previously observed household data, Batch learning provides significantly higher ACC and FWT compared with other methods. It is also worth noting the REM and LCA performance of the Static baseline. The Static baseline has perfect REM performance because the task execution performance remains the same throughout all learning sessions due to the static MRE supporting the robot's inferences (i.e. the initial performance on a household never changes). Additionally, LCA is close to 1.0 because the MRE is static and does not update between households after training on the dataset extracted from the first household (i.e. initial performance in each household after the first is the best performance for the household).

- In a *data constrained scenario* (Figure 6.6b), the robot might be subject to data-privacy restrictions and only have access to training examples for the current household to update the MRE. Batch's inference performance collapses because prior observations are unavailable. Under these constraints, DGR is the best choice, with significantly better ACC than other methods because it approximates Batch in the unconstrained scenario. The other methods besides DGR have similar ACC performance because a baseline level of performance can be achieved after training on the first household due to similarities between households (evident from the Static baseline in Figure 6.6 and household overlap in Table 6.4).
- In a *data and time constrained scenario* (Figure 6.7), a robot might have varying utilization rates in addition to the data privacy restrictions. Therefore, as robot utilization increases, the time available (in number of training epochs) to update the robot's semantic representation when transitioning between households decreases. As training time becomes more constrained, DGR is a poor choice because there is not enough time to sufficiently train the generative model. Batch learning too degrades in performance with tighter time constraints because there is not ample time to train a fresh MRE from scratch. L2R, SI, and Finetune are better choices in more time constrained scenarios as they incrementally update the MRE, requiring less training epochs to update the MRE. Static is not included in Figure 6.7 as its performance does not depend on the training time available, and therefore, did not differ from the prior two scenarios.

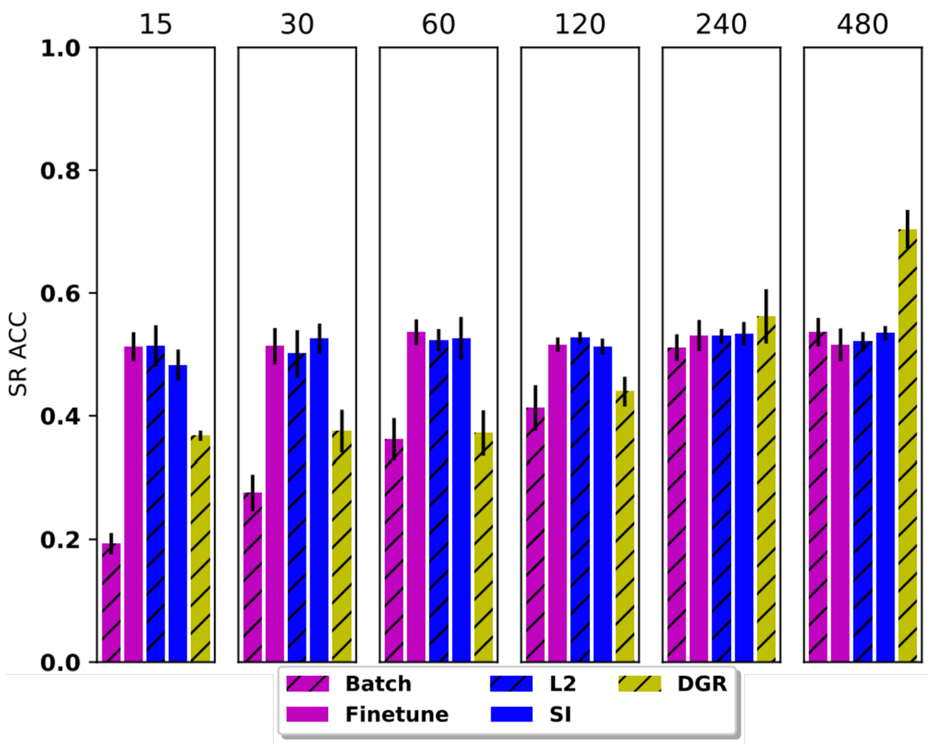


Figure 6.7: SR ACC for a range of 15 to 480 of training time to update the robot's semantic representation. Black errors bars indicate standard deviation. Best viewed in color.

6.8 Discussion & Conclusion

KG embeddings are increasingly being used as semantic representations in robotics applications, but it is difficult to update these representations incrementally. Our work introduced five representative continual learning-inspired methods for continual KG embedding (CKGE). We also introduced a heuristic sampling strategy and generated CKGE datasets based on benchmark KGs and a KG for the service robotics domain. Furthermore, we identified and built on measures for evaluating continual learning in robotics. We evaluated our embedding-generic methods on two KG embedding representations.

Experimental evaluation using the benchmark KGs provided key insights characterizing the use of our CKGE methods in terms of factors such as inference, learning speed, and memory requirements. Our evaluation using the service robotics domain knowledge characterized the use of CKGE methods in three different classes of semantics-driven robotics applications. In unconstrained scenarios, iterative batch learning provides the best performance because training data from prior learning sessions are available and training time in each learning session is unlimited. In data constrained scenarios, deep generative replay methods are the best choice as they approximate batch learning without having access to data from prior learning sessions. In data and time constrained scenarios, regularization methods are the best choice as they incrementally update embeddings leveraging weights optimized in prior learning sessions and mitigating catastrophic forgetting. Both iterative batch learning and deep generative replay methods perform worse than regularization methods in data and time constrained scenarios because there is not enough training time to learn an accurate generative model or new discriminative model.

One avenue for future work is to consider how to combine different classes of continual learning techniques to produce a method that is robust across multiple robot application scenarios (e.g., data constrained, data and time constrained). Additionally, our robot application scenarios present a broad set of categories for two types of constraints robots might encounter (e.g., time constraints due to high robot utilization). Further consideration is needed to explore the space of foreseeable robot constraints and application scenarios. Finally, our adaptations of existing architecture modification techniques (CWR) did not perform well across our experiments. We believe poor architecture modification technique performance is due to large differences between neural representations for object recognition and KG embedding. We consider the development of architecture modifications techniques for MREs another open question for future research.

6.9 Findings & Contributions

This work comprises our contributions to **the development and evaluation of methods to sequentially update multi-relational embeddings**. We did this by relaxing the assumptions of the KG representation formulated in chapter 4, namely, that all training facts are available for training in the first learning session. We developed and evaluated five continual KG embedding methods and provided guidance on which to select given the constraints of a robotics application. Additionally, by using our CKGE methods in the task generalization module from chapter 5, we showed how our KG representation can be made more adaptable to sequentially train the MRE on disjoint subsets of a KG. Therefore, our CKGE methods enable a robot operating in multiple environments with varying prior distributions (e.g., different houses, kitchen to bathroom) to incrementally include new facts observed without corrupting previous semantic knowledge about a domain.

CHAPTER 7
EXPLAINABLE KNOWLEDGE GRAPH EMBEDDING

In this chapter we present a method to provide natural-language explanations of inferences made by MREs. Interpretable and explainable decision making is critical to establish trust between autonomous robots and their end users, promoting long-term interaction and collaboration. Explainable decision making is particularly important to establish trust for robots interacting with non-expert users, like the system we developed using our proposed KG representation in chapter 5. The explainability method developed in this chapter is also incorporated into the system from chapter 5 to evaluate how the robot's decision making about generalizations of prototypical task plans can be improved using feedback from non-experts elicited by the robot's explanations.

7.1 Introduction

Prior work has shown that learned, potentially latent, KG representations can be used to support robots operating in challenging environments with non-expert users. The complex knowledge inferences afforded by learned KG representations can be used to improve a robot's robustness in ambiguous or unforeseen scenarios. Some examples include tool substitution [16] and interpolating ambiguous end user commands [13]. However, these learned KG representations are usually black-boxes that are not interpretable to a non-expert user, who would require an explanation when the robot has erratic behavior due to an incorrect knowledge inference.

XAIP seeks to explain an AI's reasoning to humans in sequential decision-making procedures to promote collaboration. In XAIP, inference reconciliation through dialogue with the AI is one method of explaining an AI's reasoning to a user [58]. The growing variety of questions users may ask an AI addressed by prior work include "Why is action a in plan

π?", "Why not this other plan π'?", "Why is this policy (action) optimal?", and others. Additionally, prior work has proposed inference reconciliation frameworks that explain how plans, policies, rationales, and scene-graphs can be leveraged to explain an AI's decision making. However, to the best of our knowledge, no existing inference reconciliation framework explains how a *KG representation* affects a robot's decision making.

In its interactions with a user, a robot might need to justify its action based on semantic *knowledge inferences* independent of the robot's plan, policy, or the scene. For example, after asking the robot to fetch their coffee, the user might ask the robot "Why are you looking in the refrigerator?", to which the robot might reply "Food is stored in the refrigerator, and coffee is a food". Such an explanation not only elucidates the robot's reasoning, but also provides a valuable opportunity for the user to correct the robot's knowledge (e.g., "coffee is stored in the pantry"). Questions and explanations of that form differ qualitatively from why questions addressed in prior work and require reasoning about the semantic relationships between coffee, refrigerators, and other world entities to provide an explanation. The aim of our work is to develop such explanation capabilities, and ultimately improve the robot's reasoning capabilities, by introducing a novel type of inference reconciliation of the form, "Why is knowledge inference i, supporting action a, true?".

We introduce an inference reconciliation framework that answers a user's questions about the knowledge inference that supports a robot's action, based on KGs, XAI, and natural language (Figure Figure 7.1). Our framework uses a pedagogical XAI approach to provide explanations to non-experts about the inferences made by a learned KG representation. We develop an interpretable graph feature model as the student, using subgraph feature extraction and decision trees. We train the graph feature model to locally approximate the predictions of the learned KG representation and provide a grounded, natural language explanation for each prediction.

We evaluate our framework across three dimensions: algorithmic performance, user preference of explanations, and robot task performance. In our algorithmic evaluation, we

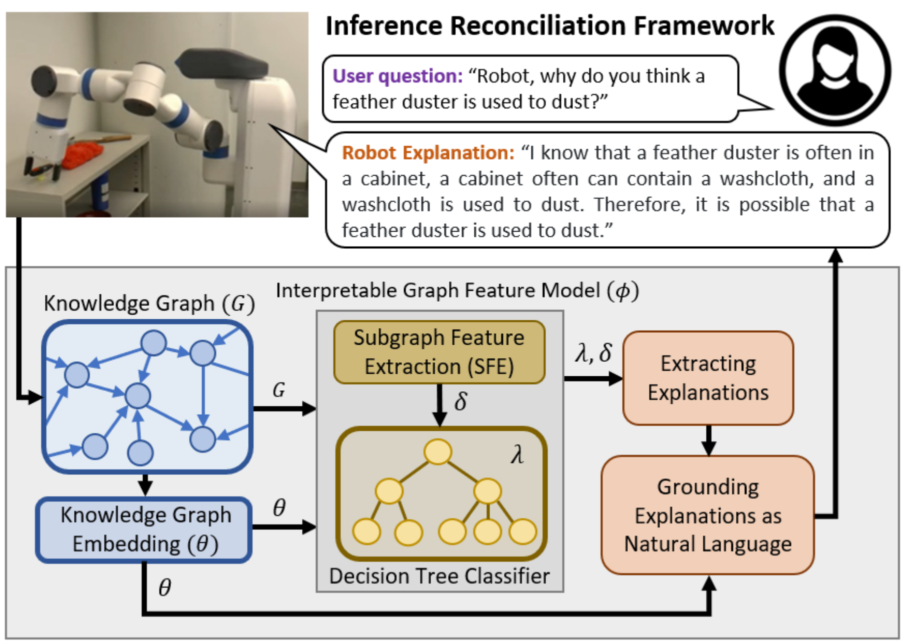

Inference Reconciliation Framework
User question: "Robot, why do you think a feather duster is used to dust?"
Robot Explanation: "I know that a feather duster is often in a cabinet, a cabinet often can contain a washcloth, and a washcloth is used to dust. Therefore, it is possible that a feather duster is used to dust."
Knowledge Graph (G)
Interpretable Graph Feature Model (ϕ)
Subgraph Feature Extraction (SFE)
δ
λ
Decision Tree Classifier
G
λ, δ
Extracting Explanations
Grounding Explanations as Natural Language
Knowledge Graph Embedding (θ)
θ
θ

Figure 7.1: Overview of our inference reconciliation framework which introduces a novel explainable KG embedding method (graph feature model), leveraging decision trees, to provide natural language explanations to users.

observe statistically significant improvements in classification fidelity over baseline interpretable graph feature models, substantiating our design choices. We further analyzed our design choices through an ablation study. In our user preference evaluation, we observe that users prefer our explanations as robot responses with statistically significant differences in 73.3% of analyzed interactions. These differences in preferences validate the need for our inference reconciliation framework as it answers "why questions" that are qualitatively different from prior work. Most importantly, in our robot task evaluation, we observe that non-expert feedback prompted by our explanations can effectively improve robot task performance (117% and 33.7% relative improvement in link-prediction and task execution success rate, respectively). The novelties of our work include the inference reconciliation framework as a system, our interpretable graph feature model, and our framework's ability to generate explanations that help non-experts improve robot task performance. Specifically:

1. We introduce an inference reconciliation framework that answers a novel type of user

questions of the form "Why is knowledge inference i, supporting action a, true?" using KGs;

2. We develop and evaluate a novel graph feature model that outperforms prior work by statistically significant margins on a household knowledge dataset;

3. We showcase a novel application of explanations within XAIP: improving downstream task performance, namely, robot behavior.

