Copyright

by

Aishwarya Padmakumar

2020

The Dissertation Committee for Aishwarya Padmakumar
certifies that this is the approved version of the following dissertation:

Dialog as a Vehicle for Lifelong Learning of Grounded Language Understanding Systems

Committee:

Raymond J. Mooney, Supervisor

Peter Stone

Scott Niekum

Joyce Chai

Dialog as a Vehicle for Lifelong Learning of Grounded Language Understanding Systems

by

Aishwarya Padmakumar

Dissertation

Presented to the Faculty of the Graduate School of
The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

The University of Texas at Austin
August 2020

Acknowledgments

This dissertation would not have been possible without the help of a number of people both during the last five years and in the many steps that led me to start a PhD. I have tried to include as many people as possible here but there are many people I have not been able to name in person, and perhaps many whose influence I have not fully appreciated, and would like to apologize to those who I may have missed out.

To begin with, my sincerest thanks go to my advisor Raymond J. Mooney for inspiring me with his vision and passion, encouraging me to collaborate and push my boundaries, being patient through my delays, and for all the book recommendations. I would like to thank Peter Stone for his advice and helpful discussions on many of my projects. I would also like to thank Scott Niekum and Joyce Chai for serving on my committee and providing feedback that has improved my dissertation. My research has also been heavily influenced by Jesse Thomason, who I would like to thank for being a wonderful mentor and friend, and for helping me from start to finish, from dealing with my likely silly questions on my first project to answering questions about administrative work before my defense.

There are many people without whom I could not have reached a point where I could start a PhD. I am indebted to my parents for instilling in me the value of education and hard work, as well as providing the opportunities I needed to reach this stage. I am grateful to my teachers at NPS Indiranagar for helping me develop my early interest in learning. I am grateful to Sridhar Sir and the teachers at ACE for paving the way for me to join IIT, Madras, furthering my interest in learning, and opening up the possibility of many new futures. I am grateful to my professors at IIT, Madras for many wonderful classes, my initial exposure to research, and encouraging me to apply for a graduate degree. In particular, I would like to thank my undergraduate advisor Prof Balaraman Ravindran for getting me interested in Machine Learning, giving me the opportunity to participate in a number of research projects, enabling me to focus on my work instead of the results, and guiding me when applying to graduate school. I would also like to thank Prof Krishna Nandivada for conversations that shaped many of my important decisions.

Moving across the world was made easier for me by Subhashini Venugopalan, who I would like to thank for her help with my questions about the transition and with finding an initial place to stay. I would also like to thank RFI and the many friends I made through it for making my first year here so special.

I also thank many friends and colleagues in the NLP groups and AI lab at UT, including Jesse Thomason, Prasoon Goyal, Nazneen Rajani, Akanksha Saran, Stephen Roller, Elad Liebman, Sanmit Narvekar, Jivko Sinapov, Harel Yedidson, Subhashini Venugopalan, Karl Pichotta, Eric Holgate, Elisa Ferracane, Dinesh Jayaraman, Pengxiang Chang, Sheena Panthaplackel, Angela Lin and Jialin Wu. Thank you for helping me out when I got stuck with something, developing research ideas, feedback on papers and talks, career advice and talking through difficult personal situations. I would also like to thank professors Joydeep Biswas, Greg Durrett and Katrin Erk, as well as members of the BWI group, and NLL and CLAMP reading groups for many discussions that broadened my general knowledge of the fields of NLP and Robotics. Also thanks to the greater NLP and language and vision communities for the work that inspired and that I built on for my research. I would also like to thank Raghav Gupta, Dilek Hakkani-Tur, Sandra Faust, Chase Kew, Marek Fiser, Alexander Toshev, and the Google Deep dialog team for a wonderful internship experience. I would also like to thank many friends, including the ones already mentioned, and Srilakshmi Pattabhiraman, Surbhi Goel, Shailee Jain, Matt Denend, John Kallaugher, Kevin Song, Tanya Goyal, Ishan Durugkar, Tushar Nagarajan, Eddy Hudson, Sushmitha Ayyanar, Shankari Subramaniam, Arun Sai Suggala, Priya Darshini, Preeti Chopra, and many others, for the wonderful memories you have helped me create and helping me through the harder parts of the last five years.

I would also like to thank Stacy Miller, Megan Booth, Katie Traughber, Amy Bush and the rest of the CS and helpreq staff for their invaluable assistance. I would also like to thank Dr Jeanne Beattie, Elana Bizer, Emily Shyrock and the rest of the doctors and staff at UHS, as well as the members of the Connecting Around Chronicity group for their support with the health issues that would have otherwise taken a greater toll on my research.

To conclude, special thanks to my brother Vishakh Padmakumar, my cousins Subramaniam M, Vishnu Mahadava Iyer and Aparna Prabha Ramakrishnan, and dear friends Dhivya Eswaran, Aarati Karaparthy, Kartik Gupta for their immense support during the last few years that helped me get through many hard moments. Finally, another particularly special mention goes to my boyfriend Abhinav Garlapati for his continuous support, patience and encouragement, and for giving me more happiness in the last few years than I would have believed possible. I would not be here today without your support.

Dialog as a Vehicle for Lifelong Learning of Grounded Language Understanding Systems

Aishwarya Padmakumar, Ph.D.
The University of Texas at Austin, 2020

Supervisor: Raymond J. Mooney

Natural language interfaces have the potential to make various forms of technology, including mobile phones and computers as well as robots or other machines such as ATMs and self-checkout counters, more accessible and less intimidating to users who are unfamiliar or uncomfortable with other types of interfaces. In particular, natural language understanding systems on physical robots face a number of challenges, including the need to ground language in perception, the ability to adapt to changes in the environment and novel uses of language, and to deal with uncertainty in understanding. To effectively handle these challenges, such systems need to perform lifelong learning - continually updating the scope and predictions of the model with user interactions. In this thesis, we discuss ways in which dialog interaction with users can be used to improve grounded natural language understanding systems, motivated by service robot applications.

We focus on two types of queries that can be used in such dialog systems – active learning queries to elicit knowledge about the environment that can be used to improve perceptual models, and clarification questions that confirm the system's hypotheses, or elicit specific information required to complete a task. Our goal is to build a system that can learn how to interact with users balancing a quick completion of tasks desired by the user with asking additional active learning questions to improve the underlying grounded language understanding components.

We present work on jointly improving semantic parsers from and learning a dialog policy for clarification dialogs, that improve a robot's ability to understand natural language commands. We introduce the framework of opportunistic active learning, where a robot introduces opportunistic queries, that may not be immediately relevant, into an interaction in the hope of improving performance in future interactions. We demonstrate the usefulness of this framework in learning to ground natural language descriptions of objects, and learn a dialog policy for such interactions. We also learn dialog policies that balance task completion, opportunistic active learning, and attribute-based clarification questions. Finally, we attempt to expand this framework to different types of underlying models of grounded language understanding.

Table of Contents

Chapter 1 Introduction 1

Chapter 2 Background and Related Work 7

- 2.1 Natural language commands to Robots 8
- 2.2 Dialog Systems 12
- 2.3 Lifelong Learning 14
- 2.4 Lifelong Learning Using Dialog Systems 17
- 2.5 Human-robot Dialog for Teaching Perceptual Concepts 18
- 2.6 Semantic Parsing 20
- 2.7 Reinforcement Learning 21
  - 2.7.1 Markov Decision Process (MDP) 22
  - 2.7.2 Partially Observable Markov Decision Process (POMDP) 22
  - 2.7.3 REINFORCE Algorithm 23
  - 2.7.4 Q-Learning 24
  - 2.7.5 Asynchronous Advantage Actor Critic 25
  - 2.7.6 KTD-Q Learning 26
- 2.8 Active Learning 27
- 2.9 Grounding Language in Perception 29
- 2.10 Visually Grounded Dialog 31
- 2.11 Interactive Image Retrieval 33

Chapter 3 Integrated Learning of Dialog Strategies and Semantic Parsing 35

- 3.1 Contributions 35

3.2 Motivation 36
3.3 Task and System Setup 38
3.4 Semantic Parser Learning 40
3.5 Dialog Policy Learning 41
3.6 Experimental Setup 44
    3.6.1 Task Interface 44
    3.6.2 Dialog agents 45
    3.6.3 Experiment hypothesis 46
3.7 Results and Discussion 47
3.8 Summary 50

Chapter 4 Opportunistic Active Learning for Grounding Natural Language Descriptions 52
4.1 Contributions 53
4.2 Motivation 53
4.3 Opportunistic Active Learning 56
4.4 Object Retrieval Task 59
4.5 Experimental Setup 63
4.6 Results and Discussion 65
4.7 Summary 65

Chapter 5 Learning a Policy for Opportunistic Active Learning 67
5.1 Contributions 67
5.2 Motivation 67

5.3 Task Setup 70
5.4 Methodology 72
  5.4.1 Grounding Descriptions 73
  5.4.2 MDP Formulation 73
  5.4.3 Identifying Candidate Queries 74
  5.4.4 Baseline Static Policy 76
  5.4.5 Policy Learning 76
  5.4.6 Features for Policy Learning 77
5.5 Experiments 80
  5.5.1 Dataset 80
  5.5.2 Experiment phases 81
5.6 Experimental Results and Analysis 82
5.7 Summary 88

Chapter 6 Dialog Policy Learning for Joint Clarification and Active Learning Queries 89
6.1 Contributions 90
6.2 Motivation 91
6.3 Task Setup 93
6.4 Methodology 95
  6.4.1 Visual Attribute Classifier 95
  6.4.2 Grounding Model 98
  6.4.3 Initial Retrieval 98
  6.4.4 MDP Formulation 100

6.4.5 Policy Learning ... 101
6.4.6 Clarification policy features ... 103
6.4.7 Active learning policy features ... 104
6.4.8 Decision policy features ... 105
6.4.9 Baseline static policy ... 105
6.5 Experimental Setup ... 108
6.5.1 Dataset ... 108
6.5.2 Experiment Phases ... 110
6.6 Results and Discussion ... 111
6.7 Summary ... 115

Chapter 7 Human Evaluation of Dialog Policy Learning for Joint Clarification and Active Learning Queries ... 117
7.1 Models ... 117
7.2 Experimental Design Changes ... 118
7.3 Modified Task Setup ... 119
7.4 User Interface Challenges ... 120
7.5 Results and Discussion ... 125
7.6 Summary ... 127

Chapter 8 Clarification in Language Grounding Models Based on Joint Embeddings ... 133
8.1 Motivation ... 133
8.2 Task Setup ... 135

8.3 Methodology 137

8.3.1 Grounding Model 137

8.3.2 Identifying and Scoring Clarifications 141

8.4 Experimental Setup 142

8.5 Results and Discussion 144

8.6 Summary 146

Chapter 9 Future Work 148

9.1 Better Language Grounding Models 148

9.2 Multimodal Language Grounding 149

9.3 Active Learning in Practice 150

9.4 Sim2Real Issues in Dialog Systems for Lifelong Learning 152

9.5 Learning From Demonstration With Dialog 154

Chapter 10 Conclusions 156

References 159

Vita 181

List of Tables

3.1 Features used in summary space.. 43

3.2 Performance metrics for dialog agents tested. Differences in dialog success and subjective metrics are statistically significant according to an unpaired t-test with $p < 0.05$.. 47

3.3 Comparison of performance of initial parser and parsers after updating various components, on paired commands and semantic forms. $*$ indicates that the difference in performance between this and the *Initial* parser on the same metric is statistically significant according to a paired t-test with $p < 0.05$ and ˆ indicates that the difference is trending significance ($p < 0.1$)... 49

3.4 Comparison of an agent using only the top hypothesis from the semantic parser and another using the top 10 parses. All differences are statistically significant according to an unpaired t-test with $p < 0.05$. 50

4.1 Fraction of successful guesses made in each condition per round...... 65

5.1 Results on dialogs sampled from the policy test set after 10 batches of classifier training. *–Guess* and *–Query* are conditions with the guess and query features, respectively, ablated. Boldface indicates that the difference in that metric with respect to the *Static* policy is statistically significant according to an unpaired Welch t-test with $p < 0.05$. 82
5.2 Results of individual feature ablation. 84

6.1 Results from the final batch of the test phase. * indicates the conditions whose performance is comparable to the best condition (in bold). 112

7.1 Results of the static and new learned policies at the end of the test phase in simulation and in interactions on Amazon Mechanical Turk. Bold indicates a statistically significant improvement over the baseline (p < 0.05) and italic indicates trending significance (p <= 0.1) according to an unpaired Welch t-test 126
7.2 A sample interaction from Amazon Mechanical Turk. 128

List of Figures

2.1 Spoken Dialog System Pipeline 12

3.1 A sample dialog in our interface on Amazon Mechanical Turk......... 39

3.2 Training pairs induced from the clarification dialog, by selecting parses that ground to the final action and its arguments. The response in red is discarded because no such parse is found................ 41

4.1 Experiment Setup – Participants described an object on Table 2 from the *active test set* to the robot in natural language, then answered the robot's questions about the objects in its *active training set* on the side Tables 1 and 3 before the robot guessed the described target object. 59

4.2 A sample interaction in our experiment.. 60

4.3 The objects used in our experiments, from fold 0 on the far left to fold 3 on the far right. 63

5.1 A sample interaction – Perceptual predicates are marked in bold....... 71

6.1 A stylized sample interaction. In this work, we use simulated dialog acts but the natural language glosses represent how such a dialog could look in an end application. 90
6.2 Visual Attribute Classifier. 96
6.3 Comparison of guess success rate with, and without clarifications across test batches. 114
6.4 Dialog turns of each type taken by the best learned policy in the test phase. 115

7.1 Initial Qualification Task Interface on Amazon Mechanical Turk. 121
7.2 Final qualification interface for Amazon Mechanical Turk. 122
7.3 Final dialog task on Amazon Mechanical Turk – description page. 125

8.1 Grounding model that projects words and images to a joint space, and uses Platt scaling to convert the distances in the space to probabilities. 137
8.2 Information gain vs difference in belief of target image $b_2(i^*) - b_1(i^*)$. Words which raise the belief of the target image are plotted in blue and others are plotted red. 145

Chapter 1

Introduction

Natural language interfaces have the potential to make various forms of technology, including mobile phones and computers, as well as robots or other machines such as ATMs and self-checkout counters, more accessible and less intimidating to users who are unfamiliar or uncomfortable with other interfaces such as command shells, button based interfaces or changing screens. Interaction in natural language is particularly desirable with versatile devices such as service robots intended to be used in environments such as homes, offices and hospitals. Such environments typically contain a variety of objects that are difficult to catalog, and service robots in such environments would need to perform a variety of functions and interact with a number of people. It is difficult to capture this rich set of possible interactions in more structured interfaces such as a panel of buttons or a list on a touch screen, thus making a natural language interface preferable.

In many of these applications, the minimal level of useful interaction would be for the system to understand high level natural language commands, and detect and indicate when it has failed to understand what a user requires. Further, it would be desirable if it could engage in a dialog with the user, clarifying their intentions in case of uncertainty. For service robots expected to be used in a variety of domains, the robot would need to expand the coverage of its models to adapt to domain specific language use such as nicknames used for people, and special objects such as a stethoscope in a hospital or a printer in an office. Robots can acquire the training data for such domain adaptation by interacting with users in their operating environment.

In recent years, voice assistants have become a common part of mobile phones and home devices. They are also increasingly being used in a variety of customer service applications. These can perform a large number of functions, but they typically assume the existence of a list of pre-defined commands that a user may want to give. They also often depend on specific language for requesting many of the tasks. However, it would be preferable if such systems could learn different ways in which people may refer to the same command, or use follow-up questions to clarify ambiguous or incomplete instructions. It would also be desirable for such systems to identify frequently requested entities that are not currently listed in the system, or desirable tasks outside its current scope. In some cases, it may be appropriate for the system to automatically learn to incorporate these, and in others, this information can provide a guideline for the system designer to expand the scope of the system.

This sort of adaptation is the goal of lifelong learning systems. In this paradigm, machine learning systems are designed such that they can expand the range of concepts or tasks they can perform, beyond what is defined initially (Chen and Liu, 2018). Most of this work assumes that the systems involved obtain additional training data as the tasks or domains of operation change, but do not discuss how such training data is to be collected. Dialog systems are in a unique position to perform lifelong learning, because by their very nature, they are continually interacting with users during operation.

Most research on dialog systems assumes access to training dialogs of the type the system is expected to perform, and aims to build a dialog system that can then engage in similar interactions. However, it is often difficult to obtain labelled data during training that adequately covers all scenarios that the agent is likely to encounter during operation. When the system is unable to adapt to these differences, it often has to resort to frustrating back-off methods such as providing the user a list of commands it can understand. Instead, it would be beneficial to leverage interactions with humans during operation to obtain additional labelled data to adapt to possible changes in domains or tasks.

Natural language understanding systems on physical robots have additional requirements. For example, consider the following command that a service robot in an office environment may need to understand – "*Bring the blue mug from Alice's office*". Understanding such commands requires many types of capabilities. The robot would need to understand compositionality - how meanings of individual words combine to give meanings of phrases. Here, it would need to know that "*Alice's office*" means an office that is owned by Alice, and that "*the blue mug*" is something which is both a mug and is blue.

The robot must also be able to ground such meanings to entities in its environment. Here, it would need to know that "*Alice's office*" refers to a physical location. It would also need to be able to identify mugs and blue objects. In contrast, most existing voice assistants such as those on mobile phones, only need to maintain a list of actions with corresponding APIs, and valid arguments for these.

Some types of knowledge, such as which office belongs to whom, can be hard coded as facts. Existing voice assistant systems often assume that all required knowledge can be encoded in this manner. However, environments such as homes and offices typically contain a large number of smaller objects such as mugs, whose existence and properties would be tedious to catalog. Thus, a robot operating in such an environment would need to be able to identify such objects through perception, perhaps using a camera or by manipulating it with an arm. The ability to ground natural language in vision can also be useful for other computer applications such as interactive online shopping or image search. Perceptual grounding is the process of associating words such as "*blue*" and "*mug*" with specific sensory information. For example, a visual classifier that uses an image of an object to determine whether it is "*blue*" or a "*mug*" can be used to ground meanings of words and phrases that reference visual properties. Most of our work is focused on enabling lifelong learning in systems that perform perceptual grounding of natural language.