7.2 Methodology

The problem of inference reconciliation is grounded in the notion that users typically have less computational ability compared to AI systems, making it difficult for users to understand an AI's solution. One solution entails providing explanations that aid in the user's inferential capabilities [58]. In our work, we consider robotic frameworks in which robot behavior is driven by knowledge inference. In particular, we assume the robot is making decisions based at least in part on a learned KG representation. To aid user understanding of robot behavior, we introduce an inference reconciliation framework [1] that answers user questions of the form: "Why is knowledge inference i, supporting action a, true?" (e.g. user asks a robot commanded to find a sponge, "Why do you think you will find a sponge in the sink?"). We leverage both KGs and interpretability techniques from XAI to provide non-expert users with natural langauage explanations about a robot's knowledge inference supporting an action. As shown in Figure 7.1, we first gather facts about a robot's task domain, forming a KG, $\mathcal{G}$. We use the KG $\mathcal{G}$ to learn a multi-relational (KG) embedding (MRE), Θ, that enables the robot to make complex knowledge inferences. However MREs are black-box, and lack a mechanism to explain inferences to non-expert users (subsection 7.2.1). To make MREs more interpretable to non-experts, we follow a pedagogical XAI approach [156] to provide explanations about Θ's inferences, using an interpretable

[1]Supplementary materials: https://github.com/adaruna3/explainable-kge

graph feature model as the student, Φ. Both $\mathcal{G}$ and Θ serve as inputs to Φ, which performs subgraph feature extraction (SFE) and trains a decision tree classifier to locally approximate the predictions of Θ (subsection 7.2.2). Given a prediction from Θ, we then use Φ and Θ to extract and ground the explanation in natural language (subsection 7.2.3).

7.2.1 Knowledge Graph Representation

KGs are modeled as a graph $\mathcal{G}$ composed of individual facts or triples (h, r, t); h and t are the head and tail entities (respectively) for which the relation r holds, e.g., (*cup, hasAction, fill*) [4, 6, 143]. KGs that model real-world domains tend to have the properties of being *large*, *sparse*, and *incomplete*. For example, a KG representing a household, while large, only represents a subset of true facts, which are sparse in a space of many potential facts. We adopt multi-relational (KG) embeddings (MRE) for our KG representation because MREs are designed for KGs that are large-scale and sparse [26]. Additionally, MREs excel at learning the underlying structure of graphs to infer new facts beyond known facts in a graph (i.e. latent feature models in section 2.5). We build upon the framework in [143], which uses a MRE to represent $\mathcal{G}$.

MREs are distributed representations that model $\mathcal{G}$ in vector space [26], learning a continuous vector representation from a dataset of triples $\mathcal{D} = \{(h, r, t)_i, y_i | \, h_i, t_i \in \mathcal{E}, r_i \in \mathcal{R}, y_i \in \{0, 1\}\}$, with $i \in \{1...|\mathcal{D}|\}$. Here y_i denotes whether relation $r_i \in \mathcal{R}$ holds between $h_i, t_i \in \mathcal{E}$. Each entity $e \in \mathcal{E}$ is encoded as a vector $\mathbf{v}_e \in \mathbb{R}^{d_\mathcal{E}}$, and each relation $r \in \mathcal{R}$ is encoded as a mapping between vectors $\mathbf{W}_r \in \mathbb{R}^{d_\mathcal{R}}$, where $d_\mathcal{E}$ and $d_\mathcal{R}$ are the dimensions of vectors and mappings respectively [26]. The embeddings for $\mathcal{E}$ and $\mathcal{R}$ are typically learned using a scoring function $f(h, r, t)$ that assigns higher (lower) values to positive (negative) triples [26]. The learning objective is thus to find a set of embeddings $\Theta = \{\{\mathbf{v}_e | \, e \in \mathcal{E}\}, \{\mathbf{W}_r | \, r \in \mathcal{R}\}\}$ that minimizes the loss $\mathcal{L}_\mathcal{D}$ over $\mathcal{D}$. Loss $\mathcal{L}_\mathcal{D}$ can take many forms depending on the KGE representation used, e.g., Negative Log-Likelihood Loss [32].

We make inferences (i.e. fact predictions) in MREs by completing a transformation in the embedding space. For example, to infer tails $\{t_j | \, t_j \in \mathcal{E} \, \forall j\}$ that might complete $(h, r, \_)$, the scores $f(h, r, t_j)$ of all j triples are computed, and triples with scores meeting some classification threshold are classified true. Each score $f(h, r, t_j)$ is the resultant of a sequence of high-dimensional geometric transformations between the head entity vector $\{\mathbf{v}_h | \, h \in \mathcal{E}\}$, relation mapping $\{\mathbf{W}_r | \, r \in \mathcal{R}\}$, and tail entity vectors $\{\mathbf{v}_{t_j} | \, t \in \mathcal{E}\}$. Given the complex and relative nature of MREs, inferences are not inherently interpretable, as discussed in section 2.5 under explainable knowledge base completion. With the ultimate objective of providing transparent explanations of MRE inferences to non-experts, we leverage explainability techniques from XAI to explain each MRE inference.

7.2.2 Interpretable Model

Our student model Φ locally approximates the inferences (i.e. fact predictions) of the MRE, denoted as Θ such that Φ can provide explanations of predictions made by Θ. As discussed in section 2.5, graph feature models use graph features to infer missing facts. We develop a novel interpretable graph feature model Φ that consists of two components: interpretable graph features δ derived from $\mathcal{G}$ and Θ, and an interpretable classifier λ trained on δ to approximate the predictions of Θ. We begin by extracting the features δ derived from a KG $\mathcal{G}$ as in [157], which is formed from a dataset of triples $\mathcal{D} = \big\{(h, r, t)_i, y_i | \, h_i, t_i \in \mathcal{E}, r_i \in \mathcal{R}, y_i \in \{0, 1\}\big\}$, with $i \in \{1...|\mathcal{D}|\}$. Here, y_i denotes whether relation $r_i \in \mathcal{R}$ holds between $h_i, t_i \in \mathcal{E}$. We then train the interpretable classifier λ over the most relevant subsets of these features to infer a triple (i.e. fact) missing from $\mathcal{G}$.

Interpretable Knowledge Graph Features

We use Subgraph Feature Extraction (SFE) from [157] to extract interpretable graph features, δ, from a graph $\mathcal{G}'$, which represents a set facts believed true the the MRE Θ. We begin by forming $\mathcal{G}'$ using the facts in $\mathcal{G}$ classified as true by Θ. We also add to $\mathcal{G}'$ other facts classified as true by Θ, which we form by switching the head h or tail t for a fact

(h, r, t) in $\mathcal{G}$ with the top K nearest neighbor entities. We used $K = 3$, but this parameter can be tuned to include or exclude more graph features for explanations. Neighbors are determined by cosine similarity in the embedding space. We then used SFE to extract our interpretable features, δ, from $\mathcal{G}'$ instead of $\mathcal{G}$ to get a larger set of features classified as true by Θ. SFE uses bi-directional breadth-first search to find all unique relation paths connecting a pair of entities in $\mathcal{G}'$. Relation paths are formed from the sequence of relations that are traversed when following a path in $\mathcal{G}'$ from a head entity h to a tail entity t, where $h, t \in \mathcal{E}$. Therefore, for a relation path $\hat{\mathcal{P}}$ composed of L relations, $\hat{\mathcal{P}}^{\ell}$ represents each relation on a path where $\ell \in \{1...L\}$. We encode the unique relation paths connecting entities h, t paired by a relation r as features using one-hot encoding.

Explainable Model Training

We train a separate interpretable model λ, a decision tree, for each knowledge inference on a subset of available features that maximize λ's classification fidelity to Θ. The joint contributions of relation path combinations extracted from $\mathcal{G}'$ may not be modeled as linear combinations of individual paths. For example, two features together exhibiting an XOR relationship. Therefore, we use a decision tree for our interpretable model, λ, given that decision trees are able to model nonlinear decision boundaries, while remaining interpretable. Additionally, decision trees have more explicit semantics about which relation paths (i.e. features) contribute to a classification (i.e. features along the decision path), excluding extraneous relation paths that may be correlated with a class but are unnecessary to make the classification.

We sampled training examples based on their semantic similarity to the fact to be explained instead of using all available examples to train the decision tree for each fact (h, r, t) to be explained. By only training over a set of the most semantically similar examples, the decision tree λ is trained to locally approximate the decision boundary of the embedding Θ. This locality is tuned as a hyper-parameter to maximize the classification fidelity between the decision tree λ and embedding Θ. Given a fact (h, r, t) to be inferred, in which $h, t \in \mathcal{E}$ and $r \in \mathcal{R}$, we implemented the locality by selecting the K nearest neighbor facts (h_k, r, t_k), where $k \in \{1...K\}$, sharing a common relation r such that the classification fidelity between Φ and Θ is maximized. Neighbors are determined by cosine similarity in the embedding space (e.g., cosine$(\mathbf{v}_h, \mathbf{v}_{h_k})$). Here K serves as a relation specific hyper-parameter to control the locality of sampled training examples.

7.2.3 Extracting and Grounding Explanations

Given a knowledge inference extracted from a user's "why question" during inference reconciliation, we use the classifier λ in Φ to extract the relevant relation paths, the embedding Θ to ground these relation paths to paths in $\mathcal{G}'$, and templates to convert the grounded paths into natural language explanations. The knowledge inference takes the form (h, r, t), where $h, t \in \mathcal{E}$ and $r \in \mathcal{R}$. Assuming Θ and Φ are in agreement about the classification of the query knowledge inference, we extract N relevant relation paths $\hat{\mathcal{P}}_n$, where $n \in \{1...N\}$, between h and t that were present on the decision path when the decision tree (i.e. λ) performed classification. For each path $\hat{\mathcal{P}}_n$, we perform bi-directional breadth first search over grounded paths between h and t using relations from $\hat{\mathcal{P}}_n$ in order to ground the relation paths using Θ. Therefore, for a relation path $\hat{\mathcal{P}}_n$ composed of L relations $\hat{\mathcal{P}}_n^{\ell}$, where $\ell \in \{1...L\}$, the breadth first search at h begins with the relation $\hat{\mathcal{P}}_n^0$ while the backward breath first search at t begins with the relation $\hat{\mathcal{P}}_n^{L-1}$. Both recursive searches repeatedly perform inference using Θ by classifying the tails (heads) that complete the previous head (tail) and relation in $\hat{\mathcal{P}}_n$ for the current search step, forming a grounded relation path. When there is overlap between the two searches, an inference is performed that simultaneously classifies the connecting entity serving as a head and tail using Θ to ensure the connecting entity exists for the path. After completing the search, all grounded paths $\mathcal{P}_n$ made up of relationships classified as true by the Θ have been found and ranked in order of belief according to Θ, which can be accumulated during the search. These grounded paths can then be post-processed using templates for each relation type $r \in \mathcal{R}$ and automatically corrected for grammar to produce natural-language explanations to users [158].

7.3 Experimental Evaluations

We evaluated our inference reconciliation framework with respect to algorithmic performance, user preference, and robot task performance. In our algorithmic evaluation, we used a household robot dataset to compare our interpretable graph feature model Φ with prior work. We followed with an ablation study on the crucial components of our algorithm design to substantiate each design choice. In our user preference evaluation, we measured whether there were significant differences in user's preferences towards our explanations when a robot is asked "why questions" during task execution. In our robot task evaluation, we measured how non-expert feedback to the robot elicited by our explanations affected robot task performance.

7.3.1 Evaluation of Interpretable Graph Feature Model

Our first evaluation qualitatively and quantitatively compared our graph feature model with baseline graph feature models. Qualitatively, we considered the different features necessary for our use case and how each baseline compared with our approach in terms of these features. Quantitatively, we measured the extent to which the classifications of each considered graph feature model Φ (i.e. ours and baselines) approximates Θ's classifications (i.e. classification fidelity). Classification fidelity is a proxy measure of whether explanations produced by Φ explain Θ's reasoning [156, 70]. For quantitative comparisons we used an evaluation procedure proposed in [70] for the test split of a dataset $\mathcal{D} = \big\{(h,r,t)_i, y_i |\, h_i, t_i \in \mathcal{E}, r_i \in \mathcal{R}, y_i \in \{0,1\}\big\}$, with $i \in \{1 ... |\mathcal{D}|\}$ (see Quantitative Comparison below). Each y_i denotes whether relation $r_i \in \mathcal{R}$ holds between entities $h_i, t_i \in \mathcal{E}$. We checked for significant differences in mean classification fidelity using five-fold cross-validation over $\mathcal{D}$.

Table 7.1: Comparison of Explainable KBC Methods

Method	User?	MRE Agnostic?	Negative Correlations?	Stand-alone?	F1 Fidelity (μ, σ)
SimplE [72]	Expert	✗	✓	✓	N/A
ITransF [71]	Expert	✗	✓	✓	N/A
CrossE [74]	Non-Expert	✗	✓	✓	N/A
DistMult [73]	Non-Expert	✓	✗	✓	N/A
OxKBC [75]	Non-Expert	✓	✓	✗	N/A
XKE [70]	Non-Expert	✓	✓	✓	$(87.9, 3.4)$
Ours	Non-Expert	✓	✓	✓	$(98.9, 0.1)$
Ours ($\forall$, DT)	Non-Expert	✓	✓	✓	$(95.2, 3.0)$
Ours ($\forall$, LR)	Non-Expert	✓	✓	✓	$(87.9, 3.4))$

Qualitative Comparison

We performed a qualitative comparison between graph feature models Φ from section 2.5 to select appropriate baselines for quantitative evaluation. The summary of our qualitative comparisons between all graph feature models Φ is shown in Table 7.1. We did not include SimpleE and ITransF in the quantitative comparison because relative attention weights between relations and entities are not interpretable to non-experts. Additionally, our graph feature model was designed to be embedding agnostic, allowing robotics practitioners to use the current state of the art MRE, eliminating CrossE as a baseline. Rule-Mining (DistMult) was not considered because rule-support cannot provide explanations in cases with no positively correlated relation paths because support does not reason about negative correlations between relation paths. We excluded OxKBC because instead of KG correlations, it uses expert annotations to determine which explanation is best suited for a classification, which may not provide interpretability into the MRE's (i.e. robot's) beliefs. Thus, the prior method we quantitatively compared against is XKE, as it met all previously mentioned considerations critical to our application.

Quantitative Comparison

We compared our approach with XKE [70] quantitatively using an evaluation procedure from [70]. We first generated the inputs held constant during evaluation, the dataset $\mathcal{D}$

Table 7.2: Dataset gathered from VirtualHome to learn Θ

Relation	$\lvert E_{head}\rvert^{\dagger}$	$\lvert E_{tail}\rvert^{\dagger}$	$\lvert \mathcal{D}_{Tr}\rvert^{\dagger}$	$\lvert \mathcal{D}_{Va}\rvert^{\dagger}/\lvert \mathcal{D}_{Te}\rvert^{\dagger}$	$\lvert \mathcal{D}\rvert$
HasEffect	31	16	25	3	31
InverseActionOf	12	12	12	1	14
InverseStateOf	16	16	14	1	16
LocInRoom	43	4	86	10	106
ObjCanBe	183	35	1,369	171	1,171
ObjInLoc	97	24	120	15	150
ObjInRoom	183	4	334	41	416
ObjOnLoc	170	33	292	36	364
ObjUsedTo	52	22	61	7	75
ObjHasState	183	20	1,065	133	1,331
OperatesOn	52	188	1,939	242	2,423

Example entities (291 total entities)	
Rooms (4)	kitchen, bedroom, bathroom, livingroom
Locations (43)	fridge, table, sink, garbage, bed, desk, cabinet, drawer
Objects (189)	chair, towel, bleach, tomato, rug, plant, fork, laptop
Actions (35)	wipe, open, pick up, turn off, bake, unplug, disinfect
States (20)	dirty, clean, on, off, cooked, broken, open, plugged in

†Values for an example fold of $\mathcal{D}$

and MRE Θ. We gathered a household robot dataset $\mathcal{D}$ of unique triples from a household simulator, VirtualHome [148], containing train $\mathcal{D}_{Tr}$, valid $\mathcal{D}_{Va}$, and test $\mathcal{D}_{Te}$ splits (Table 7.2). We used the TuckER [32] MRE, to represent MRE Θ learned from $\mathcal{D}$. TuckER represents relationships using the Tucker decomposition, i.e., $\mathcal{W} \times_1 \mathbf{v}_h \times_2 \mathbf{W}_r \times_3 \mathbf{v}_t$ where $\mathcal{W}$ is the core tensor and $\times_n$ is a tensor product along the nth dimension [32]. It uses the *1-N* scoring and Bernoulli negative log loss functions in Equation 7.1 and Equation 7.2 where $p(h, r, t_i)$ is the predicted probability of tail entity $i \in \{1, ..., |\mathcal{E}|\}$ and y_i is a label indicating whether the relation r holds between h and t_i. Note that $p(h, r, t_i)$ is the predicted probability, not scoring function. The predicted probability p is a function of scoring function f that includes batch norms and dropout before putting output activations through a sigmoid function.

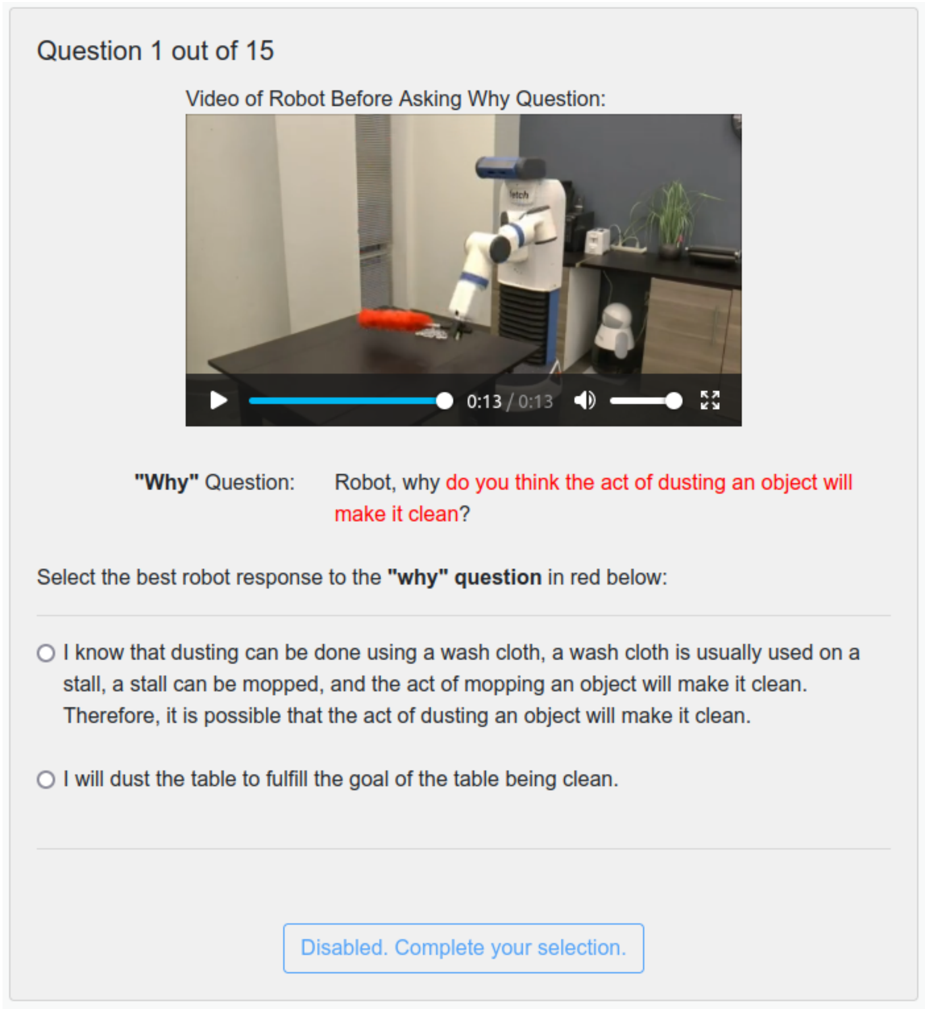
$$f(h, r, t) = \mathcal{W} \times_1 \mathbf{v}_h \times_2 \mathbf{W}_r \times_3 \mathbf{v}_t \tag{7.1}$$

$$\mathcal{L} = -\frac{1}{|\mathcal{E}|}\sum_{i=1}^{|\mathcal{E}|}\Big(y_i \cdot \log\big(p(h,r,t_i)\big) + (1-y_i)\cdot\log\big(1-p(h,r,t_i)\big)\Big) \tag{7.2}$$

We then generated a graph feature model Φ for our approach and XKE using the same inputs, the MRE Θ and dataset $\mathcal{D}$. We measured the classification fidelity of each Φ to Θ. We measured classification fidelity as F1-Fidelity between Φ and Θ classifications, in which the MRE's classifications served as labels [70]. We checked for significant differences in mean F1-Fidelity across a five-fold cross-validation over $\mathcal{D}$ using repeated-measures ANOVA and a post-hoc Tukey's test. Please visit the footnote in section 7.2 for supplementary materials detailing the implementation of Θ and each Φ, the tuning of hyper-parameters, the evaluation dataset, results, and statistical analyses that are omitted here for brevity. Our results in Table 7.1 show that there is a statistically significant (p=0.001) improvement in the mean F1-Fidelity between our graph feature model and XKE's.

Ablation Study

We further analyzed our approach by performing an ablation study to understand how each component of our graph feature model contributed to the overall improvement in performance. We toggled two novel parts of our graph feature model not present in XKE: the use of decision trees as Φ and the locality of examples to train Φ. We followed the same procedure as in the previous experiment, the results of which are in Table 7.1. The first ablation, ($\forall$, DT) in Table 7.1, shows a significant (p=0.02) drop in performance due to including all available relation paths to train Φ, which is still a decision tree. The fidelity of Φ drops here due to the challenge of making an interpretable model approximate the global decision boundary of a black box model [156]. The second ablation, ($\forall$, LR) in Table 7.1, shows a significant (p=0.001) drop in performance due to modeling Φ using using logistic regression instead of a decision tree, in addition to including all available relation paths to train Φ. Here, the fidelity of Φ drops due to inaccurately modeling the non-linear joint contributions of relation paths, discussed in subsubsection 7.2.2.


Question 1 out of 15

Video of Robot Before Asking Why Question:

"Why" Question: Robot, why do you think the act of dusting an object will make it clean?

Select the best robot response to the **"why" question** in red below:

○ I know that dusting can be done using a wash cloth, a wash cloth is usually used on a stall, a stall can be mopped, and the act of mopping an object will make it clean. Therefore, it is possible that the act of dusting an object will make it clean.

○ I will dust the table to fulfill the goal of the table being clean.

Figure 7.2: Preferences User Study GUI.

7.3.2 Evaluation of Explanation Preferences

Next, we evaluated our inference reconciliation framework from a non-expert's perspective. We performed a user study to characterize the relationship between different types of "why questions" asked to a robot and a non-expert's preferred types of explanations as responses from the robot. The study evaluated two types of "why questions" asked to discern a robot's actions during cleaning tasks: causal and knowledge inference. Our causal questions were those that inquired about the causal need for an action and had the form "Why is action a in plan π? (e.g. "Why will you move to the sink?"). Our knowledge inference questions were those that inquired about the underlying inferences supporting an action and had the form "Why is knowledge inference i, supporting action a, true?" (e.g. "Why do you think

Table 7.3: Preferences User Study Example Questions and Responses

Knowledge Inference Why Question	Causal Why Question
Why do you think you will find a washing sponge in a sink?	Why will you move to the sink?
Why do you think a scrubber is used to scrub?	Why will you grab the scrubber?
Why do you think the act of scrubbing an object will make it clean?	Why will you scrub the kitchen table with the scrubber?
Knowledge-graph-based Robot Responses	
I know that dusting can be done using a wash cloth, a wash cloth is usually used on a stall, a stall can be mopped, and the act of mopping an object will make it clean. Therefore, it is possible that the act of dusting an object will make it clean.	
I know that a cleaning rag is often in a cabinet, a cabinet often can contain a towel, and a towel is used to wipe. Therefore, it is possible that a cleaning rag is used to wipe.	
Causal-link-based Robot Responses	
I will sponge the table to fulfill the goal of the table being clean.	
I will move from the table to the sink to later grab the scrubber from the sink.	
I will take the scrubber to later be able to scrub the table with the scrubber.	
I will wipe the table to fulfill the goal of the table being clean.	

you will find a sponge in the sink?) (see Table 7.3). Prompted by the "why question", users selected their preferred explanation from a list provided as possible robot responses. Our null hypothesis was that the type of "why question" asked to a robot would not have a significant affect on the type of explanation from the robot preferred by a non-expert.

Study Design

We recruited 50 users from Amazon's Mechanical Turk (AMT) to perform the study, all whom were 18 years or older (M=40.7, SD=10.3). The study was between-subjects, given that each participant was randomly assigned to evaluate only one of two types of "why questions" for each interaction. We counterbalanced for ordering effects by randomizing the question order and robot response type order. Of the 50 participants, 17 participants were filtered out for incorrectly answering any of four filtering questions (not included in results) scattered throughout the study, each of which had one sensical and nonsensical response (i.e."..take the scrubber to later slay a fire breathing dragon.."). The remaining 33 participants were used to evaluate 15 robot interactions.

In the study, users completed an online assessment that simulated situating users in multiple robot interactions and prompted users with a question about the interaction. For each interaction, participants were shown a short video of the robot executing unique actions towards a cleaning task before being interrupted to answer a designated “why question”. Figure Figure 7.2 provides an example of the user’s interface. The “why question” in each interaction was specific to each unique action being interrupted but was either causal or knowledge inference type, as defined above.

Users were then tasked with selecting the “best robot response” to the “why question”. Users were provided with two different explanation types as robot responses: causal-link-based and knowledge-graph-based. The causal-link-based explanations were generated from a recent state-of-the-art plan verbalization and explanation method based on causal-link-chaining [61]. The knowledge-graph-based explanations were generated by our inference reconciliation framework in section 7.2 using the dataset $\mathcal{D}$ from Table 7.2 and MRE Θ from subsubsection 7.3.1. Example questions and responses in the assessment are shown in Table 7.3. Note, we do not evaluate explanations using scene-graphs, policies, or rationales (section 2.5), given that their assumptions do not align with our robot cleaning task (i.e. objects relevant to answering the “why question” were never in scene, the robot had no prior observed states relevant to the “why question”, and rationales ascribe a non-expert’s reasoning to world and agent states instead of exposing the robot’s reasoning).