In most commercial voice assistants, if the system does not understand a user's request, it often either asks the user to repeat the command, or provides a list of examples it is capable of understanding. However, a more desirable behaviour for a robot that only partially understands a command given to it, would be to ask clarification questions, such as "*What do you want me to bring?*", to obtain the missing information and avoid making mistakes. There is a lot of research in the area of task oriented dialog systems that is aimed at learning dialog policies that help a system choose between such clarification questions (Young et al., 2013).

However, most of this work assumes a fairly limited set of clarification questions, based on a fixed set of “slots” that the system needs to fill to interpret a request. Such questions are typically insufficient in situations where the dialog system needs to perform perceptual grounding. In our work, we design better clarification questions for applications that require perceptual grounding.

Dialog interactions can provide implicit cues about when words or phrases share the same meaning, which can also be used by the system to improve its ability to understand future interactions. Also, if a robot does not know the meaning of a word such as “*mug*”, it could ask the user to show examples of mugs nearby so that it knows what to look for. It could also opportunistically ask the user to show examples of other things, such as a “*book*”, so that it is better prepared to help a different user that may need a book to be fetched. We would like the robot to be able to learn both when to ask such questions, and which questions to ask, through interactions with users.

In this work, our goal is to use dialog interactions to enable lifelong learning in dialog systems that require grounded natural language understanding, motivated by applications in service robotics. Following a discussion of related work (chapter 2), we present work on jointly learning a dialog policy that enables a robot to ask clarifications when it does not fully understand natural language commands, while simultaneously using the dialogs to improve the underlying semantic parser for future commands (chapter 3). The continual improvement of the semantic parser allows the system to learn different ways people use language to refer to a set of pre-defined tasks, and the improvement of the dialog policy enables the system to avoid unnecessary clarification questions as its language understanding capacity improves.

We then present the framework of opportunistic active learning in the context of understanding natural language descriptions of objects. Opportunistic active learning is a framework for intergrating active learning queries into dialog interactions during operation. In chapter 4, we introduce the framework and demonstrate its effectiveness in learning new perceptual concepts for improving a system's ability to understand natural language descriptions of objects. We expand this in chapter 5 by learning a dialog policy that learns to choose opportunistic active learning queries for model improvement, and learns to decide between further improving the model by asking additional active learning queries and completing the task with shorter dialogs.

Subsequently, in chapter 6, we combine opportunistic active learning with attribute-based clarifications and learn a hierarchical dialog policy that learns to choose opportunistic active learning queries that are most likely to improve the underlying grounding model, and attribute-based clarification queries that are most likely to help the system successfully understand the user's request, and decide between quickly completing the current interaction using only clarification, and further improving the model through opportunistic active learning to increase the chances of succeeding in future interactions. We also attempt to extend this clarification mechanism to more general models of perceptual grounding (chapter 8). Finally, we summarize our contributions (chapter 10) and discuss directions for future work (chapter 9).

Chapter 2

Background and Related Work

In this chapter, we discuss some background concepts used in our research, as well as prior and contemporary work on some related topics. We begin with a discussion of prior work on giving various types of natural language commands to robots (section 2.1), with a focus on work that tries to improve the language understanding system from interaction. This is followed by a discussion of dialog systems (section 2.2), including a standard dialog system pipeline, research on improving various components of this pipeline, and applications for which dialog systems have been developed. We then review the concept of lifelong learning (section 2.3) and examine the extent to which existing dialog systems perform or enable lifelong learning (section 2.4), of which a special category is work on human-robot dialog for teaching perceptual concepts to robots (section 2.5). We then summarize some relevant background concepts - semantic parsing (section 2.6), reinforcement learning (section 2.7), active learning (section 2.8), and perceptual grounding (section 2.9). We finally dig deeper into two types of perceptual grounding applications closely related to our work – visually grounded dialog (section 2.10) and interactive image retrieval (section 2.11).

2.1 Natural language commands to Robots

Communicating with robots in natural language has been an area of interest for a long time. Theobalt et al. (2002) is an early attempt at developing a dialog system interface over low level navigation. Users could query the robot about it's position and command it to navigate to a specific location using rich language including landmarks. Semantic parsing with hand-coded rules was used for language understanding. Another early work is that of Lauria et al. (2002) which proposes a learning from demonstration framework that uses natural language to instruct robots at a symbolic level. These used pre-programmed language understanding, perceptual grounding and dialog policies. They also do not evaluate the performance of the system.

However, over time, there has been an increasing focus on using machine learning to improve such natural language interfaces. For example, Matuszek et al. (2013) learn a semantic parser from paired sentences and annotated semantic forms to map natural language commands to high level goals that are more independent of the environment. A related work is Chen and Mooney (2011), who learn a semantic parser to translate natural language route instructions to logical plans for a simulated environment. Other works use fixed components for parsing natural language but learn models for grounding to symbols in the robot's knowledge base. Tellex et al. (2014) develop a graphical model for grounding that makes use of a pretrained parser. The model can also be used for generating clarification questions. Arumugam et al. (2018) learn a probabilistic model for grounding to symbolic goals that act as rewards in a hierarchical state and action MDP space to handle commands of varying levels of abstraction. Also related is Bastianelli et al. (2016), who use symbolic grounding to resolve ambiguities from semantic parsing.

Other works focus on learning dialog policies for effective human-robot communication. Zhang and Stone (2015) develop a system that learns a dialog policy by modeling dialog as a POMDP (section 2.7.2). They also incorporate logical reasoning, and common sense knowledge with the result of pretrained natural language understanding components. Whitney et al. (2017) learn a policy for clarification dialogs that can incorporate both natural language responses and gestures made by users.

Dialog systems have also been used to enable robots to perform lifelong learning in different ways. One mechanism is to use dialog to teach new instructions – She et al. (2014) build a dialog system for instructing robots to combine simple instructions in a blocks world to complex ones. They use fixed components for semantic parsing, perceptual grounding and the dialog policy, but report success rates for teaching different types of instructions. Some systems include explicit queries in the dialog to gather information about the environment. Kollar et al. (2013b) use a static dialog policy to add new concepts to a knowledge base that is then used in a probabilistic language grounding model. The structure of the dialog can also provide implicit clues to improve the natural language understanding system used by the robot. Thomason et al. (2015) develop a system that learns a semantic parser for understanding natural language commands. They use a static policy to ask clarification questions, but also use responses from clarifications as weak supervision to improve the parser. We extend this in our work (chapter 3) by learning a dialog policy for clarifications while simultaneously improving the semantic parser from clarification responses.

More recently, there has been interest in learning end-to-end neural networks to map natural language instructions and observations directly to action sequences. Earlier work in this space was either on tasks that require only grounding to a knowledge base, such as mapping to formal queries (Suhr et al., 2018), or used simulated datasets that did not require real perception (Mei et al., 2016; Misra et al., 2017a).

Recently, large scale simulated datasets (Chang et al., 2017; Yan et al., 2018) have enabled the development of end to end neural networks that use complex visual observations to map natural language commands to actions for tasks such as following route instructions (Anderson et al., 2018), embodied question answering (Das et al., 2018), navigation combined with object manipulation (Misra et al., 2017a; Shridhar et al., 2020), and continuous control of a quadcopter drone (Blukis et al., 2018). The first end-to-end models for these tasks used simple recurrent networks to encode the instruction, convolutional neural networks to encode visual observations, and recurrent networks to decode action sequences.

More recently, many attempts have been made to introduce structure in different ways into these networks to increase interpretability, and to imbue them with some of the benefits of more classical pipelined approaches. Misra et al. (2017a) modify a fully convolutional network designed for image segmentation to explicitly predict a goal location given a natural language instruction and egocentric observations. This is then used to plan low level actions. Ma et al. (2019) include a module that estimates the progress made towards completing the instruction to ensure that the agent attends to the correct part of the instruction for predicting the next action. Paxton et al. (2019) explicitly predict the next observation, and interpretable subgoals in combination with associated low-level actions. Predicted subgoals and observations can be used to better analyze failure cases. Zhu et al. (2020) break up long instructions into smaller steps and learn to predict action sequences for these, leading to better generalization. Other works incorporate clarification actions that allow the agent to obtain the next navigation step using an oracle (Chi et al., 2020; Nguyen and III, 2019).

Other recent developments include the creation of more complex simulated datasets for instruction following. Thomason et al. (2020) create an interactive instruction following task, where the robot must learn to ask follow up questions to clarify ambiguous instructions. Chen et al. (2019a) combine multi-step navigational instructions with an object segmentation task at the end. This dataset also uses complex outdoor scenes. Shridhar et al. (2020) create a dataset of multi-step instructions that combine navigation, object segmentation, and categorical articulation actions simulating tasks in a kitchen. Another recent development is the use of pretraining by allowing agents to explore the action space using only visual observations, to improve generalization to unseen natural language instructions (Tan et al., 2019; Lynch and Sermanet, 2020).

2.2 Dialog Systems

Spoken Dialog Systems allow users to interact with information systems with speech as the primary form of communication (Young et al., 2013). They were originally deployed for call center operations such as airline ticket reservation (Hemphill et al., 1990), and restaurant recommendation (Wen et al., 2017). More recently, dialog systems have become popular for issuing simple commands on mobile phones through virtual assistants such as Apple's Siri, Google Voice and Amazon's Alexa.

Spoken dialog systems typically follow a pipeline similar to that in figure 2.1. The user utterance is first processed by a speech recognition module, which produces a text transcript. This is followed by a language understanding module that extracts the information provided by the user in the utterance. This is used by the dialog state tracking module to update the system's belief of what the user wishes to accomplish from the interaction. Following this, the dialog management module uses the system's dialog policy to decide which dialog action to take next, for example ask for more information. The response generation module converts this abstract dialog act into a natural language response, which is rendered into speech by the speech synthesis module.

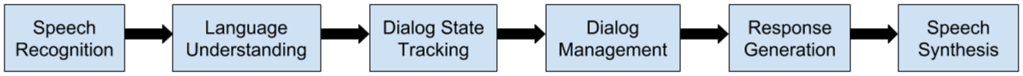


Figure 2.1: Spoken Dialog System Pipeline

There has been considerable research in goal directed dialog systems targeted at performing call-center type tasks (Young et al., 2013). These systems model dialog as a POMDP and focus on either the problem of tracking belief state accurately over the large state spaces (Young et al., 2010; Thomson and Young, 2010; Mrkšić et al., 2015) or that of efficiently learning a dialog policy over such state spaces (Gašić and Young, 2014; Pietquin et al., 2011). These systems typically assume that the other components of the pipeline are fixed. Some of our work (chapter 3) combines this research with research on learning semantic parsers from weak supervision provided by clarification dialogs.

More recently, there has been work on modeling various components of a dialog system using neural networks (Mrkšić et al., 2015; Wen et al., 2015a). There have also been some end-to-end neural network systems that simultaneously learn dialog policy and language comprehension (Wen et al., 2017; Williams and Zweig, 2016; Bordes and Weston, 2016). A major challenge in these systems is to find database entries satisfying certain constraints, and ensuring that all relevant information is included in the system's responses. Some systems assume that these functions are performed by deterministic APIs (Bordes and Weston, 2016). Others attempt to design neural networks to perform these functions (Wen et al., 2015b; Kiddon et al., 2016).

Some other tasks for which dialog systems have been developed are open domain conversations (Serban et al., 2016), playing 20 questions games about famous people (Hu et al., 2018a), and converting natural language to code snippets (Chaurasia and Mooney, 2017).

2.3 Lifelong Learning

Lifelong learning research aims to develop machine learning systems that can make use of knowledge from previous tasks to improve the performance and sample efficiency of future tasks. More specifically, Chen and Liu (2018), use the following three characteristics to identify lifelong learning systems:

- The system performs continuous learning.
- The system accumulates knowledge from previous tasks and stores them in some sort of reusable knowledge base.
- The system uses knowledge from past tasks to improve performance in future tasks.

Early work in lifelong learning was motivated by control problems in robotics (Thrun and Mitchell, 1995) in order to overcome the difficulties of acquiring accurate knowledge of the world (knowledge bottleneck), hand-coding this knowledge into a robot-accessible form (engineering bottleneck), computational intractability of optimally solving control problems in realistic settings (tractability bottleneck), and possible differences between the real world and the model of it used for planning (precision bottleneck). Many of these issues are still faced when developing and deploying AI systems for a variety of applications.

Another early work is by Banko and Etzioni (2007) where an agent must build a theory of a domain and and choose which of multiple learnable tasks to learn next. Ruvolo and Eaton (2013) focus on lifelong multi-task learning which alternates the training phase of each new task with testing phases for all previous tasks. They learn a library of latent model components that become a shared basis for all tasks which can be grouped or overlapped as needed. All of these assume the existence of a global latent task structure shared by multiple learnable tasks. This structure is exploited to transfer knowledge to future tasks.

These works are closely related to work in multi-task learning where systems learn multiple related tasks simultaneously, with the goal of improving performance of all tasks through the use of shared relevant knowledge (Caruana, 1997). A survey of multi-task learning can be found in Zhang and Yang (2017). The order in which tasks are learned may also affect the performance of the system in learning different tasks (Chen and Liu, 2018). Curriculum learning is an area of research focused on learning to order the possible tasks an agent may engage in, in order to maximally improve training speed or performance on a desired set of final tasks. A recent survey of curriculum learning can be found in Narvekar et al. (2020).

Lifelong supervised learning is a continuous supervised learning process where the learner performs a sequence of supervised learning tasks, and leverages knowledge from previous supervised learning tasks to improve performance in future ones. A special case is cumulative learning – where each task is the introduction of a new class (Fei et al., 2016). Fei et al. (2016) develop a system that incrementally detects the presence of new classes, and incrementally adds them to the existing model without retraining it from scratch. Our work on opportunistic active learning (chapters 4, 5, 6) falls under this paradigm, although our focus is on using dialog interactions to acquire the data necessary to perform lifelong learning, rather than the actual learning algorithms involved.

Other lifelong learning systems aim to learn more general knowledge that is expected to be useful for a variety of tasks. Carlson et al. (2010) aim to use some initial annotated resources to a system that alternates between extracting facts for a knowledge base by machine reading on un-annotated documents, and using extracted facts to improve machine reading. Chen et al. (2013) use Google Image Search to obtain data for training classifiers for objects, scenes and attributes, which is then used to learn visual relationships between objects, attributes and scenes. Yuyin Sun and Fox (2016) propose an extension of the above which learns a hierarchy of object names using a combination of Bayesian modelling and crowdsourced annotations to more effectively learn classifiers for an open vocabulary of objects. Of these, only Yuyin Sun and Fox (2016) explicitly address some of the difficulties around acquiring additional labels to continue learning. Research in lifelong learning assumes that the systems have access to additional training data as the domains and tasks change. In practice, since many systems require labeled data, some form of interaction is required to obtain these labels. Our work is focused on using dialog interactions as the mechanism to obtain such data needed to perform lifelong learning. We believe there are many open problems in this area that are particularly relevant to physically situated dialog, and discuss these in Padmakumar and Mooney (2020a).

2.4 Lifelong Learning Using Dialog Systems

Open-domain dialog systems consisting of learnable components that can be improved from dialogs can be considered a form of lifelong learning, since they can in principle learn to adapt to a variety of situations. However, these systems are typically designed with the objective of keeping a user engaged in conversation (Cervone et al., 2017), as opposed to expanding the range of topics the system can converse about. There is some work on extracting information such as user attributes from open ended dialogs (Wu et al., 2019) with potential applications in personalized recommendation, but the use of extracted information in such applications is yet to be tested empirically. Also related is the generation of curiosity-driven questions – questions that would enrich the system's knowledge (Scialom and Staiano, 2019). However, for practical lifelong learning, it would be additionally desirable to test that such question asking enables the system to perform better at some downstream task, for example question answering.

Although task-oriented dialogs are typically more restricted, the information-gathering style questions used in these dialogs can be used for lifelong learning. For example, Kollar et al. (2013b) use such queries to explicitly learn a knowledge base of referring expressions for people, locations and actions. More recently, She and Chai (2017) combine standard slot-value style clarification queries along with explicit knowledge seeking queries to build a knowledge base of the physical effects of actions on real world objects, while simultaneously using this to plan for and accomplish goals specified by the user.

Information from task-oriented dialogs can also be used to improve natural language understanding. Thomason et al. (2015) use the structure of task oriented dialogs, particularly the answers to clarification questions, to obtain weakly supervised training examples to improve a semantic parser. This can adapt to some changes in language use over time, as can end-to-end dialog systems (Wen et al., 2017), and those whose language understanding components can be updated over time in other ways from new dialogs (Mesnil et al., 2013). Other work has also shown that the use of clarifications can improve the future performance of an agent at following route instructions in simulated home environments (Chi et al., 2020).

There are also some dialog tasks that are designed solely for teaching specific language understanding capabilities to a system. We discuss these separately in section 2.5.

In chapter 4, we introduce the framework of Opportunistic Active Learning, which is a dialog framework intended to explicitly try to combine learning of new concepts with using them in an end task, as would be required by an agent performing lifelong learning.

2.5 Human-robot Dialog for Teaching Perceptual Concepts

A first step towards teaching robots perceptual concepts through dialog is Kollar et al. (2013a), who develop a system that uses semantic parsing for language understanding, and grounds meanings of words using SVM-based perceptual classifiers. This is trained using pairs of images and corresponding language descriptions, and can generate descriptions of objects, but these not evaluated as a complete interactive system.

Other works combine these capabilities to perform clarifications to better ground descriptions at test time (Dindo and Zambuto, 2010; Parde et al., 2015). Vogel et al. (2010) learn to ground simple perceptual concepts using only a 20-questions style game, where there is no initial description, but learning is entirely driven by the robot's queries of whether a concept applies to an object. Kulick et al. (2013) use active learning to enable a robot to learn spatial relations by manipulating objects into specific positions and querying an oracle about whether a relation holds. However they receive ground truth positions of objects and do not perform perception.

Thomason et al. (2016) demonstrate that multimodal perceptual concepts, with richer visual features, as well as auditory and haptic features, can be learned from an I Spy game, by pairing descriptions with correct guesses by the robot.

Some works also try to learn a dialog policy for learning new ways to refer to known perceptual concepts (Yu et al., 2017a). The perceptual concepts are basic, and dialogue policy is learned through reinforcement learning from a dataset of human-human conversations. Yu et al. (2016) demonstrate the importance of taking initiative, processing and expressing perceptual concepts, and understanding ellipsis for the same task.

More generally, natural language can be used to aid learning from demonstration. She and Chai (2017) learn a system that uses a hierarchical knowledge base over actions to compose simple action primitives into complex ones. This is learned from human demonstrations paired augmented by language descriptions, where the robot learns a dialog policy to ask clarifications for noun phrase grounding, effects of an action, states of objects and whether actions are necessary to achieve a goal.