Study Results

We performed a Chi-square test of independence followed by repeated Fisher’s exact method measures to analyze user responses aggregated in an individual contingency table for each robot interaction in the assessment. We accounted for Type I error due to Fisher’s repeated measures using Simes’ alpha correction [159]. In Table 7.4, we summarize the statistical analyses for each of the 15 robot interactions analyzed, excluding the filtering question instances. We observed that when asked a knowledge inference “why question”, participants

Table 7.4: Results of Preferences User Study

#	Chi P-val	Fisher P-val	Interrupted Robot Task
1	**0.01**	**0.006**	Find disinfectant brush on kitchen table
2	**0.01**	**0.006**	Grab disinfectant brush to disinfect
3	**0.02**	**0.01**	Grab scrubber to scrub
4	**0.00001**	**0.00001**	Find washing sponge in sink
5	0.58	0.44	Grab cleaning rag to wipe
6	0.23	0.19	Find scrubber in sink
7	**0.001**	**0.0005**	Disinfect table with disinfectant brush
8	**0.004**	**0.002**	Grab washing sponge to sponge
9	0.13	0.12	Find cleaning rag on kitchen counter
10	**0.02**	**0.01**	Grab feather duster to dust
11	**0.0001**	**0.00001**	Find feather duster in cabinet
12	0.15	0.15	Sponge table with washing sponge
13	**0.002**	**0.001**	Wipe table with cleaning rag
14	**0.009**	**0.007**	Scrub table with scrubber
15	**0.003**	**0.001**	Dust table with feather duster

select a knowledge-graph-based robot response more often than not in all questions except 5, and that 73.3% of these differences in selections were significant (in **bold**). Similarly, when asked causal “why questions”, participants select a causal-link-based robot response more often than not in all questions, and 73.3% of these differences are significant. Overall, our results indicate that non-experts recognize the qualitative differences in the two types of “why questions”, which tend to significantly effect their preferred type of robot response. In other words, there is a need for knowledge-based robot responses given they are better suited for knowledge inference “why questions” whereas the existing causal-link-based responses are better suited for causal “why questions”.

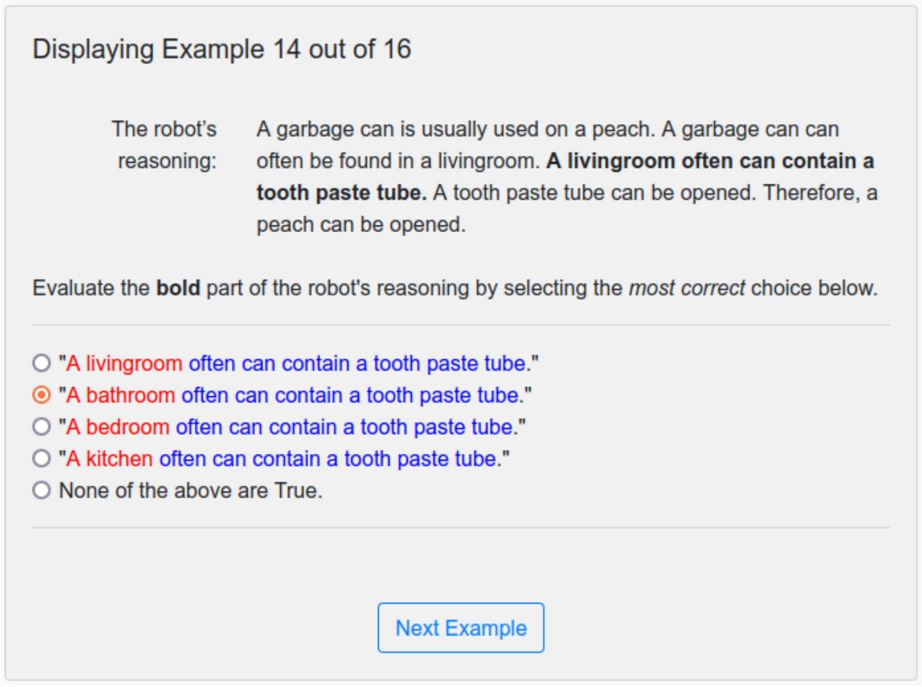
7.3.3 Validation of Explanations for Downstream Tasks

In our final experiment we evaluated whether non-experts could use our inference reconciliation framework to improve robot task performance. The robot’s task was *robust task execution*, wherein the robot is provided a demonstrated task-plan recorded in a demonstration environment, and asked to generalize the task-plan to new execution environments. Given that robots cannot be assumed to be error-free, we relaxed the assumption from prior experiments that the robot had gathered a high-quality dataset by including robot perception noise (i.e. the robot had a faulty object and affordance detector). We incorporated robot perception noise into a new dataset $\hat{\mathcal{D}}$ by randomly corrupting 30% of facts for each relation in the dataset $\mathcal{D}$ from prior experiments (Table 7.2). As a consequence, the MRE $\hat{\Theta}$ learned from the $\hat{\mathcal{D}}$ dataset often makes nonsensical knowledge inferences (e.g., (sponge, ObjUsedTo, microwave)) that lead to erratic robot behavior when executing tasks. We validated whether nonsensical facts supporting inferences made by $\hat{\Theta}$ can be revealed to non-experts using explanations generated by our inference reconciliation framework. We hypothesized that if non-experts can accurately recognize and correct nonsensical facts in explanations, then that feedback can be used to improve the MRE $\hat{\Theta}$, and in turn, improve robot generalization behaviors.

We performed a user study to measure how well non-experts can correct nonsensical facts within natural language explanations (i.e. correction accuracy). The explanations were generated by our inference reconciliation framework. Specifically, the student model Φ was trained on $\hat{\mathcal{D}}$ for false positive and false negative classifications of facts in $\mathcal{D}_{Te}$ where the embedding $\hat{\Theta}$ and student model Φ provided the same classification (see section 7.2). The results of the user study informed a confidence interval for the expected non-expert correction accuracy.

Study Design

We recruited 19 participants from AMT, all of whom were 18 years or older (M=33.4, SD=5.9). Of the 19 participants, 1 was filtered out for performing below chance on the practice portion of the study. In the study, users were tasked with identifying and correcting nonsensical facts for a series of knowledge inferences and accompanied explanations. Specifically users traversed through each supporting fact of the explanation and selected the “most correct” fact from a list of options, which included the supporting fact from the explanation, three other facts classified as most true by the MRE that could replace the


Displaying Example 14 out of 16

The robot's reasoning: A garbage can is usually used on a peach. A garbage can can often be found in a livingroom. **A livingroom often can contain a tooth paste tube.** A tooth paste tube can be opened. Therefore, a peach can be opened.

Evaluate the **bold** part of the robot's reasoning by selecting the *most correct* choice below.

○ "A livingroom often can contain a tooth paste tube."
[✓ "A bathroom often can contain a tooth paste tube."]
○ "A bedroom often can contain a tooth paste tube."
○ "A kitchen often can contain a tooth paste tube."
○ None of the above are True.

Figure 7.3: Non-expert Feedback User Study GUI.

supporting fact from the explanation, and "None of the above". An example question is presented in Figure 7.3.

We formed a confidence interval for the expected non-expert correction accuracy because exhaustively recruiting users to evaluate all grounded explanations generated for misclassified examples in $\hat{\mathcal{D}}_{Te}$ was impractical due to the large number of grounded explanations (10,000+). Instead, we randomly assigned each user 16 questions sampled from the large set of grounded explanations. In total, 18 users answered 96 sampled questions. We ensured that each sampled question received three responses (randomly sampled), and combined these responses using majority voting to create 6 meta-users. We determined the necessary population size of meta-users to be 6, by forming a 95% confidence interval with a 5% error margin using the sample standard-deviation of meta-user correction accuracy.

Study Results

The mean non-expert correction accuracy was 86.6% with an error margin of 4.1%. We confirmed non-expert correction accuracy samples were normally distributed using Shapiro-

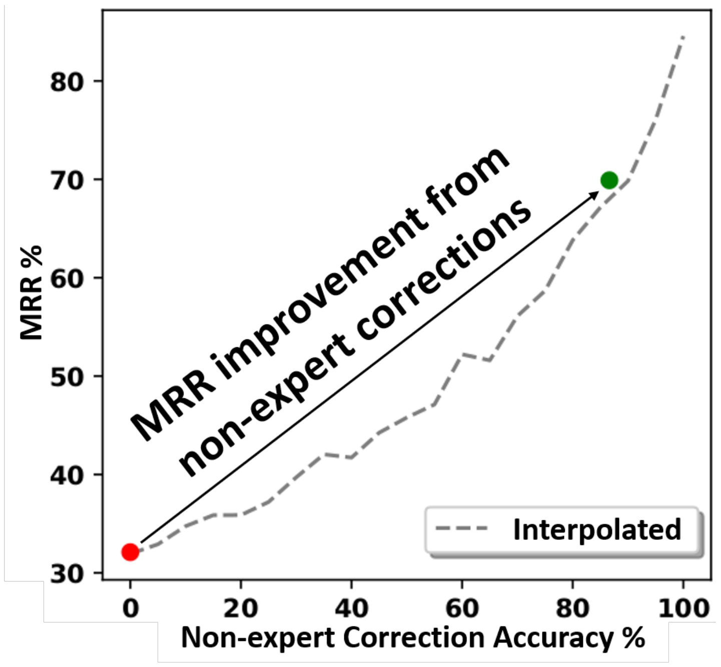


Figure 7.4: Effect of Non-expert Correction Accuracy on MRR%.

Wilk's Test (p=0.83). Additionally, to ensure no grouping bias, we performed a one-way ANOVA to test that there were no significant differences in the mean performances of users whose combined responses formed each meta-user (p=0.72). We used the mean non-expert correction accuracy to estimate the improvements in performance of downstream tasks like MRE link-prediction and robot task execution success rate.

Improvement of MRE

We characterized the effect of non-expert correction accuracy on improving the MRE by measuring inference performance. Our initial MRE was $\hat{\Theta}$ learned from $\hat{\mathcal{D}}$. We corrected 86.6% of the corrupted facts in $\hat{\mathcal{D}}$ to form an improved dataset $\bar{\mathcal{D}}$. We learned a new MRE, $\bar{\Theta}$, from $\bar{\mathcal{D}}$. Finally, we measured the difference in inference performance between $\hat{\Theta}$ and $\bar{\Theta}$ on our original dataset $\mathcal{D}$ (from Table 7.2) for the common MRE task, link-prediction [26].

In short, the link-prediction evaluation task is to rank complete triples from incomplete ones in test splits, i.e., rank heads h given (r, t) or tails t given (h, r). To perform link-prediction, each test triple (h, r, t) is first corrupted by replacing the head (or tail) entity with every other possible entity $\mathcal{E}$. Then all corrupted triples that represent a valid relationship between the corresponding entities are removed to avoid underestimating the embedding performance, known as "filtered setting" [26]. Last, scores are computed for each test triplet and its (remaining) corrupted triplets using the scoring function $f(h, r, t)$, then ranked in order of belief. For each test triple (h, r, t), the mean reciprocal rank of the test triple is calculated as a measure of inference performance.

We observed that the mean non-expert correction accuracy provided a 117% relative improvement in MRR between $\hat{\Theta}$ and $\bar{\Theta}$ for the link-prediction task. Shown in Figure 7.4 is the MRR of $\hat{\Theta}$ as a red dot (32.1%) and $\bar{\Theta}$ as a green dot (69.9%). Figure 7.4 also includes interpolated results for the range of possible non-expert correction accuracies. We believe the increasing slope as non-expert correction accuracy improves is due to the improvements in MRE generalization. Specifically, as the number of accurate facts the MRE trains on increases, the more rapidly its learned relationships generalize to unseen facts resulting in larger improvements in MRR%.

Improvement of Robot Behavior

As a final result, we characterized the effect of non-expert correction accuracy on improving the robot's behavior by measuring robot success rate. We considered a real-world scenario in which a household robot performs robust execution of household cleaning tasks, using the system and definitions in [160]. In robust task execution, the robot is provided an execution environment E_x and a demonstrated task plan T_d that is recorded in a demonstration environment E_d, and asked to find a modified task plan T_x, executable in E_x, that accomplishes the goal(s) of T_d. Task plans are defined as a sequence of primitive actions, where each primitive action may or may not be parameterized by objects. Task plans are incrementally modified using knowledge inferences from a MRE with the system defined in [160] to find an executable task plan. Execution environments are sampled by perturbing

Table 7.5: Incorrect Robot Generalization Behaviors Corrected by Non-experts

Action	Objects robot attempted to use to perform action
dust	computer, radio
wipe	crayon
disinfect	toaster, television, candle, pillow

the object used to demonstrate the task; changing the object's location, type, or both. The perturbations are made in accordance with the action-object-location distributions present in $\mathcal{D}$, ensuring that objects are not placed at implausible locations (e.g., broom inside the toilet) and that the intended generalization is not unreasonable (e.g. cleaning a table with a washing-machine). As a result, the demonstrated task plan often fails due to unsatisfied pre-conditions of primitive actions, and the robot must generalize the demonstrated task plan to formulate the *executable* task plan.

We measured the improvement in robot success rate provided by the mean non-expert correction accuracy by deploying $\hat{\Theta}$ and $\bar{\Theta}$ on a simulated robot in a simulation household environment performing robust execution of cleaning tasks. We sampled 50 initial demonstrations with 10 executions environments in each, for a total of 500 robot executions and measured the robot's execution success rate when using $\hat{\Theta}$ and $\bar{\Theta}$. Our experiment showed a 33.7% relative improvement in the robot's success rate due to the non-expert feedback (i.e. the robot succeeded in 187 sampled environments using $\hat{\Theta}$ and 250 using $\bar{\Theta}$). We measured the success rate of a robot that only repeats the default demonstration and a robot selecting random generalizations of the task plan as reference points for the difficulty of the task, which were 9 and 22 successes, respectively.

In addition to the quantitative improvement in success rate, we observed qualitative changes in the robot's generalization behaviors when performing robust task execution. We provide several examples from the 500 sampled executions in Table 7.5. Each combination of action and object in Table 7.5 is an instance of a generalization behavior the robot used to attempt when using $\hat{\Theta}$ but was corrected and no longer occurs when using $\bar{\Theta}$. For example, when using the initial MRE $\hat{\Theta}$, the robot attempted to dust using a radio and a computer. However, by using improved MRE $\bar{\Theta}$, which incorporates the non-expert feedback, those nonsensical task generalizations no longer are attempted.

7.4 Discussion & Conclusion

In summary, we introduce an inference reconciliation framework that answers a user's questions about knowledge inferences supporting robot actions using KGs, XAI, and natural language. Our framework follows a pedagogical XAI approach, by using an interpetable graph feature model to locally approximate the classifications of a black-box model, a MRE. Through a three-fold evaluation, we demonstrate the importance of our framework both with respect to interpretability as well as improved task performance. Specifically, we show via an algorithmic evaluation that leveraging a decision tree classifier as an interpretable graph feature model in our framework leads to higher F1-Fidelity compared to prior use of linear regression models for explainable MREs. Additionally, through user evaluations, we demonstrate that our explanations are preferred and accessible. Through a user preference evaluation, we demonstrate a significant preference towards our framework's explanations for knowledge inference "why questions." Additionally, when relaxing the assumption that robots are error-free, we showcase the effectiveness of our explanations in helping users identify and correct nonsensical beliefs in robots' knowledge representations, consequently improving robot task performance.

Active learning over the set of explanations to confirm with non-experts is a likely candidate for future work to reduce the amount of non-expert feedback needed. Additionally, within the broader area of XAIP, most works develop a single explanation method to answer a subset of possible end user questions. A system that understands end user questions and arbitrates between different explanation types accounting for dialogue context between the end user and robot remains an open, non-trivial problem. As the number of answerable end user questions types and robot explanation types presented in prior work reaches critical mass, such a system would seem more necessary as no single explanation type will best answer all possible end user question types.

7.5 Findings & Contributions

This work comprises our contributions to the **the development and evaluation of an inference reconciliation framework for MREs**. Our contributions in this chapter provide natural language explanations for MRE inferences, and therefore, improve the explainability of the autonomous system presented in chapter 5. We did this by leveraging interpretable graph feature models and using a pedagogical approach to explain facts inferred by a MRE. We showed the our graph feature model outperformed prior work by statistically significant margins, substantiating our design choices. Additionally, our explanations are preferred by non-expert end users for the question types we targeted. Finally, by providing interpretable explanations of MRE inferences, we enable non-experts to improve a robot's task generalization behaviors. Our novel inference reconciliation framework highlights the mutual benefits from promoting a symbiotic relationship between an autonomous robot and non-expert end user.

CHAPTER 8
CONCLUSIONS AND FUTURE WORK

Semantic reasoning about domain knowledge has enabled robots to be robust in novel or ambiguous situations. KG representations are a critical data structure enabling semantic reasoning on robot systems. Each KG representation makes assumptions and defines mathematical structures that provide a robot with a core set of reasoning abilities. Throughout this dissertation, we have examined a new class of KG representations for semantic reasoning on robot systems based on MREs. Specifically, this thesis **examines the use of multi-relational embeddings as knowledge graph representations within the context of robust task execution and develops methods to explain the inferences of and sequentially train multi-relational embeddings**.

8.1 Summary of Contributions

This dissertation has made the following contributions to validate this claim:

8.1.1 Survey of KG Representations in Robotics

We contributed a survey of prior KG representations used to perform semantic reasoning in robot systems. In addition to a comprehensive list of prior work, we provide broad performance characteristics of KG representations that directly affect common features desired of robots evident across the referenced works. These performance characteristics help to compare all surveyed KG representations with widely varying assumptions and mathematical structures.

8.1.2 Multi-Relational Embeddings as KG Representations for Robots

We contributed a novel KG representation for semantic reasoning in robot systems based on MREs. Our experiments showed that our KG representation significantly outperformed word embeddings at fact-prediction while being robust to large reductions in training data and domain transfer. Additionally, our KG representation used orders of magnitude less memory than BLNs to represent the same KG. While MREs do have limitations detailed in section 4.6, we believe the collectively distinct set of advantages of MREs from prior KG representations shows the potential to further progress of semantic reasoning on robot systems modeling large problem domains.

8.1.3 Robust Task Execution using KG Representations

We contributed a complete robot architecture that enabled robot task execution and planning to be informed by our KG representation. In simulation, we demonstrated that our KG representation significantly outperformed several baselines at robot one-shot task execution. Our experiments showed that our KG representation generalized learned domain knowledge beyond the set of observed facts and outperformed word embedding and plan network baselines. Additionally, our experiments showed that MLNs had intractable inference time when representing the same domain knowledge used in our experiments. We followed these experiments in simulation by validating the robot architecture on a mobile manipulator. Of the 50 trails, the robot generalized 38 task plans (including robot failures at the executive and behavioral levels), 10 (26.8%) of which were enabled by inferring unknown facts using the MRE.

8.1.4 Continual Knowledge Graph Embedding

We contributed five continual KG embedding methods that enable our KG representation to be sequentially trained with new domain knowledge. We evaluated each method across several datasets using a variety of continual learning metrics and provided guidance on which to select given the constraints of a robotics application. Specfically, in unconstrained scenarios, iterative batch learning performed best; in data constrained scenarios, our deep generative replay approach performed best; in data and progressively more time constrained scenarios, our regularization approaches performed best. Additionally, by using these methods in the task generalization module from chapter 5, we showed how our KG representation can be made more adaptable to sequentially train the MRE. Our simulations experiments demonstrated that our CKGE methods enable a robot operating in multiple environments with varying prior distributions (e.g., different houses, kitchen to bathroom) to incrementally include new facts observed without corrupting previously learned domain knowledge.

8.1.5 Explainable Knowledge Graph Embedding

We contributed a novel inference reconciliation framework and showcased a novel application of explanations within XAIP. Our inference reconciliation framework provides natural language explanations to non-experts about why inferences within our KG representation are classified as true, improving the interpretability of a robot's decision-making informed by a KG representation. We developed a novel graph feature model to extract explanations about knowledge inferences made by MREs. Our experiments showed that our graph feature model outperformed prior work by statistically significant margins at an evaluation task proposed in prior work, substantiating our design choices. Additionally, our users studies showed that our explanations are preferred by non-expert end users for the question types we targeted. Finally, by providing interpretable explanations of MRE inferences, our last experiments showed that we could enable non-experts to improve a robot's task generalization behaviors.

8.2 Open Questions

While the contributions of this dissertation have explored the use of MREs as KG representations from multiple angles, there remain many open questions at various levels of scope for the problem considered during this dissertation: semantic reasoning using MREs as KG representations supporting robots. The following subsections provide open questions at varying levels of scope for this problem.

8.2.1 Semantic Reasoning using MREs as KG Representations Supporting Robots

We begin with open questions at the same problem scope of this dissertation, pointing out limitations and future work for methods presented throughout this thesis.

Modeling uncertain KGs with MREs: Our proposed KG representation leveraged a variety of MREs that modeled KGs with binary labels for individual triples. We showed empirically that we could rank facts with binary labels by training over datasets with repeated triples and interpreting output scores as measures of confidence. However, the KGs modeled in robotics are often non-binary (e.g., cups have different locations with varying confidences). Recent research in MREs has begun developing methods for "uncertain KGs" where each fact is accompanied with a confidence score [161, 162]. Incorporating these methods and assumptions into the proposed KG representation is one avenue for future research.

Extending link-prediction conditioning in MREs: Our proposed KG representation is also limited in the variables that can be provided as evidence to condition a fact-prediction (see section 4.6). For example, assuming a set of possible locations for cups, the ranks of *dirty cup* locations should differ from the ranks of *clean cup* locations. How to make such conditioning possible in MREs remains an open question, although some recent research has proposed initial methods that combine multiple queries [10].

Reasoning about which explanation to provide: Our explanation mechanism produces all possible grounded explanations of a knowledge inference made using a MRE. Reasoning about which explanation to confirm with a non-expert user was out of scope for the project. However, such reasoning is necessary for practical application of our method due to the large number of grounded explanations. Additionally, within the broader area of XAIP, most works develop a single explanation method to answer a subset of possible end user questions. A system that understands end user questions and arbitrates between different explanation types accounting for dialogue context between the end user and robot remains an open, non-trivial problem. As the number of answerable end user questions types and robot explanation types presented in prior work reaches critical mass, that open question will gain priority as no single explanation type will best answer all possible end user question types.

8.2.2 Semantic Reasoning using KG Representations Supporting Robots

Other open questions at a broader problem scope are related to improving KG representations for robot systems.

Improving KG representation integration in a robot architecture: Our task generalization module in chapter 5 presents one method of integrating the KG representation to inform robot task planning and execution. The integration primarily informs the selection of which primitive actions to sequence (i.e. where to look for objects, which objects can be used for a task). However, there may be other opportunities at different levels of the robot architecture that could improve in performance if informed by the KG representation. Additionally, there may be better methods of integrating the KG representation with the robot's planning layer.

Novel KG representations: As discussed in section 3.10, although many KG representations have been used for robot systems, none excel in all the mentioned properties that affect the performance characteristics desired of robots. An open question for future research is what novel KG representations can be developed for use on robot systems that would offer a collective set of advantages that improve over prior KG representations.

Combining KG representations: With a similar goal to the previous open question, one might also consider how to combine existing complimentary KG representations. Therefore, another open question for future research is what complementary KG representations can be combined for use on robot systems that would offer a collective set of advantages that improve over prior KG representations. One such example that exists is First-order probabilistic models, but there may exist better combinations of existing KG representations.

8.2.3 Semantic Reasoning Supporting Robots

At the broadest problem scope, we consider open research areas with potentially many research questions related to improving semantic reasoning for robotics.

Multi-modal semantic reasoning for sensing and acting: An area of emerging interest in robotics is finding a bridge spanning the low-level, high dimensional data associated with robot actuation and sensing and the high-level symbols describing actions and world state at various levels of abstraction. By learning semantic representations that capture the essence and function of actions with respect to the world state, such an interface could enable non-expert general purpose use of robot systems through dialogue and natural language.

Transparent semantic reasoning for interaction: As the former research area develops, providing non-expert interpretable explanations of robot behaviors will grow in importance. Such capabilities will be needed to confirm interpretations of end user requests, justify generalizations of learned behaviors in new scenarios, and communicate represented knowledge driving decision making.

Appendices

APPENDIX A
EXTENDED CKGE EXPERIMENTS WITH ALTERNATE SAMPLING STRATEGIES

In addition to our sampling strategy in section 6.4, referred to here as *triple* sampling, we also experimented with entity and relation sampling. We experimented with triple, relation, and entity sampling strategies to get a breadth of sampling strategies one could use to generate a CKGE dataset from an established benchmark KG. For our main results, we chose triple sampling because it had the most fidelity to the original benchmark KGs, while still modeling a challenging sampling scenario a robot might encounter. By providing results from experiments with all sampling strategies, we show that the insights for our main results hold for different sampling strategies and that our main narrative avoids highlighting outlier results. Below we detail how the entity and relation sampling strategies were implemented, the CKGE datasets generated by each, and results obtained for these datasets using experimental settings similar to section 6.4.

CKGE sampling strategies: Our entity sampling strategy closely follows the sampling strategy proposed in [54]. As in section 6.4, consider a knowledge graph $\mathcal{G}$ whose triples $\mathcal{D}$ have been split into a training set $\mathcal{D}_{Tr}$, validation set $\mathcal{D}_{Va}$, and test set $\mathcal{D}_{Te}$. Our approach for generating entity sampling datasets for $n = \{1, ..., N\}$ learning sessions is:

1. *Initialize entity sampling distribution*: initialize the entity sampling distribution to uniform likelihood for any entity to be sampled.

2. *Sample entities*: sample without replacement 50% of entities in $\mathcal{E}$ of $\mathcal{G}$.

3. *Extract relations*: create a set of entities E^n and a set of relations R^n for this session from the sampled entities and the relations of triples in $\mathcal{G}$ that connect any two entities in E^n, respectively. The set of all observed entities (relations), i.e., $\mathcal{E}^n$ ($\mathcal{R}^n$) is the union of

current and prior E^n (R^n).

4. *Construct n^{th} train, validation, and test sets*: extract from $\mathcal{D}_{Tr}$, $\mathcal{D}_{Va}$, and $\mathcal{D}_{Te}$ the triples whose head, relation, and tail belong to E^n and R^n (respectively). These triples form train set $\mathcal{D}^n_{Tr}$, validation set $\mathcal{D}^n_{Va}$ and test set $\mathcal{D}^n_{Te}$ of the n^{th} session.

5. *Update entity distribution and repeat*: to bias towards sampling new triples, decrease the likelihoods of sampled entities in E^n to be sampled in future learning sessions in proportion to the number of times an entity has been sampled.