In our completed work (chapter 4), we introduce the setting of opportunistic active learning – a framework for interactive tasks that involve learning of supervised models. This framework allows a robot to ask more diverse queries across interactions, and requires the robot to trade-off between task completion and knowledge acquisition for future tasks.

2.6 Semantic Parsing

Semantic parsing maps a natural language sentence such as ”Go to Alice’s office” to a machine understandable meaning representation. In our work, we use λ-calculus logical forms such as:

$$\texttt{navigate}(\texttt{the}(\lambda x.\texttt{office}(x) \wedge \texttt{possess}(\texttt{alice}, x) \wedge \texttt{person}(\texttt{alice})))$$

This represents that the robot should navigate to a place x which is an office, and owned by a person `alice`.

This formalism reduces the number of lexical entries the system needs to learn by exploiting compositional reasoning over language. For example, if it learns that “Alice Ashcraft” also refers to the entity `alice`, it does not need to learn another lexical entry for “Alice Ashcraft’s office”.

There has been considerable work in semantic parsing using direct supervision in the form of annotated meaning representations (Wong and Mooney, 2007;

Kwiatkowski et al., 2013; Berant et al., 2013). More recent works use indirect signals from downstream tasks. Artzi and Zettlemoyer (2011) use clarification dialogs to train semantic parsers for an airline reservation system without explicit annotation of meaning representations. Thomason et al. (2015), incorporate this general approach into a system for instructing a mobile robot using a basic dialog state and fixed hand-coded policy.

In our work (chapter 3), semantic parsing is performed using probabilistic CKY-parsing with a Combinatory Categorial Grammar (CCG) (Steedman and Baldridge, 2011) and meanings associated with lexical entries (Zettlemoyer and Collins, 2005). Perceptron-style updates to parameter values are used during training to weight parses to speed search and give confidence scores in parse hypotheses.

2.7 Reinforcement Learning

Reinforcement learning is a computational process of learning to map situations to actions to maximize a numerical reward signal (Sutton et al., 1998). In a reinforcement learning problem, an agent interacts with its environment to achieve a goal. The agent can sense the state of the environment, take actions that affect the state, and have one or more goals related to this state. It typically needs to take a sequence of actions to achieve its goal and may receive only delayed numerical feedback (reward) to indicate whether progress has been made.

Reinforcement learning faces the challenge of trading off exploration and exploitation. The agent has to *exploit* actions known to be effective, to obtain reward, but must *explore* new actions to find out those that are the most effective.

We now discuss two common formulations of reinforcement learning problems (sections 2.7.1 and 2.7.2), and three algorithms used for policy learning (sections 2.7.3, 2.7.4 and 2.7.6).

2.7.1 Markov Decision Process (MDP)

A Markov Decision Process (MDP) is a tuple $\langle \mathbb{S}, \mathbb{A}, \mathbb{T}, \mathbb{R}, \gamma \rangle$, $\mathbb{S}$ is a set of states, $\mathbb{A}$ is a set of actions, $\mathbb{T}$ is a transition function, $\mathbb{R}$ is a reward function and γ is a discount factor. At any instant of time t, the agent is in a state $s_t \in \mathbb{S}$. It chooses to take an action $a_t \in \mathbb{A}$ according to a policy π, commonly represented as a probability distribution over actions where $\pi(a_t|s_t)$ is the probability of taking action a_t when the agent is in state s_t. On taking action a_t, the agent is given a real-valued reward r_t and transitions to a state s_{t+1}.

State transitions occur according to the probability distribution $P(s_{t+1}|s_t, a_t) = \mathbb{T}(s_t, a_t, s_{t+1})$, and rewards obtained follow the distribution $P(r_t|s_t, a_t) = \mathbb{R}(s_t, a_t, s_{t+1})$.

The objective is to identify a policy π that is optimal in the sense that it maximizes the expected long term discounted reward, called return, given by

$$g = \mathbb{E}_\pi \left[\sum_{t=1}^{\infty} \gamma^t r_t \right]$$

2.7.2 Partially Observable Markov Decision Process (POMDP)

A Partially Observable Markov Decision Process (POMDP) is an extension of MDPs where the agent does not know what state it is in, but only receives a noisy observation indicative of the state.

Formally, a POMDP is a tuple $(\mathbb{S}, \mathbb{A}, \mathbb{T}, \mathbb{R}, \mathbb{O}, \mathbb{Z}, \gamma, b_0)$, where $\mathbb{S}$ is a set of states, $\mathbb{A}$ is a set of actions, $\mathbb{T}$ is a transition function, $\mathbb{R}$ is a reward function, $\mathbb{O}$ is a set of observations, $\mathbb{Z}$ is an observation function, γ is a discount factor and b_0 is an initial belief state (Kaelbling et al., 1998).

These are defined as in MDPs, but the the state s_t is hidden from the agent and only a noisy observation $o_t \in \mathbb{O}$ of s_t is available to it. The agent maintains a belief state b_t which is a distribution over all possible states it could be in at time t. $b_t(s_i)$ gives the probability of being in state s_i at time t. The agent chooses actions $a_t \in \mathbb{A}$ based on b_t, according to a policy π. On taking action a_t, the agent is given a real-valued reward r_t, transitions to a state s_{t+1}, and receives a noisy observation o_{t+1} of s_{t+1}, which is used to update its belief b_{t+1}.

State transitions occur according to the probability distribution $P(s_{t+1}|s_t, a_t) = \mathbb{T}(s_t, a_t, s_{t+1})$, observations are related to the states by the probability distribution $P(o_t|s_t, a_{t-1}) = \mathbb{Z}(o_t, s_t, a_{t-1})$ and rewards obtained follow the distribution $P(r_t|s_t, a_t) = \mathbb{R}(s_t, a_t, s_{t+1})$.

The objective, again, is to identify a policy π that is optimal in the sense that it maximizes return.

2.7.3 REINFORCE Algorithm

The REINFORCE algorithm (Williams, 1992) is a simple policy gradient algorithm used to learn a policy in an MDP. The agent learns a policy $\pi(a|s;\theta)$, parameterized with weights θ that computes the probability of taking action a in state s. An example is a policy based on a feature representation $f(s, a)$ for a state-action pair (s, a):

$$\pi(a|s;\theta) = \frac{e^{\theta^T f(s,a)}}{\sum_{a'} e^{\theta^T f(s,a')}}$$

where the denominator is a sum over all actions possible in state s.

The weights are updated using a stochastic gradient ascent rule:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

where $J(\theta)$ is the expected return from the policy according to the distribution over trajectories induced by the policy.

2.7.4 Q-Learning

The quality of a policy π can be estimated using the action value function

$$Q^\pi(s,a) = \mathbb{E}_\pi \left[\sum_{t=1}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a \right]$$

The optimal policy satisfies the Bellman equation,

$$Q^*(s,a) = \mathbb{E}_{s'} \left[\mathbb{R}(s,a,s') + \gamma max_{a' \in \mathbb{A}} Q^*(s',a') \right]$$

For a POMDP, the above equations would be in terms of belief states b.

Q-learning is a temporal difference method used for policy learning. The algorithm starts off with possibly arbitrary estimates for $Q(s,a)$ and attempts to update them towards $Q^*(s,a)$ through experience. This experience can be collected using any policy, and hence, the algorithm is an off-policy algorithm. The following update is performed when the agent takes action a_t in state s_t, receiving reward r_t, and transitioning to s_{t+1}.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} Q(s_{t+1}, a'))$$

Given the final estimates $\hat{Q}(s, a)$, the corresponding learned policy would be to take the action of highest estimated value at each state. That is,

$$\pi(a|s) = \max_a Q(s, a)$$

2.7.5 Asynchronous Advantage Actor Critic

Actor-Critic algorithms use a combination of an actor policy $\pi_\theta(a|s)$ trained using policy gradient methods, with a critic that estimates the value of each state, $V(s)$, using value-iteration which can then be used to compute an advantage function $A(s, a) = Q(s, a) - V(s)$, which is used to reduce the variance in the policy gradient update.

Then, given a trajectory where the agent takes action a_t in state s_t at time step t, receiving reward r_t, and transitioning to s_{t+1}, the policy gradient estimate used is

$$\nabla_\theta J(\theta) = \sum_t \nabla_\theta \log \pi_\theta(a_t|s_t) A(s_t, a_t)$$

where $A(s_t, a_t)$ is estimated using the critic as

$$A(s_t, a_t) = r_t + \gamma V(s_{t+1}) - V(s_t)$$

The actor can then be updated using policy gradient:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

and then the critic is updated using value iteration:

$$V(s_t) = r_t + \gamma V(s_{t+1})$$

In Asynchronous Advantage Actor Critic (Mnih et al., 2016), the actor and critic are parameterized using neural networks that share all layers except the last. The final layer for the actor predicts next-action probabilities and that of the critic predicts values. Further, the same set of parameters is used by a number of agents in parallel to collect trajectories, and updates from these are used to asynchronously update the shared model.

2.7.6 KTD-Q Learning

When the state space is very large or continuous, Q^π cannot be computed for each state (or belief state) individually and is hence assumed to be a function with parameters θ over some features that represent the state. When the transition or reward dynamics are not constant (*non-stationary* problem), a suitable approximation is the Kalman Temporal Differences framework (Geist and Pietquin, 2010). This casts the function approximation as a filtering problem and solves it using Kalman filtering. The specialization for learning the optimal action value function is called the KTD-Q algorithm.

Filtering problems estimate hidden quantities X from related observations Y, modeling X and Y as random variables. When estimating action values, X corresponds to the function parameters, θ and the observations are the estimated returns, $r_t + \gamma \max_a \hat{Q}_{\theta_t}(s_{t+1}, a)$, and a random noise is added to both of these to allow for parameters to change over time. The update rules are derived from Kalman Filtering Theory and details can be found in Geist and Pietquin (2010).

2.8 Active Learning

In machine learning tasks where obtaining labeled examples is expensive, active learning is used to lower the cost of annotation without sacrificing model performance. Active learning allows a learner to iteratively query for labels of unlabeled data points that are expected to maximally improve the existing model. Research in active learning attempts to identify examples that are likely to be the most useful in improving a supervised model. A number of metrics have been proposed to evaluate examples, including uncertainty sampling (Lewis and Gale, 1994), density-weighted methods (Settles and Craven, 2008), expected error reduction (Roy and McCallum, 2001), query by committee (Seung et al., 1992), and the presence of conflicting evidence (Sharma and Bilgic, 2016); as surveyed by Settles (2010).

Multilabel active learning is the application of active learning in scenarios where multiple labels, that are not necessarily mutually exclusive, are associated with a data point (Brinker, 2006). These setups often suffer from sparsity, both in the number of labels that are positive for a data point, and in the number of positive data points per label. Standard active learning metrics are often extended to the multilabel setting, by assuming that one-vs-all classifiers are learned for each label, and that all the learned classifiers are comparable (Brinker, 2006; Singh et al., 2009; Li et al., 2004). Label statistics have also been incorporated into heuristics for selecting instances to be queried (Yang et al., 2009; Li and Guo, 2013). There have also been Bayesian approaches that select both an instance and label to be queried (Qi et al., 2009; Vasisht et al., 2014).

The most commonly used framework for active learning is pool-based active learning, where the learner has access to the entire pool of unlabeled data at once, and can iteratively query for examples. In contrast, sequential active learning is a framework in which unlabeled examples are presented to the learner in a stream (Lewis and Gale, 1994). For every example, the learner can decide whether to query for its label or not. This results in an additional challenge – since the learner cannot compare all unlabeled data points before choosing queries, each query must be chosen based on local information only. We introduce the framework of Opportunistic Active Learning (chapter 4) that extends sequential active learning to an interactive multi-label task.

Recently, there has been interest in using reinforcement learning to learn a policy for active learning. Fang et al. (2017) use deep Q-learning to acquire a policy that sequentially examines unlabeled examples and decides whether or not to query for their labels; using it to improve named entity recognition in low resource languages. Also, Bachman et al. (2017) use meta-learning to jointly learn a data selection heuristic, data representation and prediction function for a distribution of related tasks. They apply this to one shot recognition of characters from different languages, and in recommender systems. Woodward and Finn (2017) use reinforcement learning with a recurrent-neural-network-based Q-function in a sequential one-shot learning task to decide between predicting a label and acquiring the true label at a cost. We follow in this line of work to learn a policy for opportunistic active learning in a task of grounding natural language descriptions of objects (chapter 5).

2.9 Grounding Language in Perception

When humans interact with robots in natural language, they typically refer to entities in the real world, and expect robots to be able to identify these referents. For many types of entities, such as physical household objects, people typically describe them in terms of attributes such as object category, color and weight (Guadarrama et al., 2016; Thomason et al., 2016). Robots need to be able to perceive properties that humans refer to, and use these to map referring expressions to referents in the real world. This task is an instance of the symbol grounding problem (Harnad, 1990).

There has been considerable work on extending word representations to incorporate visual context. These are found to be useful for predicting lexical similarity (Silberer and Lapata, 2012, 2014; Lazaridou et al., 2015), verifying common sense assertions (Kottur et al., 2016), verifying visual paraphrasing (Kottur et al., 2016), image categorization (Silberer and Lapata, 2014; Lazaridou et al., 2015) including in the zero shot setting (Lazaridou et al., 2014), and retrieval of related images (Kottur et al., 2016). However, these do not attempt to retrieve images or objects based on free-form natural language descriptions – as desired in robotics applications.

Guadarrama et al. (2016) assemble a dataset for retrieval of objects based on open vocabulary natural language descriptions, and compare the performance of image categorization and instance recognition methods, as well as ensembles of these on this task. Misra et al. (2017b) learn a network to more intelligently compose classifiers learned for adjectives and nouns. Hu et al. (2016) propose a neural network model that uses a vector representation of a region crop, the entire image, and relative bounding box coordinates to score regions in an image to identify the one referred to by a natural language expression. Other works either align vector representations of images and descriptions/ captions using methods such as CCA (Feng et al., 2015), or learn a joint embedding of the modalities (Wang et al., 2016) to perform image-to-caption and caption-to-image retrieval. Xiao et al. (2017) learn to ground descriptions in images by learning mappings from phrases to attention vectors over the image, and combining attended regions using linguistic constraints. Hu et al. (2018b) use composable neural network modules paired with a differentiable stack memory to convert referring expressions into differentiable executable programs, that can be used to ground more complex visual relationships in referring expressions. Burns et al. (2019) demonstrate that using pooling methods based on Fisher vectors, and injecting knowledge into word embeddings results in a better language representation for many language grounding tasks than pretrained context-sensitive sentence representations. Some other models for such language grounding include language guided graph attention (Wang et al., 2019) and multimodal transformers (Ye et al., 2019). The performance of multi-modal transformers can be further improved by pretraining on large datasets of paired language and images (Lu et al., 2019; Tan and Bansal, 2019).

There are also works that focus on learning other aspects of grounding including spatial relations (Bisk et al., 2016; Yang and Narasimham, 2020), relative properties such as size, weight and rigidity of object pairs (Forbes and Choi, 2017), subject-relation-object triples (Hu et al., 2017), meanings of verbs modeled as state changes (Gao et al., 2016; Liu et al., 2016a; Gao et al., 2018), and semantic roles of a verb in videos (Yang et al., 2016). Grounding of object descriptions can also be improved by incorporating information such as temporal context and gesture (Williams et al., 2017). There have also been attempts to learn unsupervised alignments between words in speech and objects in images (Harwath et al., 2018).

While most work on grounded language learning focuses on understanding, there is also work on generating referring expressions of objects for effective human-robot communication (Fang et al., 2013, 2014).

2.10 Visually Grounded Dialog

Recently, a few new dialog tasks and datasets have been introduced that require grounding of language in images. The VisDial dataset was collected to teach a robot to coherently answer a sequence of questions about a single image (Das et al., 2017a). This has also been used to train a pair of agents, one of which asks questions about an unseen image, and another that answers them using the image (Das et al., 2017b), with the goal of the questioning agent attempting to learn a representation of what the image looks like. They find that the answering agent does provide answers similar to humans, and only pretraining constrains the agents to retain the semantics of English as used by humans. Zhang et al. (2018a) create a 20-questions game using this dataset and learn a hierarchical dialog policy where the top level decided between asking questions and guessing, while a lower level policy chooses the exact question to be asked.

Another related work is the GuessWhat?! dataset (De Vries et al., 2017) of humans playing a 20 questions game to identify an object in images of rich scenes. This is used to train a questioner that tries to identify the target object by asking yes/no questions, and the oracle that learns to answer them. It is difficult to evaluate the success of either agent, as ground truth for both agents is unlikely to be present in the training set. Further, if they are trained jointly, the challenge of retaining human semantics again arises. Another work that learns to ask questions that can discriminate between images is Li et al. (2017).

We learn a system that can learn clarification questions that can refine on an initial description (chapter 6) – which is not present in the above tasks. Further, we wish to do this in a setting where we can provide ground truth answers to questions of the system during training. Hence we use binary questions about the presence of visual attributes such as colors, object categories, and other domain specific attributes, to perform clarification.

Some related recent tasks are interactive versions of visual navigation using natural language route instructions. Some examples include adding an extra “clarification” action that allows the agent to obtain the next navigation step using an oracle (Chi et al., 2020; Nguyen and III, 2019). While these setups do allow for providing ground-truth answers to clarification questions, they simulate a somewhat awkward interaction as a user would not typically reason at the level of low-level navigation steps. Thomason et al. (2020) create a more realistic interactive instruction following task, where the robot must learn to ask follow up questions to clarify ambiguous route instructions. However, since the clarification questions are unconstrained, it is difficult to ensure that correct answers are provided to the agent during training in simulation.

2.11 Interactive Image Retrieval

Early work on interactive image retrieval requires initial images to be selected to start a search from a sample of images, and users are allowed to refine results by iteratively identifying relevant and irrelevant results (Nastar et al., 1998; Tieu and Viola, 2004). In these cases, retrieval was directed solely using the initial selections, not using natural language search queries.

Over time, there has been a large body of work in retrieving images using natural language queries (Guadarrama et al., 2014). Caption-to-image retrieval on image captioning datasets has also been used as a testbed for many language grounding models (Wang et al., 2016).