6. Repeat steps 2-5 until a predefined number of iterations are completed.

We generated two CKGE datasets with $n = 5$ sessions using the entity sampling approach on two established benchmark knowledge graphs in the graph embedding community (WN18RR and FB15K237 [150]). Table A.1 reports statistics of each dataset as in section 6.4. Note that even after 5 learning sessions, not all training, validation, nor test triples have been sampled from the benchmark datasets. Additionally, the number of triples in each learning session exceeds that of triple sampling, proving how this method often repeats sampled triples across learning sessions.

Due to the large number of triples or entities sampled in the first learning session by both triple and entity sampling strategies, fewer than 5% of all future relations sampled will be new, as shown across Table 6.1 and Table A.1. Therefore, the third and final sampling strategy we experimented with was relation sampling to encourage sampling new relations in later learning sessions. Again, consider a knowledge graph $\mathcal{G}$ whose triples $\mathcal{D}$ have been split into a training set $\mathcal{D}_{Tr}$, validation set $\mathcal{D}_{Va}$, and test set $\mathcal{D}_{Te}$. Our approach for generating relation sampling datasets for $n = \{1, ..., N\}$ learning sessions is:

1. *Initialize relation sampling distribution*: initialize the relation sampling distribution to uniform likelihood for any relation to be sampled.

2. *Sample relations*: sample without replacement 50% of relations in $\mathcal{R}$ of $\mathcal{G}$.

Table A.1: CKGE Datasets; Entity Sampling Benchmarks

	WN18RR-5-LS				
	LS-1	LS-2	LS-3	LS-4	LS-5
$\|E^n\|$	20,471/(50%)	20,471/(81%)	20,471/(95%)	20,471/(99%)	20,471/(100%)
$\|R^n\|$	11/(100%)	11/(100%)	11/(100%)	11/(100%)	11/(100%)
$\|\mathcal{D}^n_{Tr}\|$	21,729/(25%)	21,937/(47%)	21,888/(63%)	20,870/(74%)	21,852/(83%)
$\|\mathcal{D}^n_{Va}\|$	773/(25%)	774/(48%)	796/(64%)	727/(75%)	801/(84%)
$\|\mathcal{D}^n_{Te}\|$	821/(26%)	800/(48%)	798/(64%)	729/(75%)	797/(83%)
	FB15K237-5-LS				
	LS-1	LS-2	LS-3	LS-4	LS-5
$\|E^n\|$	7,270/(50%)	7,270/(81%)	7,270/(95%)	7,270/(99%)	7,270/(100%)
$\|R^n\|$	230/(97%)	231/(99%)	226/(100%)	231/(100%)	230/(100%)
$\|\mathcal{D}^n_{Tr}\|$	73,846/(27%)	69,288/(48%)	66,326/(64%)	70,519/(75%)	73,565/(84%)
$\|\mathcal{D}^n_{Va}\|$	4,683/(27%)	4,616/(49%)	4,365/(65%)	4,411/(76%)	4,932/(84%)

Table A.2: CKGE Datasets; Relation Sampling Benchmarks

	WN18RR-5-LS				
	LS-1	LS-2	LS-3	LS-4	LS-5
$\|E^n\|$	16,564/(40%)	12,447/(47%)	35,670/(96%)	37,548/(99%)	14,062/(99%)
$\|R^n\|$	5/(45%)	5/(82%)	5/(91%)	5/(100%)	5/(100%)
$\|\mathcal{D}^n_{Tr}\|$	16,886/(19%)	12,840/(26%)	37,566/(66%)	69,555/(100%)	14,442/(100%)
$\|\mathcal{D}^n_{Va}\|$	216/(7%)	201/(11%)	490/(25%)	1,979/(75%)	184/(76%)
$\|\mathcal{D}^n_{Te}\|$	205/(7%)	228/(11%)	511/(26%)	2053/(75%)	185/(76%)
	FB15K237-5-LS				
	LS-1	LS-2	LS-3	LS-4	LS-5
$\|E^n\|$	13,666/(94%)	13,522/(98%)	13,704/(100%)	13,023/(100%)	13,989/(100%)
$\|R^n\|$	118/(50%)	118/(80%)	118/(95%)	118/(99%)	118/(100%)
$\|\mathcal{D}^n_{Tr}\|$	128,045/(47%)	148,446/(78%)	154,624/(98%)	113,904/(100%)	155,556/(100%)
$\|\mathcal{D}^n_{Va}\|$	7,546/(43%)	9,814/(78%)	9,474/(98%)	6,394/(99%)	11,184/(99%)

3. *Extract entities*: create a set of relations R^n and a set of entities E^n for this session from the sampled relations and the entities of triples in $\mathcal{G}$ that are connected any relation in R^n, respectively. The set of all observed entities (relations), i.e., $\mathcal{E}^n$ ($\mathcal{R}^n$) is the union of current and prior E^n (R^n).

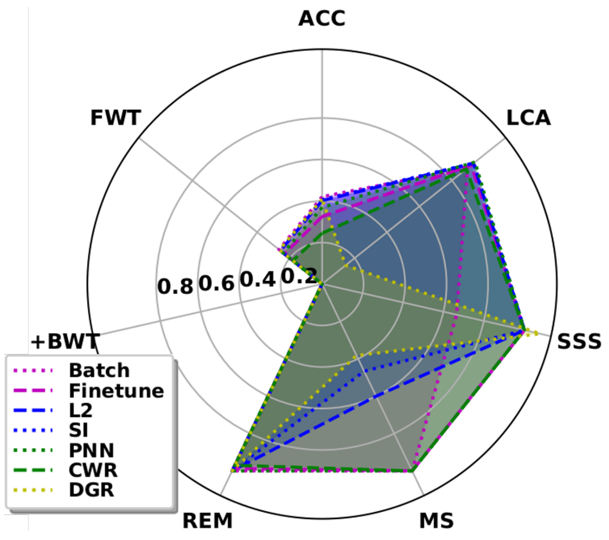
4. *Construct n^{th} train, validation, and test sets*: extract from $\mathcal{D}_{Tr}$, $\mathcal{D}_{Va}$, and $\mathcal{D}_{Te}$ the triples whose head, relation, and tail belong to E^n and R^n (respectively). These triples form train set $\mathcal{D}^n_{Tr}$, validation set $\mathcal{D}^n_{Va}$ and test set $\mathcal{D}^n_{Te}$ of the n^{th} session.

5. *Update relation distribution and repeat*: to bias towards sampling new triples, decrease the likelihoods of sampled relations in R^n to be sampled in future learning sessions in proportion to the number of times an relation has been sampled.

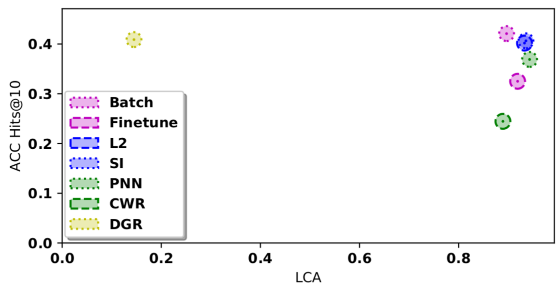
6. Repeat steps 2-5 until a predefined number of iterations are completed.

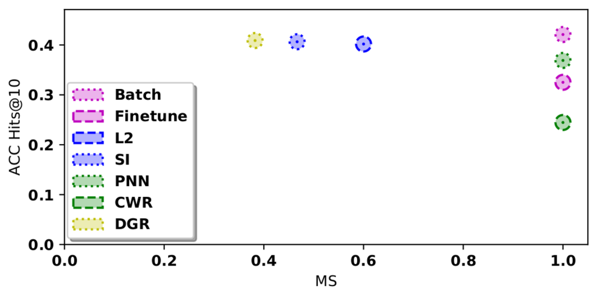
We generated two CKGE datasets with $n = 5$ sessions using the relation sampling approach on two established benchmark knowledge graphs in the graph embedding community (WN18RR and FB15K237 [150]). Table A.2 reports statistics of each dataset as in section 6.4. While relation sampling uniquely has more new relations in later learning sessions, the strategy does not model a realistic world scenario a robot might face when exploring an environment.

Our results obtained for these datasets support the results presented in section 6.5 and section 6.7, with some minor differences depending on the dataset or embedding method being used. Similar to section 6.5, results reported below are the average of five test runs in each experimental scenario; statistical significance is tested using repeated-measures ANOVA and a post-hoc Tukey's test. Any mention of 'significance' implies statistical significance at 95% significance level (i.e. $p < 0.05$). As in section 6.5, *Batch* and *Finetune* baselines are reported as upper and lower bounds for the expected inference performance of the CKGE methods.



(a)



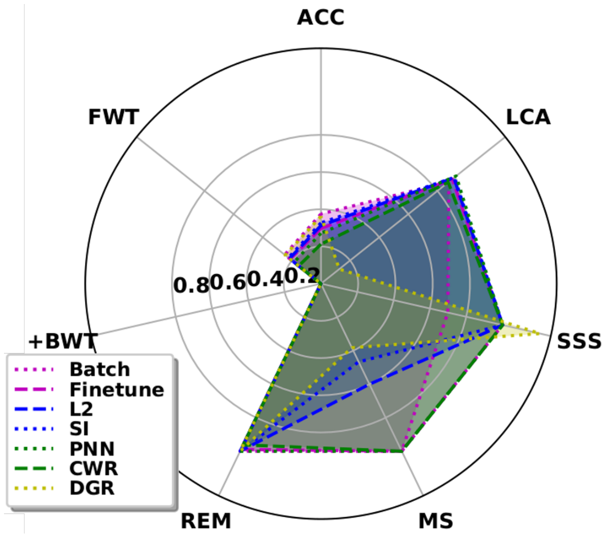


(b) (c)

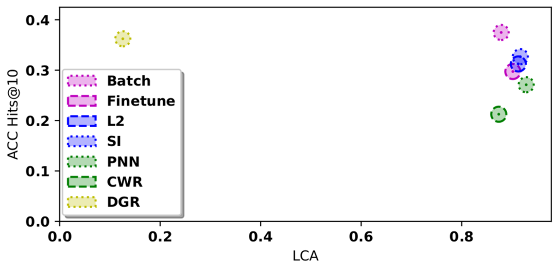
Figure A.1: Measures averaged for all datasets in Table A.1 and graph embedding representations in section 6.4. Hits@10 used for ACC, FWT, +BWT, and REM. Best viewed in color.

Entity sampling evaluations: Figure A.1a summarizes the results of experiments using entity sampling knowledge graph datasets of Table A.1 (WN18RR, FB15K237), where the range of each measure is $[0, 1]$ and larger values are better. Although DGR significantly outperforms other methods in terms of inference (i.e., using ACC), there are insights and trade-offs to consider based on other factors.

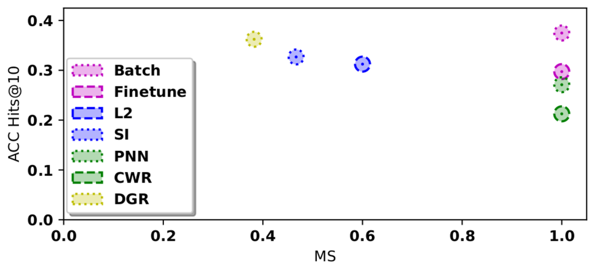
- Figure A.1b shows that DGR has a significantly lower learning speed (based on LCA) than the other methods since a new generative model must be trained in each learning session. PNN had the best learning speed, having significantly better learning speeds than Batch across all datasets and embedding methods. Following close behind were the regularization techniques (L2R and SI), which had significantly better learning speeds than Batch in all cases except when using the Analogy embedding method for WN18RR.



(a)



(b)



(c)

Figure A.2: Measures averaged for all datasets in Table A.2 and graph embedding representations in section 6.4. Hits@10 used for ACC, FWT, +BWT, and REM. Best viewed in color.

- Figure A.1c indicates that methods with good inference performance also tend to have higher model memory growth (i.e., MS measure); among the methods with significantly better inference performance than Finetune, PNN had the smallest MS followed by L2R, SI and DGR.
- PNN outperforms Finetune in this dataset shown in Figure A.1. The difference in PNN performance from triple to entity sampling is likely because more entities are frozen sooner in the triple sampling datasets with fewer observed training triples than in the entity sampling datasets, as evident in the entity coverage statistics for each dataset.
- Figure A.1 indicates that the CWR has significantly lower inference performance than Finetune in all experiments. CWR's poor inference performance is again attributable to the challenges of directly manipulating representations in the embedding space.

Relation sampling evaluations: Figure A.2a summarizes the results of experiments using relation sampling knowledge graph datasets of Table A.2 (WN18RR, FB15K237), where the range of each measure is $[0, 1]$ and larger values are better. Although DGR significantly outperforms other methods in terms of inference (i.e., using ACC and FWT) except for SI in the FB15K237 datasets, there are insights and trade-offs to consider based on other factors.

- Figure A.2b shows that DGR has a significantly lower learning speed (based on LCA) than the other methods since a new generative model must be trained in each learning session. PNN had the best learning speed, having significantly better learning speeds than Batch across all datasets and embedding methods. Following close behind were the regularization techniques (L2R and SI), which varied in significant improvements in LCA compared to Batch.
- Figure A.2c indicates that methods with good inference performance also tend to have higher model memory growth (i.e., MS measure); among the methods with significantly better inference performance than Finetune, L2R had the smallest MS followed by SI and DGR.
- Figure A.2 indicates that the CKGE methods based on architecture modification (i.e., PNN and CWR) have significantly lower inference performance than Finetune in all experiments. The difference in performance between PNN and the regularization-based methods shows the importance of flexibility over prior concepts for CKGE. Also, CWR's poor inference performance highlights the challenges of directly manipulating representatins in the embedding space because, although CWR can learn TE well in isolation, CE is quickly corrupted by the averaging performed to merge embeddings.