Other works attempt to augment such an initial retrieval with further interaction, often motivated by applications in online shopping. Robotics applications have long used visual attributes as a means of asking clarification questions for refining natural language object retrieval. Early systems either assume access to oracle predictions of visual attributes (Liu et al., 2013) or use handcrafted features for classifying visual attributes (Dindo and Zambuto, 2010; Parde et al., 2015). In shopping applications, more focus has been placed on allowing users to provide relative feedback – allowing users to specify how they would like the retrieval results to differ from the current results. Kovashka et al. (2012) allow users to refine results indicating relative feedback along specific attributes, for examples requesting for "more formal" shoes. More recently, the task of relative captioning (Guo et al., 2018; Bhattacharya et al., 2019) allows users to provide such feedback using unconstrained natural language. Saha et al. (2018) extend this to incorporate providing additional information and answering questions about products. Recent works use convolutional neural networks to perform visual attribute prediction (Liu et al., 2016b; Guo et al., 2019). A complementary direction is for the system to take an initiative using similar visual attributes to narrow down the search space for the user, which we explore in our work on attribute based clarifications (chapters 6 and 8).

Chapter 3

Integrated Learning of Dialog Strategies and Semantic Parsing

Robots need to be able to understand high-level natural language commands to be accessible to a variety of users. Since the types of commands, and language usage vary across domains, it is desirable that a robot should be able to improve through interaction with users in its operating environment. For an interactive dialog system, prior work had demonstrated different methods to independently improve either the natural language understanding component or the dialog strategy. In this chapter, We discuss an approach to integrate the learning of both a dialog strategy using reinforcement learning, and a semantic parser for robust natural language understanding, using only natural dialog interaction for supervision. The main challenge involved is choosing an appropriate reinforcement learning algorithm, and training procedure, as the simultaneous training of the semantic parser violates the assumption of a non-stationary environment, made by most reinforcement learning algorithms.

3.1 Contributions

This work was originally presented in Padmakumar et al. (2017). My contribution to this work included designing and implementing the joint learning framework, experimenting with reinforcement learning algorithms for policy learning, and implementing and conducting the final experiments. Jesse Thomason provided the implementation of the semantic parser and a template for the web interface used in the final experiments, and assisted with other issues that arose during implementation. Prof Raymond Mooney proposed the project and provided guidance in the choice of algorithms and experimental conditions considered.

3.2 Motivation

Prior research in dialog systems was primarily focused on the problems of accurate dialog state tracking (Young et al., 2010; Thomson and Young, 2010; Mrkšić et al., 2015; El Asri et al., 2016) and learning a policy for the dialog system to respond appropriately in various scenarios (Gašić and Young, 2014; Pietquin et al., 2011; Png et al., 2012). Dialogs are typically modeled using Partially Observable Markov Decision Processes (POMDPs), and various reinforcement learning algorithms have been proposed and evaluated for the task of learning optimal policies over these representations to accomplish user goals effectively and efficiently (Gašić and Young, 2014; Pietquin et al., 2011; Young et al., 2013). However, such systems typically assumed a fixed language understanding component that is available a priori.

Semantic parsing is the task of mapping natural language to a formal meaning representation. It has the potential to allow for more robust mapping of free-form natural language to a representation that can be used to interpret user intentions and track dialog state. It leverages the compositionality of meaning inherent in language to perform more robust language understanding. There has been considerable work in semantic parsing that use both direct supervision in the form of annotated meaning representations (Wong and Mooney, 2007; Kwiatkowski et al., 2013; Berant et al., 2013) and indirect signals from downstream tasks (Artzi and Zettlemoyer, 2011, 2013; Thomason et al., 2015). In particular, Thomason et al. (2015) show that a semantic parser, incrementally updated from conversations, is helpful in dialogs for communicating commands to a mobile robot. In this chapter, we show that incremental learning of a POMDP-based dialog policy allows for further improvement in dialog success.

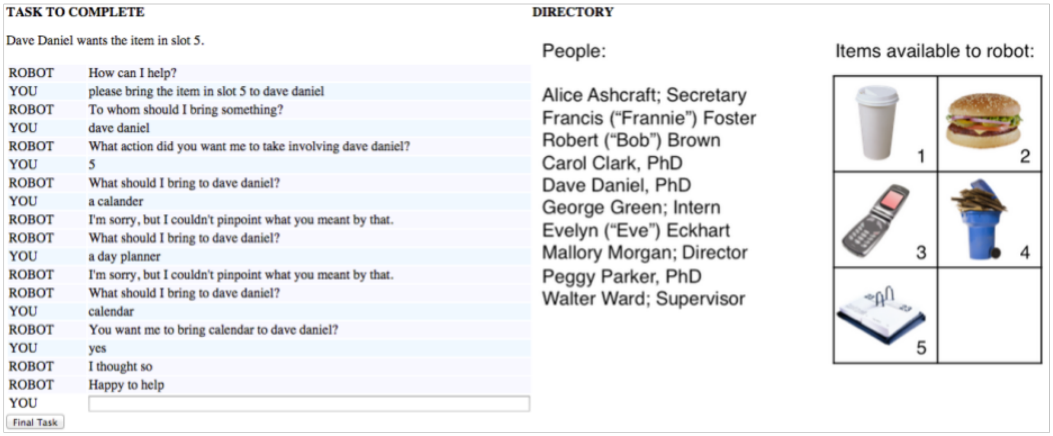
A major challenge with this setup is that ongoing parser learning results in a violation of a common assumption of *a stationary environment* made by reinforcement learning (RL) algorithms. For example, the improved semantic parser may be able to extract more information from a response to a question, which the old parser could not parse. So the RL algorithm may have assumed that asking that question is not useful, but that is not the case with the new parser. Our results show that this effect can be mitigated if we break the allowed budget of training dialogs into batches, updating both parser and policy after each batch. As the next training batch gets collected using the updated parser, the policy can be updated using this experience to adapt better to it. We demonstrate, using crowd-sourced results with a simulated robot, that by integrating learning of *both* a dialog manager *and* a semantic parser in this manner, task success is improved over cases where the components are trained individually.

We also use a reinforcement learning algorithm that is explicitly designed to model the non-stationary environment by using techniques from Kalman Filtering theory to model how the transition and reward functions change over time (Geist and Pietquin, 2010). Some alternative methods to model non-stationary RL environments include the use of Gaussian processes (Engel et al., 2005) and Bayes-Adaptive POMDPs (Ross et al., 2008). Another alternative is to use RL algorithms such as Double Q-learning that do not explicitly model the non-stationary environment but use batching techniques similar to our proposed method to avoid overestimation of Q-values (Hasselt, 2010). We leave the comparison of these different RL algorithms in this setting to future work.

Since the original publication of this work, there has been more interest in end-to-end trained dialog systems (Wen et al., 2017; Serban et al., 2016). The training procedure of such systems is automatically incorporates the ideas we demonstrate in this chapter – that it is helpful to update all components of a dialog system with information from dialogs, and this is more effective when training is done in multiple batches so that changes in components can be made visible to other components.

3.3 Task and System Setup

Our goal is to develop a dialog system that enables a robot to translate high level commands such as *"go to Alice's office"* to a command represented as an action with associated semantic roles. The user can command the system to perform two actions: navigation and delivery. Navigation has a single parameter for the destination. For example *go to Alice's office* would be a possible way to command



TASK TO COMPLETE

Dave Daniel wants the item in slot 5.

ROBOT	How can I help?
YOU	please bring the item in slot 5 to dave daniel
ROBOT	To whom should I bring something?
YOU	dave daniel
ROBOT	What action did you want me to take involving dave daniel?
YOU	5
ROBOT	What should I bring to dave daniel?
YOU	a calander
ROBOT	I'm sorry, but I couldn't pinpoint what you meant by that.
ROBOT	What should I bring to dave daniel?
YOU	a day planner
ROBOT	I'm sorry, but I couldn't pinpoint what you meant by that.
ROBOT	What should I bring to dave daniel?
YOU	calendar
ROBOT	You want me to bring calendar to dave daniel?
YOU	yes
ROBOT	I thought so
ROBOT	Happy to help
YOU	

DIRECTORY

People:

Alice Ashcraft; Secretary
Francis ("Frannie") Foster
Robert ("Bob") Brown
Carol Clark, PhD
Dave Daniel, PhD
George Green; Intern
Evelyn ("Eve") Eckhart
Mallory Morgan; Director
Peggy Parker, PhD
Walter Ward; Supervisor

Figure 3.1: A sample dialog in our interface on Amazon Mechanical Turk.

the robot to perform a mavigation command whose location is a room that is the office of a person *alice*. Delivery has two parameters: the item to be delivered and the person to receive it. For example, *bring Alice a hamburger* would be a possible way to specify a delivery command whose patient is an item *hamburger* and recipient is a person *alice*.

Our system initiates the dialog by requesting the user for a command. The robot makes an initial guess of the desired action from the user's response, and then may ask clarification questions in case of insufficient understanding. At each step, it can respond with one of four dialog acts: asking the user to repeat their command, confirming a command or an argument value, requesting a specific argument of a command, and executing an action (thereby ending the dialog). A sample dialog is shown in figure 3.1.

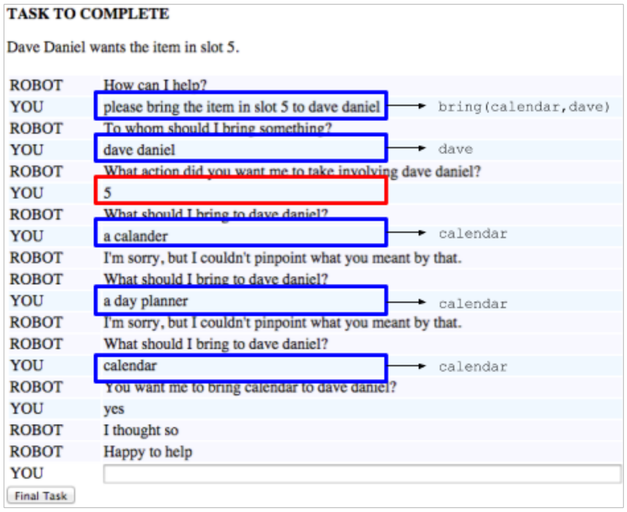
Given a user response, a semantic parser is used to convert utterances to logical forms (section 2.6) which are then grounded to actions and semantic roles using world knowledge (e.g. resolving the representation of "Alice's office" to a particular room). The robot maintains a belief state that represents the probability that a particular combination of action and semantic roles is the command intended by the user. This is updated based on the grounded response, and is then used by the dialog policy to choose the next action to be taken by the robot. Once the conversation is complete, the parser and policy can be updated from the dialog (sections 3.4 and 3.5 respectively).

The dialog is considered a success if the final action taken is correct and a failure otherwise. The user also has the ability to prematurely end the dialog, and any conversation terminated in this manner is also considered a failure.

3.4 Semantic Parser Learning

The semantic parser is trained using paired sentences and logical forms. A small supervised training set is used to initialize the parser. Training continues using pairs obtained through weak supervision collected from user dialogs (Thomason et al., 2015).

Figure 3.2 shows an example of the training pairs induced from the example dialog. To obtain these, we obtain multiple semantic parses for these responses, and parses are syntactically valid, and that ground to the action finally taken by the robot or its arguments, are paired with the response to training pairs. These paired responses and semantic forms can then be used to retrain the parser between conversations. While this weak supervision may be noisy, the syntactic and grounding constraints remove most spurious examples. For example, if we receive "Go to



TASK TO COMPLETE

Dave Daniel wants the item in slot 5.

ROBOT	How can I help?
YOU	please bring the item in slot 5 to dave daniel
ROBOT	To whom should I bring something?
YOU	dave daniel
ROBOT	What action did you want me to take involving dave daniel?
YOU	5
ROBOT	What should I bring to dave daniel?
YOU	a calander
ROBOT	I'm sorry, but I couldn't pinpoint what you meant by that.
ROBOT	What should I bring to dave daniel?
YOU	a day planner
ROBOT	I'm sorry, but I couldn't pinpoint what you meant by that.
ROBOT	What should I bring to dave daniel?
YOU	calendar
ROBOT	You want me to bring calendar to dave daniel?
YOU	yes
ROBOT	I thought so
ROBOT	Happy to help
YOU	

Figure 3.2: Training pairs induced from the clarification dialog, by selecting parses that ground to the final action and its arguments. The response in red is discarded because no such parse is found.

Bob's office" as a response when we expect an argument value, since the response is an imperative sentence, not a noun phrase such as "Bob's office", no training example would be generated from it. Also, if we knew from a successful dialog that the desired goal location is Bob's office, we would ignore intermediate responses that appear to parse to other locations.

3.5 Dialog Policy Learning

We use a POMDP to model dialog and learn a policy (section 2.2), adapting the Hidden Information State model (HIS) (Young et al., 2010) to track the belief state as the dialog progresses. The key idea behind this approach is to group states into equivalence classes called partitions, and maintain a probability for each partition instead of each state. States within a partition are those that are indistinguishable to the system given the current dialog.

More concretely, our belief state can be factored into two main components - the goal intended by the user $\mathbf{g}$ and their most recent utterance $\mathbf{u}$. A partition p is a set of possible goals which are equally probable given the conversation so far.

After every user response, a beam of possible choices for $\mathbf{u}$ can be obtained by grounding the beam of top-ranked parses from the semantic parser. Grounding is performed by looking up a knowledge base of entities such as people, and relations such as who owns an office. Given the previous system action $\mathbf{m}$, The belief $b(p, \mathbf{u})$ is calculated as in the HIS model as follows

$$b(p, \mathbf{u}) = k * P(\mathbf{u}) * T(\mathbf{m}, \mathbf{u}) * M(\mathbf{u}, \mathbf{m}, p) * b(p)$$

Here, $P(\mathbf{u})$ is the probability of the utterance hypothesis $\mathbf{u}$ given the user response, which is obtained from the semantic parser. $T(\mathbf{m}, \mathbf{u})$ is a 0-1 value indicating whether the response is relevant given the previous system question, determined from the semantic type of the response. $M(\mathbf{u}, \mathbf{m}, p)$ is a 0-1 value indicating whether goals in partition p are relevant to the response and previous system question. $b(p)$ is the belief of partition p before the update, obtained by marginalizing out $\mathbf{u}$ from $b(p, \mathbf{u})$. k is a normalization constant that allows the expression to become a valid probability distribution.

We extract features from the belief state to form a summary space over which a dialog policy is learned as in prior work (Young et al., 2010; Gašić and Young, 2014). Table 3.1 contains the features used to learn the policy.

The choice of policy learning algorithm is important because learning POMDP

Table 3.1: Features used in summary space.

Probability of top hypothesis
Probability of second hypothesis
Number of goals allowed by the partition in the top hypothesis
Number of parameters of the partition in the top hypothesis, required by its action, that are uncertain (set to the maximum value if there is more than one possible action)
Number of dialog turns used so far
Do the top and second hypothesis use the same partition (0-1)
Type of last user utterance
Action of the partition in the top hypothesis, or *null* if this is not unique

policies is challenging and dialog applications exhibit properties not often encountered in other reinforcement learning applications (Daubigney et al., 2012). We use Kalman Temporal Difference Q-learning (Geist and Pietquin, 2010), or KTD-Q, to learn the dialog policy as it was designed to satisfy some of these properties and tested in a dialog system with simulated users (Pietquin et al., 2011). The properties we wished to be satisfied by the algorithm were the following:

- Low sample complexity in order to learn from limited user interaction.
- An off-policy algorithm to enable the use of existing dialog corpora to bootstrap the system, and crowdsourcing platforms such as Amazon Mechanical Turk during training and evaluation.
- A model-free rather than a model-based algorithm because it is difficult to design a good transition and observation model for this problem (Daubigney et al., 2012).

- Robustness to non-stationarity because the underlying language understanding component changes with time (Section 3.4), which is likely to change state transitions.

To learn the policy, we provided a high positive reward for correct completion of the task and a high negative reward when the robot chose to execute an incorrect action, or if the user terminated the dialog before the robot was confident about taking an action. The system was also given a per-turn reward of -1 to encourage shorter dialogs.

3.6 Experimental Setup

The semantic parser was initialized using a small seed lexicon and trained on a small set of supervised examples constructed using templates. The dialog policy was initialized with an approximation of a good static policy.

3.6.1 Task Interface

Our experiments were done through Mechanical Turk as in previous work (Thomason et al., 2015; Wen et al., 2017). The setup is shown in figure 3.1. During the training phase, each user interacted with one of four dialog agents (described in section 3.6.2), selected uniformly at random. Users were not told of the presence of multiple agents and were not aware of which agent they were interacting with. They were given a prompt for either a navigation or delivery task and were asked to have a conversation with the agent to accomplish the given task. No restrictions were placed on the language they could employ. We use visual prompts for the tasks to avoid linguistic priming (e.g. a picture of a hamburger instead of the word "hamburger"). Training dialogs were acquired in 4 batches of 50 dialogs each across all agents. After each batch, agents were updated as described in section 3.6.2.

A final set of 100 test conversations were then conducted between Mechanical Turk users and the trained agents. These test tasks were novel in comparison to the training data in that although they used the same set of possible actions and argument values, the same combination of action and argument values had not been seen at training time. For example, if one of the test tasks involved delivery of a `hamburger` to `alice`, then there may have been tasks in the training set to deliver a `hamburger` to other people and there may have been tasks to deliver other items to `alice`, but there was no task that involved delivery of a `hamburger` to `alice` specifically.

3.6.2 Dialog agents

We compared four dialog agents. The first agent performed only parser learning (described in Section 3.4). Its dialog policy was always kept the static policy used to initialize the KTD-Q algorithm. Its parser was incrementally updated after each training batch. This agent is similar to the system used by Thomason et al. (2015) except that it uses the same state space as our other agents, and multiple hypotheses from the parser, for fairer comparison.

The second agent performed only dialog policy learning. Its parser was always kept to be the initial parser that all agents started out with. Its policy was incrementally updated after each training batch using the KTD-Q algorithm. The third agent performed both parser and dialog learning; but instead of incrementally updating the parser and policy after each batch, they were trained at the end of the training phase using dialogs across all batches. This would not allow the dialog manager to see updated versions of the parser in batches after the first and adapt the policy towards the improving parser. We refer to this as *full* learning of parser and dialog policy. The fourth agent also performed both parser and dialog learning. Its parser and policy were updated incrementally after each training batch. Thus for the next training batch, the changes due to the improvement in the parser from the previous batch could, in theory, be demonstrated in the dialogs and hence contribute towards updating the policy in a manner consistent with it. We refer to this as *batchwise* learning of parser and dialog policy.

3.6.3 Experiment hypothesis

We hypothesized that the agent performing *batchwise* parser and policy learning would outperform the agents performing only parser or only dialog learning as we expect that improving both components is more beneficial. However, we did not necessarily expect the same result from *full* parser and dialog learning because it did not provide any chance to allow updates to propagate even indirectly from one component to another, exposing the RL algorithm to a more *non-stationary* environment. Hence, we also expected *batchwise* learning to outperform *full* learning.

3.7 Results and Discussion

The agents were evaluated on the test set using the following objective performance metrics: the fraction of successful dialogs and the length of successful dialogs. We also included a survey at the end of the task asking users to rate on a 1–5 scale whether the robot understood them, and whether they felt the robot asked sensible questions.

Table 3.2 gives the agents' performance on these metrics. All differences in dialog success and the subjective metrics are statistically significant according to an unpaired t-test with $p < 0.05$. In dialog length, the improvement of the *batchwise* learning agent over the agents performing only parser or only dialog learning are statistically significant.

Table 3.2: Performance metrics for dialog agents tested. Differences in dialog success and subjective metrics are statistically significant according to an unpaired t-test with $p < 0.05$.

Learning involved	% successful dialogs	Average dialog length	Robot understood	Sensible questions
Parser	75	12.43	2.93	2.79
Dialog	59	11.73	2.55	2.91
Parser & Dialog - *full*	72	12.76	2.79	**3.28**
Parser & Dialog - *batchwise*	**78**	**10.61**	**3.30**	3.17

As expected, the agent performing *batchwise* parser and dialog learning outperforms the agents performing only parser or only dialog learning, in the latter case by a large margin. We believe the agent performing only parser learning performs much better than the agent performing only dialog learning due to the relatively high sample complexity of reinforcement learning algorithms in general, especially in the partially observable setting. In contrast, the parser changes considerably even from a small number of examples. Also, we observe that *full* learning of both components does not in fact outperform only parser learning. We believe this is because the distribution of hypotheses obtained using the initial parser at training time is substantially different from that obtained using the updated parser at test time. We believe that *batchwise* training mitigates this problem because the distribution of hypotheses changes after each batch of training and the policy when updated at these points can adapt to some of these changes. The optimal size of the batch is a question for further experimentation. Using a larger batch is less likely to overfit updates to a single example but breaking the total budget of training dialogs into more batches allows the RL algorithm to see less drastic changes in the distribution of hypotheses from the parser.

With dialog policy learning, a qualitative change observed is that the system tends to confirm or act upon lower probability hypotheses than is recommended by the initial hand-coded policy. This is possibly because as the parser improves, its top hypotheses are more likely to be correct.

We also observe quantitative improvements in parser accuracy for agents whose parsers were trained. To attempt to quantify the improvement in accuracy of the parsers after training from dialog, we manually annotated semantic forms for commands from the test set. We evaluated the final parser used for testing in each of the conditions in terms of $Recall@1$, which is the fraction of times the correct parse is the top parse predicted by the parser, and $Recall@10$, which is the fraction of times the correct parse occurs in the top 10 parses predicted by the parser. The results are included in table 3.3.

Table 3.3: Comparison of performance of initial parser and parsers after updating various components, on paired commands and semantic forms. $*$ indicates that the difference in performance between this and the *Initial* parser on the same metric is statistically significant according to a paired t-test with $p < 0.05$ and ˆ indicates that the difference is trending significance ($p < 0.1$)

Learning involved	Recall@1	Recall@10
None	0.564	0.611
Only parser	0.588	**0.671***
Only dialog	0.564	0.623
Parser and dialog - *full*	0.588	**0.647** ˆ
Parser and dialog - *batchwise*	0.576	**0.670***

As expected, we observe that the initial parser (no learning) and the parser from the system performing only dialog learning, perform worse than the others, as the other systems update the parser used by these. The parser of the system performing only dialog learning is in fact a copy of the initial parser and was included only for completeness. Any difference in their performance is due to randomness. The parsers updated from dialogs improve in accuracy but the differences are found to be statistically significant only on $Recall@10$. The modest improvement is unsurprising given that the supervision provided is both noisy and weak. However, even this modest improvement is sufficient to improve overall dialog success.

We also attempted to evaluate the benefit of using a state representation that takes into account a beam of top-n parses from the semantic parser. We expect that this is beneficial in cases where that the correct hypothesis is not the top ranked but present in this beam. To perform this ablation, we trained an agent that used the same parser and policy as in the *batchwise* condition, but only the top ranked parse from the parser to update its state, as opposed to a beam of parses when updating its state. These two systems differed in no other components, and were otherwise trained using similar interactions on Amazon Mechanical Turk. The results of this ablation are included in Table 3.4.

Table 3.4: Comparison of an agent using only the top hypothesis from the semantic parser and another using the top 10 parses. All differences are statistically significant according to an unpaired t-test with $p < 0.05$.

Number of parses considered	% successful dialogs	Dialog length
1	0.59	9.17
10	**0.64**	12.18

As expected, the agent using multiple parses performs the correct action a significantly higher fraction of times. The system using a single hypothesis has a shorter average length among its successful dialogs because it rarely succeeds in more complicated dialogs where the system needs repeated clarification or answers to multiple specific questions.

3.8 Summary

In this work, we have demonstrated that continuous dialog strategy learning and semantic parser learning can be successfully combined in a dialog system to enable an agent to better understand commands provided in natural language. Both the semantic parser and the dialog strategy can be automatically improved simultaneously using weak feedback provided during interaction with users rather than manually-labeled or artificially constructed training data. Ongoing parser learning could have confused the RL dialog learner by altering the underlying language understanding system while it was searching for an effective dialog policy. However, our results show that by using an appropriate RL algorithm and batchwise training regimen, this potential difficulty can be avoided, and both language understanding and dialog management can be improved simultaneously.

Chapter 4

Opportunistic Active Learning for Grounding Natural Language Descriptions

An important skill required by robots in a home or office setting is retrieving objects based on natural language descriptions. We find a number of objects such as books, mugs and bottles in such environments, that users typically refer to using a descriptive phrase invoking attributes of the object (eg: *"the blue mug"*), rather than having a unique name for each object. The set of such objects in these environments keeps changing, and sometimes even their properties might (eg: a water bottle becomes lighter as it gets emptied). This makes it near impossible to catalog the objects present, and their attributes, requiring robots to use perception to ground such descriptions of objects. Further, it is impossible to determine beforehand the attributes that people are likely to use such objects, and collect annotations for them. Thus to learn perceptual models for objects and attributes, a robot needs to be able to acquire labeled examples during interactions with users. In this work, we introduce the framework of opportunistic active learning, where a robot queries for labeled examples that are not immediately required, in anticipation of using them for future interactions.

4.1 Contributions

This work was originally presented in Thomason et al. (2017), and involved an equal contribution from Jesse Thomason and myself. It is also included in his thesis (Thomason, 2018). My contribution to this work was designing the framework of opportunistic active learning, and the task setup, as well as assisting with its implementation. Jesse Thomason was also involved in the implementation of the system, and he conducted the experiments presented. Jivko Sinapov and Justin Hart assisted in the setup of the infrastructure on the physical robot necessary for the experiment, and professors Peter Stone and Raymond Mooney helped shape the experimental design.

4.2 Motivation

In machine learning tasks where obtaining labeled examples is expensive, active learning is used to lower the cost of annotation without sacrificing model performance. Active learning allows a learner to iteratively query for labels of unlabeled data points that are expected to maximally improve the existing model. One of the challenges with developing learned systems on physical robots is that collecting labeled data with a physical robot is time consuming – making active learning an attractive option to reduce the amount of labeled data needed.

The most commonly used framework for active learning is pool-based active learning, where the learner has access to the entire pool of unlabeled data at once, and can iteratively query for examples. In contrast, sequential active learning is a framework in which unlabeled examples are presented to the learner in a stream (Lewis and Gale, 1994). For every example, the learner can decide whether to query for its label or not. This results in an additional challenge – since the learner cannot compare all unlabeled data points before choosing queries, each query must be chosen based on local information only. When a robot has to collect labels from users, the process is typically sequential. Even if the robot has an inventory of objects in its environment, it cannot predict where user interactions will occur, and hence, what objects are likely to be present in the context of the interaction.

Multilabel active learning is the application of active learning in scenarios where multiple labels, that are not necessarily mutually exclusive, are associated with a data point (Brinker, 2006). These setups often suffer from sparsity, both in the number of labels that are positive for a data point, and in the number of positive data points per label. Object descriptions require multilabel learning because objects can typically be described by a variety of properties including category, color, weight or typical use.

We propose the framework of Opportunistic Active Learning, which incorporates multilabel sequential active learning into an interactive task. Instead of having a separate training phase where labels are collected using active learning, and a testing phase where the learned models are used, we consider a setup where a robot is engaged in a sequence of interactions. In each of these, the robot must learn to trade off acquiring additional labels using active learning, with completing the task for which the user initiated the interaction. This is closer to the real-world scenario where the robot must integrate continuous learning into the regular tasks it is expected to perform.

In opportunistic active learning, an agent is engaged in a series of sequential decision-making tasks. The agent uses one or more supervised models to complete each task. Each task involves some sampled examples from a given feature space, and the agent is allowed to query for labels of these examples to improve its models for current and future tasks. Queries in this setting have a higher cost than in traditional active learning as the agent may choose to query for labels that are not relevant for the current task, but expected to be of use for future tasks. Such opportunistic queries enable an agent to learn from a greater number of interactions, by allowing it to ask queries that would aid future tasks when it is sufficiently confident of completing the current task. They also allow an agent to focus on concepts that could have more impact than those relevant to the current task – for example by choosing a frequently used concept as opposed to a rare one. Further, identifying which queries are optimal for model improvement is more difficult as the agent does not have access to the entire pool of unlabeled examples at any given time, similar to sequential active learning settings.

Another sample application of opportunistic active learning could be in a task oriented dialog system providing restaurant recommendations to a user. In this case, a possible opportunistic query would be to ask the user for a Chinese restaurant they liked, when the user is searching for an Italian one. The query is not relevant to the immediate task of recommending an Italian restaurant but would improve the underlying recommendation system.

4.3 Opportunistic Active Learning

Active learning identifies data points from a pool of unlabeled examples whose labels, if made available, are most likely to improve the predictions of a supervised model. Opportunistic Active Learning (OAL) is a setting that incorporates active learning queries into interactive tasks. Let $O = \{o_1, o_2, \ldots o_n\}$ be a set of examples, and $M = \{m_1, m_2, \ldots m_k\}$ be supervised models trained for different concepts, using these examples. For the problem of understanding natural-language object descriptions, O corresponds to the set of objects, M corresponds to the set of possible concepts that can be used to describe the objects, for example their categories (such as *ball* or *bottle*) or perceptual properties (such as *red* or *tall*).

In each interaction, an agent is presented with some subset $O_A \subseteq O$, and must make a decision based on some subset of the models $M_A \subseteq M$. Given a set of candidate objects O_A and a natural language description l, M_A would be the set of classifiers corresponding to perceptual predicates present in l. The decision made by the agent is a guess about which object is being described by l. The agent receives a score or reward based on this decision, and needs to maximize expected reward across a series of such interactions. Ideally, the reward function over interactions is set up to capture the relative importance of completing an interaction in a manner satisfactory to the user, while also minimizing any frustration they may face due to the system asking active learning queries. As with most dialog systems, a common proxy for this is to use a combination task success rate and dialog length. It is assumed that a user is more likely to be satisfied if the agent is successful at completing the task for which the interaction was initiated, and if the dialog is shorter. This is often captured by providing a large positive reward at the end of a successful interaction, a large negative reward at the end of a failed interaction, and a small negative reward for each additional dialog turn taken by the system. In this work, our goal is to evaluate whether off-topic questions improve a system's ability to perform object retrieval. As a result, we only compare dialog policies of fixed dialog length and only evaluate the agents on their average guess success rate.

During the interaction, the agent may also query for the label of any of the examples present in the interaction $o \in O_A$, for any model $m \in M$. Note that the agent cannot ask queries about objects $o \notin O_A$ as these are not assumed to be present in the context of the interaction. We call these queries opportunistic since the agent is limited to the objects present in the context of the current interaction, and hence cannot ask, or often even identify, a "globally optimal" query. The agent is said to be on-topic when it chooses to query for a label $m \in M_A$ and off-topic when it chooses to query for a label $m \notin M_A$. Labels from off-topic queries will not affect the decision made in the current interaction, and can only help with future interactions. For example, given a description "*the red box*", asking whether an object is *red*, could help the agent make a better guess, but asking whether an object is *round*, would be an off-topic query and can only help with future interactions. Queries have a cost, and hence the agent needs to trade-off the number of queries with the success at guessing across interactions.

The agent participates in a sequence of such interactions, and the models improve from labels acquired over multiple interactions. Thus the agent's expected reward per interaction is expected to improve as more interactions are completed. This setting differs from the traditional application of active learning in the following key ways:

- The agent cannot query for the label of any example from the unlabeled pool. It is restricted to the set of objects available in the current interaction, O_A.
- The agent is evaluated on the reward per interaction, rather than the final accuracy of the models in M.
- The agent may make opportunistic queries (for models $m \notin M_A$) that are not relevant to the current task.

Due to these differences, this setting provides challenges not seen in most active learning scenarios:

- Since the agent never sees the entire pool of unlabeled examples, it can neither choose queries that are globally optimal, nor use variance reduction strategies that still use near-optimal queries (such as sampling from a beam of near globally optimal queries).
- Since the agent is evaluated on task completion, it must learn to trade-off finishing the task with querying to improve the models.
- The agent needs to estimate the usefulness of a model across multiple interactions, to identify good opportunistic queries.

4.4 Object Retrieval Task

To test the effectiveness of opportunistic active learning, we created an object identification task using a real robot. Figure 4.1 shows the physical setup of our task.

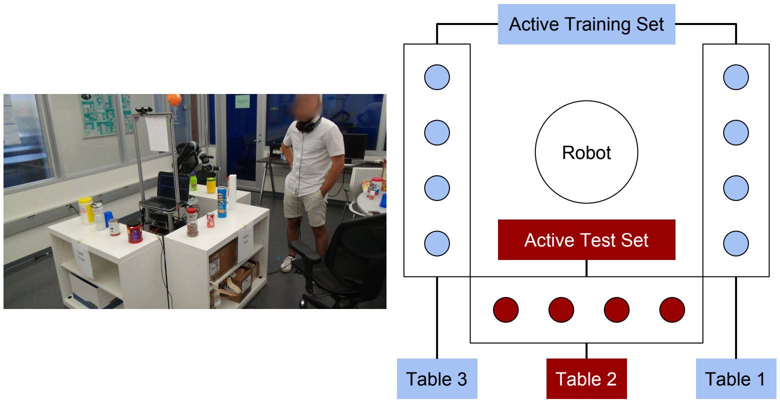
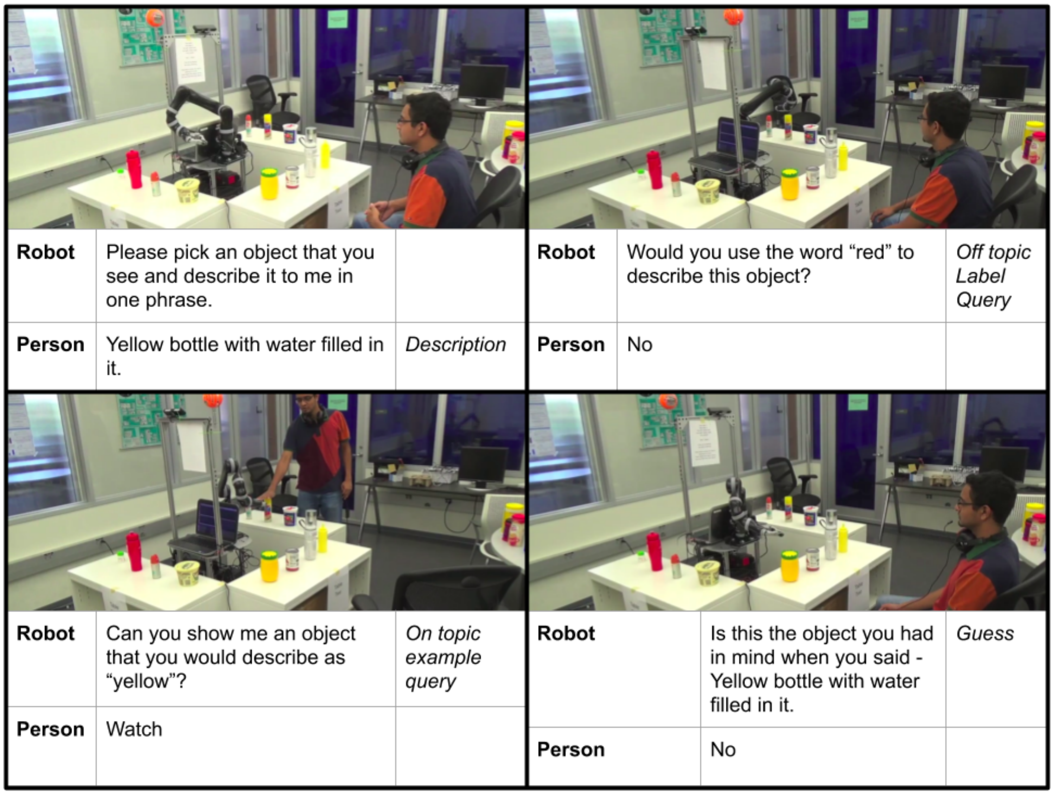


Figure 4.1: Experiment Setup – Participants described an object on Table 2 from the *active test set* to the robot in natural language, then answered the robot's questions about the objects in its *active training set* on the side Tables 1 and 3 before the robot guessed the described target object.

We split the set of objects in the current interaction O_A into an active training set O_A^{tr}, and an active test set O_A^{te}. The target object being described is in the active test set and the robot can query objects present in the active training set. This ensures that the robot needs to learn generalizable perceptual classifiers. It also simulates the situation where the target object is in a different room, and the robot needs to query about local objects (active training set) to learn classifiers that can be used later to identify the target.

The human participant and robot both started facing Table 2, which held the active test set. The tables on the sides (Tables 1 and 3) contained objects in the active training set.



Robot	Please pick an object that you see and describe it to me in one phrase.	
Person	Yellow bottle with water filled in it.	*Description*

Robot	Would you use the word "red" to describe this object?	*Off topic Label Query*
Person	No	

Robot	Can you show me an object that you would describe as "yellow"?	*On topic example query*
Person	Watch	

Robot	Is this the object you had in mind when you said - Yellow bottle with water filled in it.	*Guess*
Person	No	

Figure 4.2: A sample interaction in our experiment.