Conclusions: We detailed results obtained with all sampling strategies used for the benchmark knowledge graphs WN18RR and FB15K237. Our results from these extended experiments support the main results presented in chapter 6. Namely that, while DGR outperforms the other CKGE methods in general, there are trade-offs to consider in terms of learning speed and memory requirements depending on the robotics application.

REFERENCES

[1] M. Cakmak and L. Takayama, "Towards a comprehensive chore list for domestic robots," in *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, IEEE, 2013, pp. 93–94.

[2] X. Chen, S. Jia, and Y. Xiang, "A review: Knowledge reasoning over knowledge graph," *Expert Systems with Applications*, vol. 141, p. 112 948, 2020.

[3] M. Tenorth and M. Beetz, "Knowrob: A knowledge processing infrastructure for cognition-enabled robots," *The International Journal of Robotics Research*, vol. 32, no. 5, pp. 566–590, 2013.

[4] S. Chernova *et al.*, "Situated bayesian reasoning framework for robots operating in diverse everyday environments," in *Robotics Research*, Springer, 2020, pp. 353–369.

[5] A. Saxena, A. Jain, O. Sener, A. Jami, D. K. Misra, and H. S. Koppula, "Robobrain: Large-scale knowledge engine for robots," *arXiv preprint arXiv:1412.0691*, 2014.

[6] Y. Zhu, A. Fathi, and L. Fei-Fei, "Reasoning about object affordances in a knowledge base representation," in *European conference on computer vision*, Springer, 2014, pp. 408–424.

[7] H. Celikkanat, G. Orhan, and S. Kalkan, "A probabilistic concept web on a humanoid robot," *IEEE Transactions on Autonomous Mental Development*, vol. 7, no. 2, pp. 92–106, 2015.

[8] M. Beetz, D. Beßler, A. Haidu, M. Pomarlan, A. K. Bozcuoğlu, and G. Bartels, "Know rob 2.0—a 2nd generation knowledge processing framework for cognition-enabled robotic agents," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 512–519.

[9] S. Lemaignan, R. Ros, L. Mösenlechner, R. Alami, and M. Beetz, "Oro, a knowledge management platform for cognitive architectures in robotics," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, IEEE, 2010, pp. 3548–3553.

[10] W. Liu, D. Bansal, A. Daruna, and S. Chernova, "Learning Instance-Level N-Ary Semantic Knowledge At Scale For Robots Operating in Everyday Environments," in *Proceedings of Robotics: Science and Systems*, Virtual, Jul. 2021.

[11] J. Arkin *et al.*, “Multimodal estimation and communication of latent semantic knowledge for robust execution of robot instructions,” *The International Journal of Robotics Research*, p. 0 278 364 920 917 755, 2020.

[12] J. Thomason, J. Sinapov, R. J. Mooney, and P. Stone, “Guiding exploratory behaviors for multi-modal grounding of linguistic descriptions,” in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[13] D. Nyga *et al.*, “Grounding robot plans from natural language instructions with incomplete world knowledge,” in *Conference on Robot Learning*, 2018, pp. 714–723.

[14] M. Tenorth and M. Beetz, “Knowrob—knowledge processing for autonomous personal robots,” in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2009, pp. 4261–4266.

[15] M. Tenorth, L. Kunze, D. Jain, and M. Beetz, “Knowrob-map-knowledge-linked semantic object maps,” in *2010 10th IEEE-RAS International Conference on Humanoid Robots*, IEEE, 2010, pp. 430–435.

[16] A. Boteanu, A. St. Clair, A. Mohseni-Kabir, C. Saldanha, and S. Chernova, “Leveraging large-scale semantic networks for adaptive robot task learning and execution,” *Big data*, vol. 4, no. 4, pp. 217–235, 2016.

[17] A. Pronobis and P. Jensfelt, “Large-scale semantic mapping and reasoning with heterogeneous modalities,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, 2012, pp. 3515–3522.

[18] N. Fulda, N. Tibbetts, Z. Brown, and D. Wingate, “Harvesting common-sense navigational knowledge for robotics from uncurated text corpora,” in *Conference on Robot Learning*, 2017, pp. 525–534.

[19] Z. Wang, J. Zhang, J. Feng, and Z. Chen, “Knowledge graph embedding by translating on hyperplanes.,” in *Aaai*, vol. 14, 2014, pp. 1112–1119.

[20] R. Gupta, M. J. Kochenderfer, D. Mcguinness, and G. Ferguson, “Common sense data acquisition for indoor mobile robots,” in *AAAI*, 2004, pp. 605–610.

[21] F. Mahdisoltani, J. Biega, and F. Suchanek, “Yago3: A knowledge base from multilingual wikipedias,” in *7th biennial conference on innovative data systems research*, CIDR Conference, 2014.

[22] G. A. Miller, “Wordnet: A lexical database for english,” *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[23] W. Liu, A. Daruna, and S. Chernova, "A survey of semantic reasoning frameworks for robotic systems," *Robotics and Autonomous Systems*, In preparation.

[24] M. Hanheide *et al.*, "Robot task planning and explanation in open and uncertain worlds," *Artificial Intelligence*, vol. 247, pp. 119–150, 2017.

[25] M. Waibel *et al.*, "Roboearth," *IEEE Robotics & Automation Magazine*, vol. 18, no. 2, pp. 69–82, 2011.

[26] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 11–33, 2015.

[27] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[28] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 11–33, 2016.

[29] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, 2013, pp. 2787–2795.

[30] H. Liu, Y. Wu, and Y. Yang, "Analogical inference for multi-relational embeddings," *arXiv preprint arXiv:1705.02426*, 2017.

[31] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.

[32] I. Balažević, C. Allen, and T. M. Hospedales, "Tucker: Tensor factorization for knowledge graph completion," *arXiv preprint arXiv:1901.09590*, 2019.

[33] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, 2013, pp. 2787–2795.

[34] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *International conference on machine learning*, PMLR, 2016, pp. 2071–2080.

[35] I. Chami, A. Wolf, D.-C. Juan, F. Sala, S. Ravi, and C. Ré, "Low-dimensional hyperbolic knowledge graph embeddings," *arXiv preprint arXiv:2005.00545*, 2020.

[36] Z. Zhang, J. Cai, Y. Zhang, and J. Wang, "Learning hierarchy-aware knowledge graph embeddings for link prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 3065–3072.

[37] Z. Zhang, J. Cai, and J. Wang, "Duality-induced regularizer for tensor factorization based knowledge graph completion," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 604–21 615, 2020.

[38] A. Bosselut, H. Rashkin, M. Sap, C. Malaviya, A. Celikyilmaz, and Y. Choi, "Comet: Commonsense transformers for automatic knowledge graph construction," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 4762–4779.

[39] L. Yao, C. Mao, and Y. Luo, "Kg-bert: Bert for knowledge graph completion," *arXiv preprint arXiv:1909.03193*, 2019.

[40] D. Li, M. Yi, and Y. He, "Lp-bert: Multi-task pre-training knowledge graph bert for link prediction," *arXiv preprint arXiv:2201.04843*, 2022.

[41] A. Vaswani *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[42] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[43] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[44] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data," in *Icml*, 2011.

[45] B. Wang, T. Shen, G. Long, T. Zhou, Y. Wang, and Y. Chang, "Structure-augmented text representation learning for efficient knowledge graph completion," in *Proceedings of the Web Conference 2021*, 2021, pp. 1737–1748.

[46] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, 2019.

[47] Y.-C. Hsu, Y.-C. Liu, and Z. Kira, "Re-evaluating continual learning scenarios: A categorization and case for strong baselines," *arXiv preprint arXiv:1810.12488*, 2018.

[48] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," *Proceedings of machine learning research*, vol. 70, p. 3987, 2017.

[49] J. Kirkpatrick *et al.*, “Overcoming catastrophic forgetting in neural networks,” *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[50] A. A. Rusu *et al.*, “Progressive neural networks,” *arXiv preprint arXiv:1606.04671*, 2016.

[51] V. Lomonaco and D. Maltoni, “Core50: A new dataset and benchmark for continuous object recognition,” *arXiv preprint arXiv:1705.03550*, 2017.

[52] H. Shin, J. K. Lee, J. Kim, and J. Kim, “Continual learning with deep generative replay,” in *Advances in Neural Information Processing Systems*, 2017, pp. 2990–2999.

[53] G. M. van de Ven and A. S. Tolias, “Generative replay with feedback connections as a general strategy for continual learning,” *arXiv preprint arXiv:1809.10635*, 2018.

[54] H.-J. Song and S.-B. Park, “Enriching translation-based knowledge graph embeddings through continual learning,” *IEEE Access*, vol. 6, pp. 60 489–60 497, 2018.

[55] F. Zenke, B. Poole, and S. Ganguli, “Continual learning through synaptic intelligence,” *Proceedings of machine learning research*, vol. 70, p. 3987, 2017.

[56] N. Díaz-Rodríguez, V. Lomonaco, D. Filliat, and D. Maltoni, “Don’t forget, there is more than forgetting: New metrics for continual learning,” *arXiv preprint arXiv:1810.13166*, 2018.

[57] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Díaz-Rodríguez, “Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges,” *Information Fusion*, vol. 58, pp. 52–68, 2020.

[58] T. Chakraborti, S. Sreedharan, and S. Kambhampati, “The emerging landscape of explainable ai planning and decision making,” *arXiv preprint arXiv:2002.11697*, 2020.

[59] B. Seegebarth, F. Müller, B. Schattenberg, and S. Biundo, “Making hybrid plans more clear to human users-a formal approach for generating sound explanations,” in *Int. Conf. on Automated Planning and Scheduling*, 2012.

[60] P. Bercher *et al.*, “Plan, repair, execute, explain-how planning helps to assemble your home theater.,” in *ICAPS*, 2014.

[61] G. Canal, S. Krivic, P. Luff, and A. Coles, “Task plan verbalizations with causal justifications,” in *ICAPS 2021 Workshop on Explainable AI Planning (XAIP)*, 2021.

[62] B. Krarup, M. Cashmore, D. Magazzeni, and T. Miller, "Model-based contrastive explanations for explainable planning," 2019.

[63] J. Hoffmann and D. Magazzeni, "Explainable ai planning (xaip): Overview and the case of contrastive explanation," *Reasoning Web. Explainable Artificial Intelligence*, pp. 277–282, 2019.

[64] O. Khan, P. Poupart, and J. Black, "Minimal sufficient explanations for factored markov decision processes," in *Int. Conf. on Automated Planning and Scheduling*, vol. 19, 2009, pp. 194–200.

[65] T. Dodson, N. Mattei, J. T. Guerin, and J. Goldsmith, "An english-language argumentation interface for explanation generation with markov decision processes in the domain of academic advising," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 3, no. 3, pp. 1–30, 2013.

[66] U. Ehsan, P. Tambwekar, L. Chan, B. Harrison, and M. O. Riedl, "Automated rationale generation: A technique for explainable ai and its effects on human perceptions," in *Proceedings of the 24th International Conference on Intelligent User Interfaces*, 2019, pp. 263–274.

[67] M. Göbelbecker, T. Keller, P. Eyerich, M. Brenner, and B. Nebel, "Coming up with good excuses: What to do when no plan can be found," in *Int. Conf. on Automated Planning and Scheduling*, 2010.

[68] J. Moon *et al.*, "Towards explanations of plan execution for human-robot teaming," CEUR Workshop Proceedings, 2019.

[69] D. Das and S. Chernova, "Semantic-based explainable ai: Leveraging semantic scene graphs and pairwise ranking to explain robot failures," in *Int. Conf. on Intelligent Robots and Systems*, IEEE, 2021.

[70] A. Ruschel, A. C. Gusmão, G. P. Polleti, and F. G. Cozman, "Explaining completions produced by embeddings of knowledge graphs," in *European Conference on Symbolic and Quantitative Approaches with Uncertainty*, Springer, 2019, pp. 324–335.

[71] Q. Xie, X. Ma, Z. Dai, and E. Hovy, "An interpretable knowledge transfer model for knowledge base completion," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 950–962.

[72] S. M. Kazemi and D. Poole, "Simple embedding for link prediction in knowledge graphs," *Advances in neural information processing systems*, vol. 31, 2018.

[73] B. Yang, S. W.-t. Yih, X. He, J. Gao, and L. Deng, “Embedding entities and relations for learning and inference in knowledge bases,” in *Int. Conf. on Learning Representations*, 2015.

[74] W. Zhang, B. Paudel, W. Zhang, A. Bernstein, and H. Chen, “Interaction embeddings for prediction and explanation in knowledge graphs,” in *Int. Conf. on Web Search and Data Mining*, 2019, pp. 96–104.

[75] Y. Nandwani, A. Gupta, A. Agrawal, M. S. Chauhan, P. Singla, *et al.*, “Oxkbc: Outcome explanation for factorization based knowledge base completion,” in *Automated Knowledge Base Construction*, 2020.

[76] M. Wise, M. Ferguson, D. King, E. Diehr, and D. Dymesich, “Fetch and freight: Standard platforms for service robot applications,” in *Workshop on autonomous mobile service robots*, 2016.