A sample interaction is shown in Figure 4.2. The experiment starts with the robot asking the participant to choose an object from the active test set and describe it using a noun phrase, and the robot needs to identify the object described. Before guessing, it is allowed to ask queries of the following two types:

- Label queries - A yes/no question about whether a predicate can be used to describe one of the objects in the active training set, e.g. "*Would you use the word "red" to describe this object?*". The robot asks these queries pointing to the object being referenced.

- Example queries - Asking for an object, in the available training set, that can be described by a particular predicate, e.g. "*Can you show me an object that you would describe as "yellow"?*". This is used for acquiring positive examples since most predicates tend to be sparse.

The referenced objects in label queries, and the objects used to answer example queries must belong to the active training set, that is they must be on the side tables 1 and 3 (Figure 4.1). In Figure 4.2, the label query is off-topic as it references the concept "*red*" that is not present in the current description, "*Yellow bottle with water filled in it*". The example query is on-topic as it uses the concept "*yellow*" which is part of the description.

Participants could physically handle the objects before the start of the interaction and were encouraged to use attributes instead of categories to describe objects. They were also told that the robot had previously looked at and interacted with the objects physically using its arm. A video of a complete sample interaction can be seen at `https://youtu.be/f-CnIF92_wo`.

Natural Language Grounding.

We assume that the description provided is a conjunction of one-word predicates. Given a description, the agent tokenizes it and removes stopwords. Each remaining word is treated as a perceptual predicate. For each perceptual predicate, the robot trained an SVM classifier based on multimodal features. These were originally collected for a different study, Sinapov et al. (2016), which contains further details of the features extracted. We did not restrict the choice of words that participants were allowed to use to describe objects, so our system learned from an open vocabulary. For every predicate $p \in P$ for P the set of predicates known to the agent and object $o \in O_A$, a decision $d(p, o) \in \{-1, 1\}$ and a confidence [1] $\kappa(p, o)$ in that decision are calculated.

Active Learning Dialog Policy.

After the participant described a chosen target object in natural language, the robot asked a fixed number of questions about objects in its active training set before guessing a target object. We chose each type of query – label or example – with a fixed probability.

In a label query, to select the predicate p and object $o \in O_A^{tr}$ to ask about, we first find the objects in O_A^{tr} with the lowest confidence κ per predicate (ties broken randomly).

$$o_{\text{min}}(p) = \text{argmin}_{o \in O_A^{tr}}(\kappa(p, o)).$$

and sample predicates inversely proportional to their confidence in their least confident labels.

$$prob(p) = \frac{1 - \kappa(p, o_{\text{min}}(p))}{\sum_{q \in P \setminus \{p\}} 1 - \kappa(q, o_{\text{min}}(q))}. \tag{4.1}$$

For example queries, a predicate p was selected uniformly at random from those with insufficient data to fit a classifier.

The robot updated relevant perceptual classifiers with each answer, and after

---

[1] Cohen's kappa estimated from cross-validation performance on available examples.

all questions, identified the best guess $o^* \in O_A^{te}$, using classifiers of predicates $P_A \subseteq P$ present in the description as follows,

$$o^* = \text{argmax}_{o \in O_A^{te}} \left(\sum_{p \in P_A} d(p,o)\kappa(p,o) \right). \tag{4.2}$$

If the robot guessed incorrectly, the human pointed out the correct object. The target object was then considered a positive example for predicates P_A.

4.5 Experimental Setup

Figure 4.3 shows the set of objects used in our experiments. These had originally been chosen for a previous study, Sinapov et al. (2016), during which the objects were explored to obtain the visual, audio and haptic features. We divided the dataset into 4 folds, {0, 1, 2, 3}, of 8 objects each.

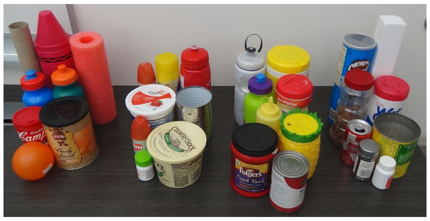


Figure 4.3: The objects used in our experiments, from fold 0 on the far left to fold 3 on the far right.

We compared two possible static dialog policies that the robot could use -

- *baseline* - This agent remained on-topic and was only allowed to ask questions mentioned in the description provided in the current interaction.

- *inquisitive* - This agent accumulated words across descriptions, and could ask queries using any word previously used in an interaction with it.

For example, if a person described a target object as “a pink cylinder”, the *baseline* agent could only ask queries about the predicates “pink” and “cylinder”, but the *inquisitive* query about other predicates such as “heavy”. Both agents asked a fixed number of questions per interaction before guessing, but the *inquisitive* agent was set to ask two extra questions per interaction, thus making it both more talkative and less task-oriented. At test time, both agents asked the same number of questions so that any differences would be due to differences in knowledge acquired from previous rounds.

We conducted three rounds of interaction, updating the classifiers of both agents between rounds. In round 1, the active train set consisted of fold 1 and the active test set consisted of fold 0. In round 2, the active train set consisted of fold 2 and the active test set consisted of fold 1. In round 3, the active train set consisted of fold 3 and the active test set consisted of fold 2. Round 3 was the test round in which both agents were set to ask the same number of questions. However, in all rounds, the active test set consisted of objects that had not been used to train the classifiers.

We hypothesized that:

1. The *inquisitive* agent would have a higher success rate at guessing objects.
2. Users would not qualitatively dislike the *inquisitive* agent for asking too many questions and being off-topic.

Table 4.1: Fraction of successful guesses made in each condition per round.

Round	Baseline	Inquisitive
1	0.175	0.35
2	0.225	0.325
3	0.175	0.325

4.6 Results and Discussion

In each round, we had five participants conduct two interactions in each dialog setting. Each participant filled an exit survey for every interaction they completed.

We present the success rate in the two conditions in Table 4.1. The *inquisitive* agent is consistently more successful than the *baseline* agent at identifying the correct object. Its opportunistic strategy allows it to leverage the overlap of predicates across rounds.

The *inquisitive* agent was also perceived as asking too many questions only in round 2, although it asked more questions than the *baseline* even in round 1, and asked off-topic questions even in round 3. However, participants also rated it as being more fun, and more likely to be usable. This is likely due to its higher success rate at guessing objects correctly.

4.7 Summary

In this chapter, we propose the framework of opportunistic active learning to integrate active learning questions at test time to enable a robot to perform lifelong learning. Our framework allows a robot to identify new perceptual concepts and build classifiers for them only using labels obtained during test time interactions. We demonstrate that a robot performing opportunistic active learning is more successful than a baseline agent at using knowledge obtained from test-time queries at improving its ability to retrieve objects based on natural language descriptions, and that people find such a robot more fun and usable.

Chapter 5

Learning a Policy for Opportunistic Active Learning

In chapter 4, we demonstrated that an interactive robot following an opportunistic active learning policy is better able to ground natural language descriptions of objects across interactions. However, in that work, we compared two static dialog policies that the robot could use for the task. In this work, we learn a dialog policy from interactions using reinforcement learning, that effectively trades off task completion with model improvement that would benefit future tasks.

5.1 Contributions

This work was originally presented in Padmakumar et al. (2018). My contribution to this work included designing and implementing the all components of the dialog system and the user simulator, as well as designing and conducting the experiments. Professors Raymond Mooney and Peter Stone provided feedback that shaped the choice of algorithms and experimental design.

5.2 Motivation

In chapter 4, we demonstrated that a robot engaged in an interactive object retrieval task learns to perform better at identifying objects correctly over time if it follows an opportunistic dialog policy that queries for labels not necessarily relevant to the current interaction. However, in that work, we compared two of many possible static dialog policies. The design of those static policies ensured that all dialogs have the same length. However, in practice, it may be more useful to ask for queries in some interactions than others. For example, the objects in context of some interactions may overall be more informative than in others. We may also want the robot to learn to decide whether the likelihood of correctly completing the task the user wants should influence the number of active queries being included in the interaction. To enable the robot to adapt to such changes, we would like the robot to be able to learn a dialog policy using reinforcement learning to trade off asking active queries and completing the tasks for which the users initiated interactions.

Also, in chapter 4, we used uncertainty sampling (Lewis and Gale, 1994) to choose which label query to ask for a given predicate. However, there are many possible metrics that could be used for choosing active learning queries, including density-weighted methods (Settles and Craven, 2008), expected error reduction (Roy and McCallum, 2001), query by committee (Seung et al., 1992), and the presence of conflicting evidence (Sharma and Bilgic, 2016). A survey can be found in Settles (2010). However, most of these are assumed for a setting where a single binary classifier is being learned. The most common way to extend this to the multilabel setting is to assume that one-vs-all classifiers are learned for each label, and that all the learned classifiers are comparable (Brinker, 2006; Singh et al., 2009; Li et al., 2004). This was the procedure we used in chapter 4. Since it may be difficult to decide ahead of time which metric is most likely to be useful, it would be desirable to evaluate many such metrics and learn to choose between them. We also may want the robot to use other information in this decision, such as how frequently a predicate is used to describe objects. Thus we would also like our dialog policy to learn to choose between different possible queries.

Recently, there has been interest in using reinforcement learning to learn a policy for active learning. Fang et al. (2017) use deep Q-learning to acquire a policy that sequentially examines unlabeled examples and decides whether or not to query for their labels; using it to improve named entity recognition in low resource languages. Also, Bachman et al. (2017) use meta-learning to jointly learn a data selection heuristic, data representation and prediction function for a distribution of related tasks. They apply this to one shot recognition of characters from different languages, and in recommender systems. In contrast to these works, we learn a policy for a task that contains *both* possible actions that are active learning queries, *and* actions that complete the current task, thus resulting in a greater exploration-exploitation trade-off.

More similar to our setup is that of Woodward and Finn (2017) which uses reinforcement learning with a recurrent-neural-network-based Q-function in a sequential one-shot learning task to decide between predicting a label and acquiring the true label at a cost. This setup also has a higher cost than standard active learning where the test set is separated out. This is a continuous task without clearly separated interactions or episodes. In our setting, each episode or interaction allows for querying and requires completion of an interaction, which further increases the trade-off between model improvement and exploitation. Further, we consider a multilabel setting, which increases the number of actions at each decision step.

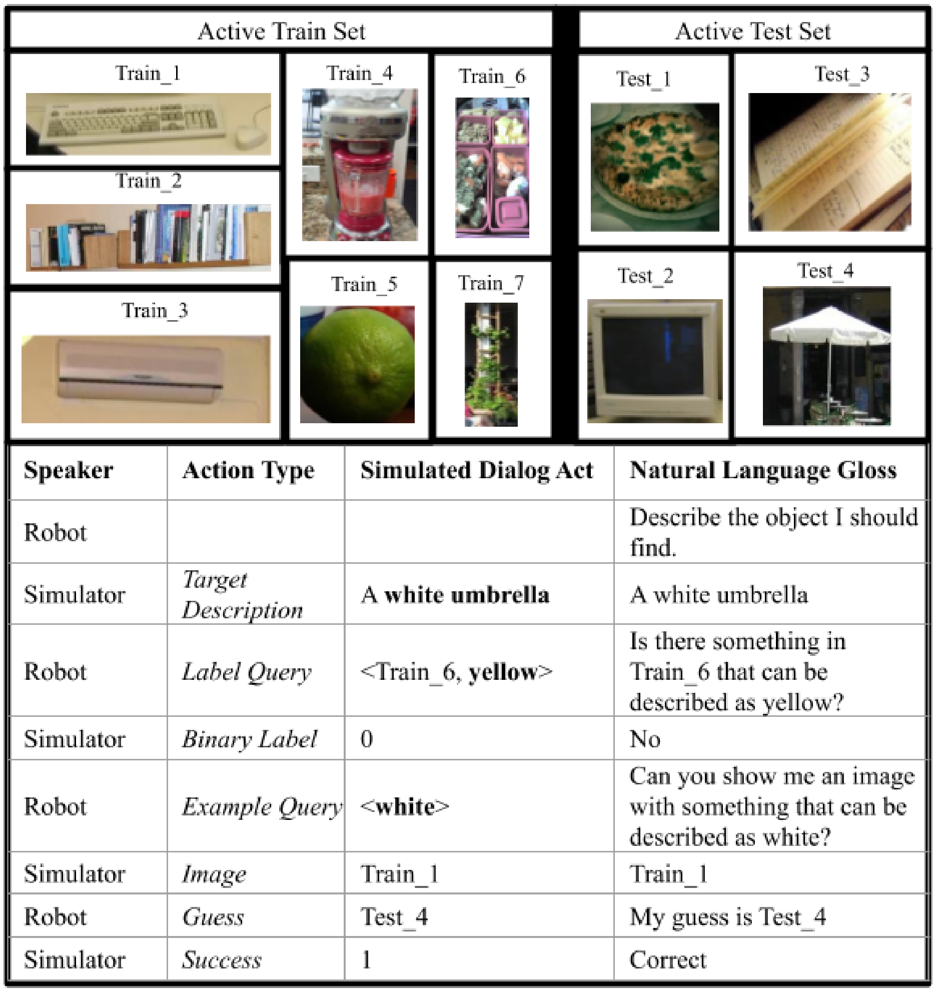
5.3 Task Setup

We consider the same general task as in chapter 4 (section 4.4). However, we set it up in simulation using the Visual Genome dataset (Krishna et al., 2017) as we need a large number of dialogs to learn a dialog policy. The Visual Genome dataset contains images with regions (crops) annotated with natural-language descriptions. Bounding boxes of objects present in the image are also annotated, along with attributes of objects. Region descriptions, objects and attributes are annotated using unrestricted natural language, which leads to a diverse set of predicates. Using the annotations, we can associate a list of objects and attributes relevant to each image region, and use these to answer queries from the agent.

A sample interaction is seen in figure 5.1. For each interaction, we uniformly sample 4 regions to form the active test set, and 8 regions to form the active training set. [1] One region is then uniformly sampled from the active test set to be the target object. Its description, from annotations, is provided to the agent to be grounded, that is, the agent must identify the image in the active test set that best fits the description. In Figure 5.1, this is the description "*A white umbrella*". Following this, the agent can ask label and example queries on the active training set, before guessing which object was being described. In Figure 5.1, the agent first asks whether

---

[1] The regions in the dataset are divided into separate pools from which the active training and active test sets are sampled (described as classifier-training and classifier-test sets in section 5.5.1), to ensure that the agent needs to learn classifiers that generalize across objects.



Speaker	**Action Type**	**Simulated Dialog Act**	**Natural Language Gloss**
Robot			Describe the object I should find.
Simulator	*Target Description*	A **white umbrella**	A white umbrella
Robot	*Label Query*	<Train_6, **yellow**>	Is there something in Train_6 that can be described as yellow?
Simulator	*Binary Label*	0	No
Robot	*Example Query*	<**white**>	Can you show me an image with something that can be described as white?
Simulator	*Image*	Train_1	Train_1
Robot	*Guess*	Test_4	My guess is Test_4
Simulator	*Success*	1	Correct

Figure 5.1: A sample interaction – Perceptual predicates are marked in bold.

the label "*yellow*" applies to a specific image. This query is also off-topic as the word "*yellow*" is not present in the current description - "*A white umbrella*". The annotated objects and attributes associated with active training regions are used to answer queries. A predicate is labeled as being applicable to a region if it is present in the list of objects and attributes associated with the region. In Figure 5.1, the simulator provides a binary label 0 as the label "yellow" is not present in the list of annotated objects and attributes associated with the queried image. The second query in Figure 5.1 is an example query where the agent asks for an example for the predicate “*white*”. This is an on-topic query as this is a word present in the current description. The simulator answers the query by finding an image which has “*white*” included in its list of annotated attributes. If there was no such image present in the active train set, it would be able to answer `None of these`. Finally the agent makes a guess and obtains a binary indicator of success from the simulator based on whether the guessed image is the target.

5.4 Methodology

We assume that the description provided is a conjunction of one-word predicates. Given a description, the agent tokenizes it and removes stopwords. Each remaining word is stemmed and treated as a perceptual predicate. This method allows the agent to learn an open vocabulary of predicates, but unable to handle multi-word predicates or non-compositional phrases. The agent learns a separate binary SVM for each predicate, trained over deep features extracted from images. These are obtained from the penultimate layer of the VGG network (Simonyan and Zisserman, 2014) pretrained on ImageNet (Russakovsky et al., 2015). The agent has no initial classifiers for any predicate, and learns these classifiers purely from labels acquired during interactions.

5.4.1 Grounding Descriptions

Grounding is performed similar to previous work (section 4.4). Given predicates $P_A \subseteq P$ present in the target description l, a decision $d(p, o) \in \{-1, 1\}$ from the classifier for predicate p for object o, and the confidence of the classifier $C(p)$ (estimated F1 from cross-validation on acquired labels), the best guess o^* is computed as,

$$o^* = \text{argmax}_{o \in O_A^{te}} \sum_{p \in P_A} d(p, o) * C(p)$$

5.4.2 MDP Formulation

We model interactions as episodes in a Markov Decision Process (MDP) (section 2.7.1). At any point, the agent is in a state consisting of the VGG features of the regions in the current interaction, the predicates in the current description, and the agent's classifiers. The agent can choose from the set of actions which includes an action for guessing, and an action for each possible query the agent can currently make, including both label and example queries. The guess action always terminates the episode, and query actions transition the agent to a new state as one of the classifiers gets updated. The agent gets a reward for each action taken. Query actions have a small negative reward, and guessing results is a large positive reward when the guess is correct, and a large negative reward when the guess is incorrect. In our experiments, we treat the reward values as hyperparameters that can be tuned. We fixed an arbitrary dialog length representing what we expected would be the maximum number of questions a user would be willing to answer, and selected hyperparameters that results in the maximum dialog success rate within this dialog length on validation data. The best results were obtained with a reward of 200 for a correct guess, -100 for an incorrect guess and -1 for each query. Our final reward function does not answer the question of how to optimally balance the two metrics of dialog success rate and dialog length. We believe that this balance is likely to be domain dependent and would need to be chosen by a user study that measures how many questions people are typically willing to tolerate and to what extent this varies with dialog success rate. It is difficult to conduct such a study on a crowdsourced platform since crowdworkers are not sufficiently invested in the success of the dialog interactions.

5.4.3 Identifying Candidate Queries

In any interaction, the agent can make label or example queries. In a label query, the agent can ask for the label of any object for a specific predicate. If O_A is the set of objects present in the active training set of the current interaction, and P is the set of predicates that have been seen by the agent in all interactions so far, then the set of possible label queries is $P \times O_A$. Once the agent chooses a predicate p and object o to be queried, it obtains the corresponding label and can update its classifier for p. In an example query, the agent asks for a positive example for any predicate $p \in P$. The agent will either receive a positive label for p for some object $o \in O_A$ or learn that the label is negative $\forall\ o \in O_A$, and can appropriately update the classifier for p.