[77] S. Banerjee *et al.*, “Taking recoveries to task: Recovery-driven development for recipe-based robot tasks,” *arXiv preprint arXiv:2001.10386*, 2020.

[78] M. Quigley *et al.*, “Ros: An open-source robot operating system,” in *ICRA workshop on open source software*, Kobe, Japan, vol. 3, 2009, p. 5.

[79] R. B. Rusu and S. Cousins, “Point cloud library (pcl),” in *ICRA*, 2011, pp. 1–4.

[80] W. Wohlkinger and M. Vincze, “Ensemble of shape functions for 3d object classification,” in *Int. Conf. on Robotics and Biomimetics*, IEEE, 2011, pp. 2987–2992.

[81] S. Chitta, I. Sucan, and S. Cousins, “Moveit![ros topics],” *Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[82] A. ten Pas and R. Platt, “Using geometry to detect grasp poses in 3d point clouds,” in *Robotics Research*, Springer, 2018, pp. 307–324.

[83] D. Kent and R. Toris, “Adaptive autonomous grasp selection via pairwise ranking,” in *IROS*, IEEE, 2018, pp. 2971–2976.

[84] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz, “Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective,” in *HRI*, ACM, 2012, pp. 391–398.

[85] B. Moldovan, P. Moreno, M. van Otterlo, J. Santos-Victor, and L. De Raedt, “Learning relational affordance models for robots in multi-object manipulation tasks,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, 2012, pp. 4373–4378.

[86] D. Paulius, Y. Huang, R. Milton, W. D. Buchanan, J. Sam, and Y. Sun, "Functional object-oriented network for manipulation learning," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, IEEE, 2016, pp. 2655–2662.

[87] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," in *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, Association for Computational Linguistics, 1994, pp. 133–138.

[88] D. Paulius, A. B. Jelodar, and Y. Sun, "Functional object-oriented network: Construction & expansion," *arXiv preprint arXiv:1807.02189*, 2018.

[89] N. Abdo, C. Stachniss, L. Spinello, and W. Burgard, "Robot, organize my shelves! tidying up objects by predicting user preferences," in *2015 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2015, pp. 1557–1564.

[90] Y. Yang, A. Guha, C. Fermüller, and Y. Aloimonos, "Manipulation action tree bank: A knowledge resource for humanoids," in *2014 IEEE-RAS International Conference on Humanoid Robots*, IEEE, 2014, pp. 987–992.

[91] S. Vasudevan, S. Gächter, V. Nguyen, and R. Siegwart, "Cognitive maps for mobile robots-an object based approach," *Robot. Auton. Syst.*, vol. 55, no. 5, pp. 359–371, May 2007.

[92] J. Aleotti and S. Caselli, "Part-based robot grasp planning from human demonstration," in *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 4554–4560.

[93] A. Jain, B. Wojcik, T. Joachims, and A. Saxena, "Learning trajectory preferences for manipulators via iterative improvement," in *Advances in neural information processing systems*, 2013, pp. 575–583.

[94] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[95] B. Limketkai, L. Liao, and D. Fox, "Relational object maps for mobile robots," in *IJCAI*, 2005, pp. 1471–1476.

[96] M. Günther, J. Ruiz-Sarmiento, C. Galindo, J. Gonzalez-Jimenez, and J. Hertzberg, "Context-aware 3d object anchoring for mobile robots," *Robotics and Autonomous Systems*, vol. 110, pp. 12–32, 2018.

[97] D. Song, C. H. Ek, K. Huebner, and D. Kragic, "Task-based robot grasp planning using probabilistic inference," *IEEE transactions on robotics*, vol. 31, no. 3, pp. 546–561, 2015.

[98] R. Paul, A. Barbu, S. Felshin, B. Katz, and N. Roy, "Temporal grounding graphs for language understanding with accrued visual-linguistic context," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, AAAI Press, 2017, pp. 4506–4514.

[99] M. Kim and I. H. Suh, "Active object search in an unknown large-scale environment using commonsense knowledge and spatial relations," *Intelligent Service Robotics*, vol. 12, no. 4, pp. 371–380, 2019.

[100] S. Zhang and M. Sridharan, "A survey of knowledge-based sequential decision making under uncertainty," *arXiv preprint arXiv:2008.08548*, 2020.

[101] F. Baader, I. Horrocks, and U. Sattler, "Description logics," *Foundations of Artificial Intelligence*, vol. 3, pp. 135–179, 2008.

[102] I. Bratko, *Prolog programming for Artificial Intelligence*. Pearson Education Limited, 2012.

[103] M. Gelfond, *Answer sets. handbook of knowledge representation*, 2008.

[104] L. De Raedt and A. Kimmig, "Probabilistic (logic) programming concepts," *Machine Learning*, vol. 100, no. 1, pp. 5–47, 2015.

[105] L. De Raedt, A. Kimmig, and H. Toivonen, "Problog: A probabilistic prolog and its application in link discovery.," in *IJCAI*, Hyderabad, vol. 7, 2007, pp. 2462–2467.

[106] C. Baral, M. Gelfond, and N. Rushton, "Probabilistic reasoning with answer sets," in *International Conference on Logic Programming and Nonmonotonic Reasoning*, Springer, 2004, pp. 21–33.

[107] W. Hwang, J. Park, H. Suh, H. Kim, and I. H. Suh, "Ontology-based framework of robot context modeling and reasoning for object recognition," in *International Conference on Fuzzy Systems and Knowledge Discovery*, Springer, 2006, pp. 596–606.

[108] I. H. Suh, G. H. Lim, W. Hwang, H. Suh, J.-H. Choi, and Y.-T. Park, "Ontology-based multi-layered robot knowledge framework (omrkf) for robot intelligence," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, IEEE, 2007, pp. 429–436.

[109] G. H. Lim, I. H. Suh, and H. Suh, "Ontology-based unified robot knowledge for service robots in indoor environments," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 41, no. 3, pp. 492–509, 2011.

[110] L. Jacobsson, J. Malec, and K. Nilsson, "Modularization of skill ontologies for industrial robots," in *ISR 2016: 47st International Symposium on Robotics; Proceedings of*, VDE, 2016, pp. 1–6.

[111] X. Li, S. Bilbao, T. Martín-Wanton, J. Bastos, and J. Rodriguez, "Swarms ontology: A common information model for the cooperation of underwater robots," *Sensors*, vol. 17, no. 3, p. 569, 2017.

[112] M. Diab, A. Akbari, J. Rosell, *et al.*, "An ontology framework for physics-based manipulation planning," in *Iberian Robotics conference*, Springer, 2017, pp. 452–464.

[113] M. Tenorth and M. Beetz, "Representations for robot knowledge in the knowrob framework," *Artificial Intelligence*, vol. 247, pp. 151–169, 2017.

[114] D. Nitti, T. De Laet, and L. De Raedt, "A particle filter for hybrid relational domains," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2013, pp. 2764–2771.

[115] E. Erdem and V. Patoglu, "Applications of asp in robotics," *KI-Künstliche Intelligenz*, vol. 32, no. 2, pp. 143–149, 2018.

[116] E. Erdem, E. Aker, and V. Patoglu, "Answer set programming for collaborative housekeeping robotics: Representation, reasoning, and execution," *Intelligent Service Robotics*, vol. 5, no. 4, pp. 275–291, 2012.

[117] S. Zhang, M. Sridharan, and J. L. Wyatt, "Mixed logical inference and probabilistic planning for robots in unreliable worlds," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 699–713, 2015.

[118] L. De Raedt and K. Kersting, "Statistical relational learning," in *Encyclopedia of Machine Learning*, Springer, 2011, pp. 916–924.

[119] M. Richardson and P. Domingos, "Markov logic networks," *Machine learning*, vol. 62, no. 1-2, pp. 107–136, 2006.

[120] S. Imai, N. Jain, and A. Ching, "Bayesian estimation of dynamic discrete choice models," *Econometrica*, vol. 77, no. 6, pp. 1865–1899, 2009.

[121] L. Getoor and B. Taskar, *Introduction to statistical relational learning*. MIT press, 2007.

[122] L. De Raedt, *Logical and relational learning*. Springer Science & Business Media, 2008.

[123] D. Nyga, F. Balint-Benczedi, and M. Beetz, "Pr2 looking at things—ensemble learning for unstructured information processing with markov logic networks," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2014, pp. 3916–3923.

[124] B. Moldovan and L. D. Raedt, "Occluded object search by relational affordances," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 169–174, 2014.

[125] D. Nitti, T. De Laet, and L. De Raedt, "Relational object tracking and learning," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, IEEE, 2014, pp. 935–942.

[126] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[127] J. Sung, S. H. Jin, and A. Saxena, "Robobarista: Object part based transfer of manipulation trajectories from crowd-sourcing in 3d pointclouds," in *Robotics Research*, Springer, 2018, pp. 701–720.

[128] X. Wang, Y. Ye, and A. Gupta, "Zero-shot recognition via semantic embeddings and knowledge graphs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6857–6866.

[129] J. Sung, I. Lenz, and A. Saxena, "Deep multimodal embedding: Manipulating novel objects with point-clouds, language and trajectories," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 2794–2801.

[130] W. Liu, A. Daruna, and S. Chernova, "Cage: Context-aware grasping engine," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 2550–2556.

[131] T. Mikolov, W.-t. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, 2013, pp. 746–751.

[132] K. Marino, R. Salakhutdinov, and A. Gupta, "The more you know: Using knowledge graphs for image classification," *CoRR*, vol. abs/1612.04844, 2016.

[133] W. Yang, X. Wang, A. Farhadi, A. Gupta, and R. Mottaghi, "Visual semantic navigation using scene priors," *arXiv preprint arXiv:1810.06543*, 2018.

[134] R. M. Neal, *Bayesian learning for neural networks*. Springer Science & Business Media, 2012, vol. 118.

[135] J. Heinsohn, "Probabilistic description logics," in *Uncertainty Proceedings 1994*, Elsevier, 1994, pp. 311–318.

[136] E. Kolve *et al.*, "Ai2-thor: An interactive 3d environment for visual ai," *arXiv preprint arXiv:1712.05474*, 2017.

[137] A. Chang *et al.*, "Matterport3d: Learning from rgb-d data in indoor environments," *arXiv preprint arXiv:1709.06158*, 2017.

[138] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[139] D. Jain and M. Beetz, "Soft evidential update via markov chain monte carlo inference," in *Annual Conference on Artificial Intelligence*, Springer, 2010, pp. 280–290.

[140] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[141] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[142] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, 2020.

[143] A. Daruna, W. Liu, Z. Kira, and S. Chetnova, "Robocse: Robot common sense embedding," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 9777–9783.

[144] D. Nyga, M. Picklum, M. Beetz, *et al.*, *pracmln – markov logic networks in Python*, [Online; accessed ¡date¿], 2013.

[145] J. Orkin and D. Roy, "The restaurant game: Learning social behavior and language from thousands of players online," *Journal of Game Development*, vol. 3, no. 1, pp. 39–60, 2007.

[146] R. Scalise, J. Thomason, Y. Bisk, and S. Srinivasa, "Improving robot success detection using static object data," *arXiv preprint arXiv:1904.01650*, 2019.

[147] D. Paulius, N. Eales, and Y. Sun, "A motion taxonomy for manipulation embedding," *arXiv preprint arXiv:2007.06695*, 2020.

[148] X. Puig *et al.*, “Virtualhome: Simulating household activities via programs,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8494–8502.

[149] R. Speer, J. Chin, and C. Havasi, “Conceptnet 5.5: An open multilingual graph of general knowledge,” *arXiv preprint arXiv:1612.03975*, 2016.

[150] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, “Convolutional 2d knowledge graph embeddings,” in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[151] P. Wang, S. Li, and R. Pan, “Incorporating gan for negative sampling in knowledge representation learning,” in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[152] L. Cai and W. Y. Wang, “Kbgan: Adversarial learning for knowledge graph embeddings,” *arXiv preprint arXiv:1711.04071*, 2017.

[153] Y. Dai, S. Wang, X. Chen, C. Xu, and W. Guo, “Generative adversarial networks based on wasserstein distance for knowledge graph embeddings,” *Knowledge-Based Systems*, vol. 190, p. 105 165, 2020.

[154] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, “Generating sentences from a continuous space,” *arXiv preprint arXiv:1511.06349*, 2015.

[155] R. Das *et al.*, “Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning,” *arXiv preprint arXiv:1711.05851*, 2017.

[156] M. T. Ribeiro, S. Singh, and C. Guestrin, “” why should i trust you?” explaining the predictions of any classifier,” in *Int. Conf. on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[157] M. Gardner and T. Mitchell, “Efficient and expressive knowledge base completion using subgraph feature extraction,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1488–1498.

[158] K. v. Deemter, M. Theune, and E. Krahmer, “Real versus template-based natural language generation: A false opposition?” *Computational linguistics*, vol. 31, no. 1, pp. 15–24, 2005.

[159] G. Shan and S. Gerstenberger, “Fisher’s exact approach for post hoc analysis of a chi-squared test,” *PloS one*, vol. 12, no. 12, 2017.

[160] A. Daruna, L. Nair, W. Liu, and S. Chernova, “Towards robust one-shot task execution using knowledge graph embeddings,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 11 118–11 124.

[161] X. Chen, M. Chen, W. Shi, Y. Sun, and C. Zaniolo, “Embedding uncertain knowledge graphs,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3363–3370.

[162] F. Salehi, R. Bamler, and S. Mandt, “Probabilistic knowledge graph embeddings,” 2018.