Ideally, we would like the agent to learn a policy over all possible queries.

However, since $|P|$ grows across interactions as the agent encounters more predicates in descriptions, the number of candidate actions in a state increases over time, so searching the entire space of possible queries can become intractable. Hence we sample a few promising queries and learn a policy to choose between them. In our previous work, predicates were sampled according to a distribution that weighted them inversely proportional to the confidence in their current classifiers. However, if the space of possible predicates is large, then this results in no classifier obtaining a reasonable number of training examples. In this scenario, it is desirable to focus on a small number of predicates, possibly stopping the improvement on a predicate once the classifier for it has been sufficiently improved. We sample queries from a distribution designed to capture this intuition. The probability assigned to a predicate by this distribution increases linearly, for confidence below a threshold, and decreases linearly thereafter.

Let $w(p_i)$ be the weight for predicate p_i with estimated F1 $k(p_i)$. Weights start at w_{min} for $k = 0.0$ and increase linearly to w_{max} at some $k_{max} \in (0, 1)$. For $k > k_{max}$, weights again linearly decrease to w_{min} for $k = 1.0$. That is, for $k <= k_{max}$,

$$w(p_i) = \frac{k(p_i)}{k_{max}}(w_{max} - w_{min})$$

For $k > k_{max}$,

$$w(p_i) = \frac{1.0 - k(p_i)}{1.0 - k_{max}}(w_{max} - w_{min})$$

The weights are then normalized to obtain a probability distribution. A beam of label queries can then be sampled from this.

5.4.4 Baseline Static Policy

As a baseline, we use a static policy similar to that used in chapter 4. At each state, a single label query and example query are sampled. The agent asks a fixed number of queries before guessing. In chapter 4, we use thresholds that prevent queries from being asked when there are no predicates whose classifiers have sufficiently low estimated accuracy. Since we used a dataset with a much larger number of predicates, these thresholds were always crossed if the agent had even one candidate query.

5.4.5 Policy Learning

We experimented with multiple policy learning algorithms to learn a dialog policy for the MDP, including Q-learning (section 2.7.4), KTD Q-learning (section 2.7.6), REINFORCE (section 2.7.3), Actor-Critic and A3C (section 2.7.5). In general policy gradient based methods performed better than the Q-learning based methods, but there was no statistically significant difference between individual policy gradient algorithms. We report results using REINFORCE as that had the highest mean dialog success rate.

The agent learns a policy $\pi(a|s;\theta)$, parameterized with weights θ that computes the probability of taking action a in state s. Given a feature representation $f(s,a)$ for a state-action pair (s,a), the policy is of the form:

$$\pi(a|s;\theta) = \frac{e^{\theta^T f(s,a)}}{\sum_{a'} e^{\theta^T f(s,a')}}$$

where the denominator is a sum over all actions possible in state s. The weights are updated using a stochastic gradient ascent rule:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

where $J(\theta)$ is the expected return from the policy according to the distribution over trajectories induced by the policy.

The state consists of the predicates in the current description, the candidate objects, and the current classifiers. Since both the number of candidate objects and classifiers varies, and the latter is quite large, it is necessary to identify useful features for the task to obtain a vector representation needed by most learning algorithms. In our problem setting, the number of candidate actions available to the agent in a given state is variable. Hence we need to create features for state-action pairs, rather than just states.

5.4.6 Features for Policy Learning

The object retrieval task consists of two parts – identifying useful queries to improve classifiers, and correctly guessing the image being referred to by a given description. The current dialog length is also provided to influence the trade-off between guessing and querying.

Guess-success features

Let $P_A = \{p_1, p_2, \ldots p_k\}$ be the predicates extracted from the current description. For each predicate $p \in P_A$, we have the estimated F1 of the classifier

$C(p)$, and for each object o in the active test set, we have a decision $d(p, o) \in \{-1, 1\}$ from the classifier. We refer to $s(p, o) = d(p, o) * C(p)$ as the score of the classifier of p for object o. The following features are used to predict whether the current best guess is likely to be correct:

- Lowest, highest, second highest, and average estimated F1 among classifiers of predicates in P_A – learned thresholds on these values can be useful to decide whether to trust the guess.
- Highest score among regions in the active test set, and the differences between this and the second highest, and average scores respectively – a good guess is expected to have a high score to indicate relevance to the description, and substantial differences would indicate that the guess is discriminative. Similar features are also formed using the unweighted sum of decisions.
- An indicator of whether the two most confident classifiers agree on the decision of the top scoring region, which increases the likelihood of its being correct.

We compared directly using these features to training a regressor that uses them to predict the probability of a successful guess, and then using this as a higher-level policy feature. We found no difference between the two methods and the results reported directly use these features in the vector provided to the policy learner.

Query-evaluation features

The following features are expected to be useful in predicting whether it is useful to query for the label of a particular predicate:

- Indicator of whether the predicate is new or already has a classifier – this allows the policy to decide between strengthening existing classifiers or creating classifiers for novel predicates.
- Current estimated F1 of the classifier for the predicate – as there is more to be gained from improving a poor classifier.
- Fraction of previous dialogs in which the predicate has been used, and the agent's success rate in these – as there is more to be gained from improving a frequently used predicate but less if the agent already makes enough correct guesses for it.
- Is the query off-topic – as these will not help the current guess.

Label queries also have an image region specified, and for these we have additional features that use the VGG feature space in which the region is represented for classification:

- Margin of the image region from the hyperplane of the classifier of the predicate – motivated by uncertainty sampling.
- Average cosine distance of the image region to others in the dataset – motivated by density weighting to avoid outliers.
- Fraction of the k-nearest neighbors of the region that are unlabeled for this

predicate – motivated by density weighting to identify a data point that can influence many labels.

5.5 Experiments

5.5.1 Dataset

The Visual Genome dataset contains a total of 108,077 images with 5,406,592 annotated regions. Since objects and attributes are annotated with free-form text rather than from a fixed, pre-defined vocabulary, there is considerable diversity in the language used for annotation. There are 80,908 unique objects annotated and 44,235 attributes. We assume that any objects that partially overlap with a region are present in it, as these are usually used in descriptions.

Using the annotations, we can associate a list of objects and attributes relevant to each image region. We lower-case all annotations, remove special characters and perform stemming to help normalize terms. Note that the labels obtained from this list may be noisy, since not all objects and attributes may be annotated, and we assume that any word not present in this list is not applicable to the target image. However, we expect that a certain amount of noise is also likely to be present in live human interactions.

Sampling Dialogs

We want the agent to learn a policy that generalizes to novel predicates available at test time. In order to be able to evaluate this, we divide the set of possible regions in the Visual Genome dataset into policy training and policy test regions as follows. We select all objects and attributes present in at least 1,000 regions. Half of these were randomly assigned to the policy test set. All regions that contain one of these objects or attributes are assigned to the policy test set, and the rest to the policy training set. Thus regions seen at test time may contain predicates seen during training, but will definitely contain at least one novel predicate. Further, the policy training and policy test sets are respectively partitioned into a classifier training and classifier test set using a uniform 60-40 split. The final policy training split has 1,417,967 regions and the final policy test split has 490,655 regions.

During policy training, for each dialog, 8 regions are sampled from the classifier-training subset of the policy-training regions to form the active training set, and 4 regions are sampled from the classifier-test subset of the policy-training set to form the active test set. During policy testing, the active training set of each dialog is sampled from the classifier training subset of the policy test regions, and the active test set of the dialog is sampled from the classifier test subset of the policy test set.

5.5.2 Experiment phases

Our baseline is a static policy similar to that used in previous work (4.4).

For efficiency, we run dialogs in batches, and perform classifier and policy updates at the end of each batch. We use batches of 100 dialogs each. Our experiment runs in 3 phases:

- Initialization – Since learning starting with a random policy can be difficult,

Table 5.1: Results on dialogs sampled from the policy test set after 10 batches of classifier training. *–Guess* and *–Query* are conditions with the guess and query features, respectively, ablated. Boldface indicates that the difference in that metric with respect to the *Static* policy is statistically significant according to an unpaired Welch t-test with $p < 0.05$.

Policy	Success rate	Average Dialog Length
Learned	**0.44**	**12.95**
–Guess	0.37	**6.12**
–Query	0.35	**6.16**
Static	0.29	16.00

we first run batches of dialogs on the policy training set using the static policy, and update the RL policy from these episodes. This "supervised" learning phase is used to initialize the RL policy.

- Training – We run batches of dialogs on the policy training set using the RL policy, starting it without any classifiers. In this phase, the policy is updated using its own experience.
- Testing – We fix the parameters of the RL policy, and run batches of dialogs on the policy test set. During this phase, the agent is again reset to start with no classifiers. We do this to ensure that performance improvements seen at test time are purely from learning a strategy for opportunistic active learning, not from acquiring useful classifiers in the process of learning the policy.

5.6 Experimental Results and Analysis

We initialize the policy with 10 batches of dialogs, and then train on another 10 batches of dialogs, both sampled from the policy training set. Following this, the policy weights are fixed, the agent is reset to start with no classifiers, and we test on 10 batches of dialogs from the policy test set. Table 5.1 compares the average success rate (fraction of successful dialogs in which the correct object is identified), and average dialog length (average number of system turns) of the best learned policy, and the baseline static policy on the final batch of testing. We also compare the effect of ablating the two main groups of features. The learned agent guesses correctly in a significantly higher fraction of dialogs compared to the static agent, using a significantly lower number of questions per dialog.

When either the group of guess or query features is ablated, the success rate clearly decreases. While the mean success rate still remains above the baseline, the difference is no longer statistically significant. Further, at the end of the initialization phase, the average dialog length in all three conditions is about the same. In the two ablated conditions, the dialog length does not increase to become close to that of the static policy, which suggests that the agent does not learn that asking more queries improves dialog success. This is expected because the agent is either not able to evaluate the usefulness of queries, or the likelihood of success of a guess. However, in the learned policy with all features, the agent is able to identify a benefit in asking queries, and utilizes them to improve its success rate.

It is important to note that it is non-trivial to decide how to trade-off dialog success with dialog length. This should be decided for any given application by comparing the cost of an error with that of the user time involved in answering queries, and the reward function should be set appropriately based on this. Ideally, we would like to see an increase in dialog success rate *and* a decrease in dialog length, as is the case when comparing the learned and static policies. However, depending on the application, it may also be beneficial to see a smaller increase in success rate with a larger decrease in dialog length, as is the case in the ablated conditions.

We also explored ablating individual features. The results are presented in Table 5.2. We use the notation p_{best} to represent the classifier relevant to the current interaction with maximum F1, that is, the classifier for $p_{best} = argmax_{p \in P_A} C(p)$. The second best classifier is $p_{sec} = argmax_{p \in P_A - p_{best}} C(p)$.

Table 5.2: Results of individual feature ablation. Boldface indicates that the difference in success rate with respect to *Static* is statistically significant according to an unpaired Welch t-test with $p < 0.05$.

Feature Ablated	Success rate	Average Dialog Length
None	**0.44**	12.95
Number of system turns used - normalized	0.41	3.8
Density of object in label query	0.4	12.89
Fraction of previous dialogs using predicate in query that have succeeded	0.39	5.46
Score (normalized) of top region	0.39	6.3
Fraction of k nearest neighbours of the object in label query, which are unlabeled	0.39	10.41
Indicator for guess action	0.38	7.21
Minimum value of $C(p)$ for $p \in P_A$	0.37	6.37

Continued on next page

Table 5.2: Results of individual feature ablation. Boldface indicates that the difference in success rate with respect to *Static* is statistically significant according to an unpaired Welch t-test with $p < 0.05$.

Feature Ablated	Success rate	Average Dialog Length
Decision of p_{sec} for object with highest score	0.37	11.21
Difference between decision of p_{best} for object with highest score, and the average of its decisions for objects in the active test set	0.36	2.78
Indicator of whether the question is on-topic	0.36	5.25
Is decision of p_{best} same for objects with top two scores	0.36	5.32
Indicate of whether the predicate in the query has a classifier	0.36	13.97
Frequency of use of the predicate in query - normalized	0.35	4.48
Indicator for the action of asking a positive example	0.35	5.36
Second highest value of $C(p)$ for $p \in P_A$	0.35	5.9
Current estimated F1 for classifier of the predicate in query	0.35	6.53
Decision of p_{best} for object with highest score	0.34	3.85
Indicator for the action of asking a label	0.34	7.05

Continued on next page

Table 5.2: Results of individual feature ablation. Boldface indicates that the difference in success rate with respect to *Static* is statistically significant according to an unpaired Welch t-test with $p < 0.05$.

Feature Ablated	Success rate	Average Dialog Length
Average value of $C(p)$ for $p \in P_A$	0.34	7.63
Margin of object in label query	0.34	8.08
Maximum value of $C(p)$ for $p \in P_A$	0.33	6.31
Difference between decision of p_{sec} for object with highest score, and the average of its decisions for objects in the active test set	0.33	8.84
Difference between top two scores in the active test set	0.32	8.05
Is decision of p_{best} same for objects with top two scores	0.32	10.01
Difference between top score and average score in the active test set	0.31	7.18
Baseline	0.29	16

We found that the effect of ablating most single features is similar to that of ablating a group of features. The mean success rate decreases compared to the full policy with all features. It remains better than that of the static policy, but in most cases the difference stops being statistically significant. Among features for evaluating the guess, the removal of the difference between the two highest scores in the active test set has a fairly large effect, compared with the value of the highest score. This is expected because for retrieval it is sufficient if an object is simply scored higher than the other candidates. Further, since classifiers improve over time, the score threshold that indicates a good guess changes, and hence would be difficult to learn. Among query evaluation features, we find, unsurprisingly, that removal of the feature providing the margin of the object in a label query affects performance much more than removal of features such as density and fraction of labeled neighbors, which merely indicate whether the object is an outlier.

Qualitatively, we found that the dialog success rate was higher for both short, and very long dialogs, with a decrease for dialogs of intermediate length. We also found that most active learning queries were off-topic. This suggests that longer dialogs are used to accumulate labels via opportunistic off-topic questions, as opposed to on-topic questions. The learned policy still suffers from high variance in dialog length suggesting that trading off task completion against model improvement is a difficult decision to learn. We find that the labels collected by the learned policy are more equitably distributed across predicates than the static policy, resulting in a tendency to have fewer classifiers of low estimated F1. There is relatively little difference in the number of predicates for which classifiers are learned. This suggests that the policy learns to focus on a few predicates, as the baseline does, but learn all of these equally well, in contrast to the baseline which has much higher variance in the number of labels collected per predicate.

5.7 Summary

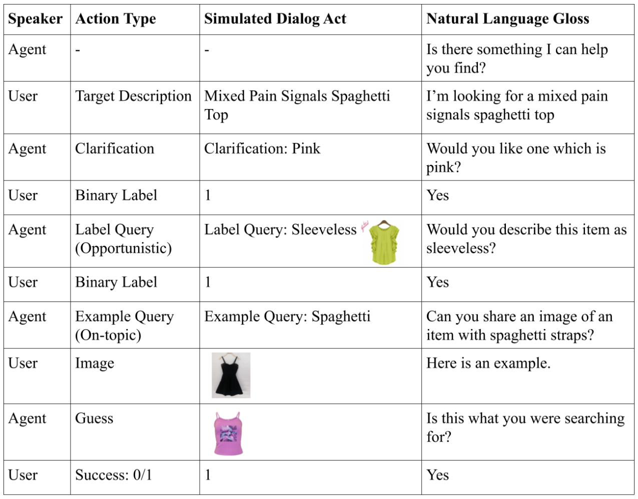
In this chapter, we showed how to formulate an opportunistic active learning problem as a reinforcement learning problem, and learn a policy that can effectively trade-off opportunistic active learning queries against task completion. We evaluated this approach on the task of grounded object retrieval from natural language descriptions and learn a policy that retrieves the correct object in a larger fraction of dialogs than a previously proposed static baseline, while also lowering average dialog length.

Chapter 6

Dialog Policy Learning for Joint Clarification and Active Learning Queries

Robots and other intelligent systems need to be able to recover from mistakes, resolve uncertainty, and adapt to novel concepts not seen during training. In chapter 5 we demonstrate how a dialog agent can learn to incorporate queries into existing user interactions, that allow it to obtain labeled examples to improve its models and adapt to the domain of operation. In this chapter we demonstrate how this can be combined with prior work on using clarifications questions in dialog for correction and resolving uncertainty (section 2.2). We train a hierarchical dialog policy to jointly perform *both* clarification and active learning in the context of an interactive language-based image retrieval task, and demonstrate that jointly learning dialog policies for clarification and active learning is more effective than the use of static dialog policies for one or both of these functions.

In this chapter, we use a different application, motivated by online shopping, to demonstrate the use of our dialog policy. A sample interaction is shown in Figure 6.1. We consider an application where a dialog system is combined with a retrieval system to help a customer find an article of clothing. Instead of showing a large number of results obtained from the retrieval system, the dialog system attempts to use clarifications to refine the search query, and active learning questions to obtain labelled examples for concepts it has not been trained on. We chose to work with this application so that we could use a dataset where all images were annotated both with natural language descriptions, as well as binary labels for a consistent set of attributes (Guo et al., 2019). This allows us to simulate dialogs in a manner similar to chapter 5 but with less noise in the responses provided by the dialog system.



Speaker	Action Type	Simulated Dialog Act	Natural Language Gloss
Agent	-	-	Is there something I can help you find?
User	Target Description	Mixed Pain Signals Spaghetti Top	I'm looking for a mixed pain signals spaghetti top
Agent	Clarification	Clarification: Pink	Would you like one which is pink?
User	Binary Label	1	Yes
Agent	Label Query (Opportunistic)	Label Query: Sleeveless	Would you describe this item as sleeveless?
User	Binary Label	1	Yes
Agent	Example Query (On-topic)	Example Query: Spaghetti	Can you share an image of an item with spaghetti straps?
User	Image		Here is an example.
Agent	Guess		Is this what you were searching for?
User	Success: 0/1	1	Yes

Figure 6.1: A stylized sample interaction. In this work, we use simulated dialog acts but the natural language glosses represent how such a dialog could look in an end application.

6.1 Contributions

This work was originally presented in Padmakumar and Mooney (2020b). My contribution to this work included designing and implementing the all components of the dialog system and the user simulator, as well as designing and conducting the experiments. Professors Raymond Mooney provided feedback that shaped the choice of algorithms and experimental design.

6.2 Motivation

Despite the high levels of accuracy that can be obtained in a variety of natural language interaction tasks, in natural language assistants that are a part of applications made available to users, it is important for the system to detect cases when it is likely to have misunderstood the user's intent, as well as cases where the user is ambiguous or has not provided all relevant information. Also, even if the system has been trained well, the specific environment in which a system is deployed may also contain domain specific vocabulary or concepts that were not encountered during training. For example, a dialog system in a shopping domain may need to be updated with the introduction of new clothing styles. A dialog system on an office robot may need to be updated as new types of office equipment become available and are brought into the office. Hence it is desirable for a natural language interaction to be conducted as part of a dialog where the system can ask both clarification questions to ensure that it has correctly understood the user's intent, as well as active learning questions to keep its underlying models up to date.

However, prior work on dialog and user interaction typically focuses either exclusively on clarification/information-seeking settings (Young et al., 2013; Padmakumar et al., 2017), or building/improving models through active learning (Woodward and Finn, 2017; Padmakumar et al., 2018). In this work, we attempt to combine these functions by learning a hierarchical dialog policy that can jointly perform *both* clarification and active learning to improve performance on an interactive language-based image retrieval task. We demonstrate that the use of active learning queries improves the effectiveness of clarification questions for objects that contain novel attributes, and that it is possible to trade off model improvement via active learning with both task completion and other dialog acts such as clarification.

Most task-oriented dialog tasks require the system to identify one or more goals specified by a user, using a slot-filling model (Young et al., 2013). These can be considered as learning to choose between a set of clarification questions that can confirm or obtain the value of various slots. However, for tasks such as natural language image retrieval, it is non-trivial to extend the slot-filling paradigm to perform clarification, as there is no standard set of slots that natural language descriptions of images can be divided into. Also, learned models are needed to identify components such as objects or attributes, and it is difficult to enumerate all expected types of these.

Some tasks such as GuessWhat?! (De Vries et al., 2017) or discriminative question generation (Li et al., 2017) allow the system to ask unconstrained natural language clarification questions. However in these settings, specially designed models are still needed to ensure that learned questions actually decrease the size of the search space (Lee et al., 2018; Zhang et al., 2018b). Such open ended questions are also difficult to answer in simulation, which is often necessary for learning good dialog policies. Hence, in these tasks, the system often learns to ask "easy" questions that can be reliably answered by a learned answering module (Zhu et al., 2017).

In this work, we explore a middle-ground approach with a form of attribute-based clarification (Farhadi et al., 2009). We use the term "attribute" to refer to a mix of concepts including categories such as “shirt” or “dress”, more conventional attributes such as colors, and domain specific attributes such as “sleeveless” and “V-neck”. Although we work with a dataset that contains a fixed set of attributes annotated for each image, we simulate the setting where novel visual attributes are encountered at test time. There exist some previous works that use visual attributes for clarification (Dindo and Zambuto, 2010; Parde et al., 2015) but these do not use this information for improving the underlying language understanding model.

Also related to this work is the area of interactive image retrieval such as allowing a user to mark relevant and irrelevant results (Nastar et al., 1998; Tieu and Viola, 2004), which acts as a form of clarification. Recent works allow users to provide additional feedback using language to refine search results (Guo et al., 2018; Bhattacharya et al., 2019; Saha et al., 2018). However, in these systems, the system does not take the initiative to help users narrow down their search space or explicitly learn new concepts. These directions are complementary to our work and can potentially be combined with it in the future.

6.3 Task Setup

We consider an interactive task of retrieving an image of a product based on a natural-language description. Given a set of candidate images and a natural-language description, the goal of the system is to identify the image being referred to. Before trying to identify the target image, the system can ask the user a combination of both clarification and active learning questions.

Similar to our prior work (chapters 4 and 5), in each interaction we present the system with two sets of images:

- An active test set O_A^{te} consisting of the candidate images that the description could refer to.
- An active training set O_A^{tr} which is the set of images that can be queried for active learning.

It is also presented with a description of the target image. Before attempting to identify the target, the system can ask clarification or active learning questions. We assume the system has access to a set of attributes W that can be used in natural language descriptions of products. Given these attributes, the types of questions the system can ask are as follows (see Figure 6.1 for examples of each):

- Clarification query - A yes/no query about whether an attribute $w \in W$ is applicable to the target.
- Label query: A yes/no query about whether an attribute $w \in W$ is applicable to a specific image i in the active training set O_A^{tr}.
- Example query: Ask for a positive example in the active training set O_A^{tr} for an attribute $w \in W$.

The dialog ends when the system makes a guess about the identity of the target and is considered successful if it this is correct. The goal of the task is to maximize the number of successful guesses, while also keeping dialogs as short as possible.

6.4 Methodology

6.4.1 Visual Attribute Classifier

We train a multilabel classifier for predicting visual attributes given an image. The network structure for the classifier is shown in Figure 6.2. We extract features $\phi(i)$ for the images using the penultimate layer of an Inception-V3 network (Szegedy et al., 2016) pretrained on ImageNet (Russakovsky et al., 2015). These are passed through two separate fully connected (FC) layers with ReLU activations, that are summed to produce the final representation $f(i)$ used for classification. This is converted into per-class probabilities $p(i)$ using a sigmoid layer with temperature correction (Guo et al., 2017). We obtain another set of per-class probabilities $p'(i)$ by passing the one of the intermediate representations $\psi'(i)$ through a sigmoid layer with temperature correction. Mathematically, given features $\phi(i)$ for image i, we have,

$$\begin{aligned} \psi(i) &= ReLU(w^T\phi(i) + b) \\ \psi'(i) &= ReLU(w'^T\phi(i) + b') \\ f(i) &= \psi(i) + \psi'(i) \end{aligned} \qquad \begin{aligned} p(i) &= \sigma(f(i) \odot \frac{1}{\tau}) \\ p'(i) &= \sigma(\psi'(i) \odot \frac{1}{\tau'}) \end{aligned}$$

where w, w', τ and τ' are learned vectors and b and b' are learned biases.

We train the network using a loss function that combines cross-entropy loss on $p(i)$ over all examples with the cross entropy loss over $p'(i)$ only for positive

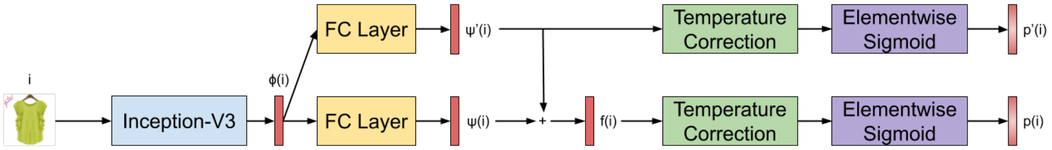


Figure 6.2: Visual Attribute Classifier.

labels. That is,

$$L = (1-\lambda)\sum_i y_i \log p(i) + (1-y_i)\log(1-p(i)) + \lambda \sum_i y_i \log p'(i)$$

where y_i is the label vector for image i. This forces part of the network to focus on positive examples for each class. This is required because we use a heavily imbalanced dataset where most classes have very few positive examples. We find this more effective than a standard weighted cross entropy loss, and the results in this paper use $\lambda = 0.9$. We also maintain a validation set of images labeled with attributes, that can be extended using active learning queries. Using this, we can estimate per-attribute precision, recall and F1. These metrics are used for tuning classifier hyperparameters and for dialog policy learning.

We initially experimented with alternate classifier designs such as binary SVMs using features extracted from Inception-V3 and fine-tuning Inception-V3 after altering the number of classes. We also experimented with alternate loss functions for fine-tuning Inception-V3 such as weighted cross entropy, and a ranking loss that maximizes the difference between the predicted probabilities of positive and negative attributes. Additionally, we compared fine-tuning all layers of Inception-V3 with training/fine-tuning only the extra/final layers. We used Inception-

V3 as the backbone network due to the results reported in the original paper (Guo et al., 2019).

However in contrast to the original paper, we found that our particular network design and loss function were required for obtaining reasonable classifier performance. Additionally, we found that it was required to initialize Inception-V3 with weights pretrained on ImageNet and train only the new layers on the iMaterialist dataset. These differences could be due to the differences in the data split. Our choice of data split results in many attributes always having a negative label during the training phase.

We also found that it was sometimes possible to obtain increases in the multilabel F1 metric proposed in the original paper (Guo et al., 2019) without any improvement on per attribute F1. For example, it is possible to obtain a multilabel F1 of 36.0 on the original validation set by identifying the 13 attributes with the largest number of positive examples, always predicting 1 for these, and always predicting 0 for the other attributes. Hence, we used the average per-attribute F1 to choose the design and hyperparameters of the classifier.

To initialize the classifier, we train for 100 epochs with a batch size of 8,192 and using RMSProp (Tieleman and Hinton, 2012) for optimization. We start with a learning rate of 0.1 which is decayed exponentially with a decay rate of 0.9 every 400 steps. For updating the classifier in between dialog batches, we use a batch size of 128 and perform a single epoch over images for which the label of at least one attribute has been updated. The optimizer and hyperparameters were tuned using grid search.

6.4.2 Grounding Model

We assume that a description is a conjunction of attributes, and use heuristics based on string matching to determine the set of attributes referenced by the natural language description. Let the subset of attributes referenced in the description be $W_d \subseteq W$.

Given an image i, we obtain the probability vector $p(i)$ as in section 6.4.1. We represent the probability of attribute w in this vector as $p_w(i)$. Then for each image i in the active test O_A^{te}, assuming independence of attributes, the probability that i is the target image, $b(i)$ is

$$b(i) = \prod_{w \in W_d} p_w(i) \tag{6.1}$$

Suppose we additionally obtain from clarifications stating that attributes $W_p \subseteq W$ apply to the target image, and attributes $W_n \subseteq W$ do not apply, this can be updated to

$$b(i) = \prod_{w \in W_d} p_w(i) \prod_{w \in W_p} p_w(i) \prod_{w \in W_n} (1 - p_w(i)) \tag{6.2}$$

Then, at any stage, the best guess the system can make is the image with max belief, that is

$$i_{guess} = \text{argmax}_{i \in O_A^{te}} b(i) \tag{6.3}$$

6.4.3 Initial Retrieval

We wish to simulate dialogs as refinements of an initial retrieval based on the product description. At the start of each batch of interactions, for each description corresponding to an image in the current *classifier_test* subset, we rank all images in this subset according to a variant of the score in Equation 6.1. Instead of directly using classifier probabilities, we threshold the probabilities $p_w(i)$ to obtain decisions $d_w(i)$. The threshold for each attribute is chosen to maximize the F1 score for that attribute on the current set of validation images and labels. This is initially the *classifier_test* subset of the *policy_pretrain* set, and gets expanded with a fraction of the labels obtained using active learning queries. This F1 score is also used in the baseline static policy (section 6.4.9) and features for the learned dialog policies (sections 6.4.6, 6.4.7 and 6.4.8).

Also, while the initial belief in the dialogs (Equation 6.1) only assumes that attributes mentioned in the description are positive for the target image, in the retrieval phase, we additionally assume that attributes not mentioned in the description are negative. Then,

$$\begin{aligned} b(i) &= \prod_{w \in W_d} p_w(i) \prod_{w \notin W_d} (1 - p_w(i)) \approx \prod_{w : d_w(i)=1} c_1 \prod_{w : d_w(i)=0} c_2 \\ s(i) &= \log b(i) = c_1 |\{w : d_w(i) = 1\}| + c_2 |\{w : d_w(i) = 0\}| \end{aligned}$$

We use the score $s(i)$ to rank images, where c_1 and c_2 are hyperparameters tuned on the *classifier_test* subset of the *policy_pretrain* set. Our reported results use $c_1 = 0.9$ and $c_2 = 0.1$.

6.4.4 MDP Formulation

We model each interaction as an episode in a Markov Decision Process (MDP) where the state consists of the images in the active training and test sets, the attributes mentioned in the target description, the current parameters of the classifier, and the set of queries asked and their responses. At each state, the agent has the following available actions:

- A special action a_g for guessing – the image is chosen using Equation 6.3.
- A set of clarification actions A_C – one for each attribute $w \in W$.
- A set of actions A_L corresponding to possible active learning queries – one example query per attribute and one label query corresponding to each pair (w, i) for $w \in W$, $i \in O_A^{tr}$.

We do not allow actions to be repeated. We learn a hierarchical dialog policy composed of 3 parts –

- A clarification policy π_C to choose the best possible clarification action $a_C^* \in A_C$ in the current state, using features $\phi_C(s, a_C)$ (section 6.4.6) for action a_C in state s.
- An active learning policy π_{AL} to choose the best possible active learning query $a_{AL}^* \in A_{AL}$ in the current state, using features $\phi_{AL}(s, a_{AL})$ (section 6.4.7) for action a_{AL} in state s. We reduce the action space of the active learning policy to one example query action per attribute, and one label query action per attribute corresponding to the image with probability closest to 0.5

for that attribute ($argmin_{i \in O_A^{tr}} |p_w(i) - 0.5|$).

- A decision policy π_D that chooses between a_g, a_C^* and a_{AL}^*, using features $\phi_D(s, a)$ (section 6.4.8) for action a in state s.

An episode ends either when the guess action is chosen, or when a dialog length limit is reached, at which point the system is forced to make a guess. If the episode ends with a correct guess, the agent gets a large positive reward. Otherwise the agent gets a large negative reward at the end of the episode. Additionally, the agent gets a small negative reward for each query to encourage shorter dialogs. In our experiments, we treat these rewards as tunable hyperparameters. We fixed an arbitrary dialog length representing what we expected would be the maximum number of questions a user would be willing to answer, and selected hyperparameters that results in the maximum dialog success rate within this dialog length on validation data. We do not attempt to answer the question of how to optimally balance the two metrics of dialog success rate and dialog length. We believe that this balance is likely to be domain dependent and would need to be chosen by a user study that measures how many questions people are typically willing to tolerate and to what extent this varies with dialog success rate. It is difficult to conduct such a study on a crowdsourced platform since crowdworkers are not sufficiently invested in the success of the dialog interactions.

6.4.5 Policy Learning

We experiment with using both Q-learning and A3C (Mnih et al., 2016) for policy learning. We use the same model structure for all three policies but no shared parameters. In the following discussion about the model structure for the policy, we will refer to state-action features $\phi(s, a)$ and policy $\pi(s, a)$, which is intended to represent the appropriate input and output for each policy.

For Q-learning we use a single-layer neural network with hidden layer size 100, whose input is the feature vector $\pi(s, a)$, and output is the Q-value $Q_\pi(s, a)$ of action a in state s under policy π. Suppose action a is taken in state s resulting in reward r and next state s', we update the network with new targets:

$$Q_\pi(s, a) = r + \gamma * \max_{a'} Q_\pi(s', a')$$

In the policy training phase, we choose ϵ-greedy actions with $\epsilon = 0.1$ and in the policy validation and testing phases, we choose actions greedily. We use $\gamma = 1.0$.

For A3C, as a critic, we use a network similar to Q-learning, predicting $Q_\pi(s, a)$. The actor uses a policy representation:

$$\pi_\theta(s, a) = \frac{e^{\theta^T \phi(s,a)}}{\sum_{a'} e^{\theta^T \phi(s,a')}}$$

where θ is a learned parameter vector. Suppose action a is taken in state s resulting in reward r and next state s', the critic network is updated similar to Q-learning and the actor weights are updated as:

$$\begin{aligned} & \theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s, a) A_\pi(s, a) \\ \text{where} \quad & A_\pi(s, a) = r + \gamma * V_\pi(s') - V_\pi(s) \\ \text{where} \quad & V_\pi(s) = \sum_a \pi_\theta(s, a) Q_\pi(s, a) \end{aligned}$$

where $Q_\pi(s, a)$ is the estimate from the critic network. We use $\gamma = 1.0$ and $\alpha = 0.01$.

6.4.6 Clarification policy features

For the clarification policy π_C, we need to extract features of a state-action pair, that is, features describing a candidate clarification given the current state. The features should provide information that could be used to identify useful clarifications. We use the following features -

- Metrics about the current beliefs $\{b(i) : i \in O_A^{te}\}$ and what they would be for each possible answer, if the question were asked:
  - Entropy: A higher entropy suggests that the agent is more uncertain. A decrease in entropy could indicate a good clarification.
  - Top two highest beliefs and their difference: A high value of the maximum belief, or a high difference between the top two beliefs could indicate that the agent is more confident about its guess. An increase in these could indicate a good clarification.
  - Difference between the maximum and average beliefs: A large difference suggests that the agent is more confident about its guess. An increase in these could indicate a good clarification.
- Information gain of the query as calculated in section 6.4.2.
- Current F1 of the attribute associated with the query: The system is likely to make better clarifications using attributes with high predictive accuracy.

6.4.7 Active learning policy features

For the active learning policy π_{AL}, we need to extract features describing a candidate label or example query given the current state. We use the following features:

- Current F1 of the attribute associated with the query, since the system is likely to benefit more from improving an attribute whose current predictive accuracy is not high.
- Fraction of previous dialogs in which the attribute has been used, since it is beneficial to focus on frequently used attributes that will likely benefit future dialogs.
- Fraction of previous dialogs using the attribute that have been successful, since this suggests that the attribute may be modelled well enough already.
- Whether the query is off-topic (i.e. opportunistic), since this would not benefit the current dialog.

Additionally in label queries,

- For query (w, i), $|p_w(i) - 0.5|$ as a measure of (un)certainty.
- Average cosine distance of the image to others in the dataset; this is motivated by density weighting to avoid selecting outliers.
- Fraction of k-nearest neighbors of the image that are unlabelled for this attribute, since a higher value suggests that the query could benefit multiple images.

6.4.8 Decision policy features

For the decision policy π_D, we need features of the current state that allow the system to trade-off between clarification, active learning, and guessing.

- Features of the current belief as in section 6.4.6. These can be used to determine whether a guess is likely to be successful.
- Information gain of the best clarification action – to decide the utility of the clarification.
- Margin from the best active learning query if it is a label query – to decide the utility of the label query.
- F1 of attributes in clarification and active learning queries. High F1 is desirable for clarification and low F1 for active learning.
- Mean F1 of attributes in the description. A high value suggests that the belief is more reliable.
- Number of dialog turns completed.

6.4.9 Baseline static policy

As a baseline, we use an intuitive manually-designed static policy that is also hierarchical and was tailored to perform well in preliminary experiments.

The static clarification policy chooses the query (among those with F1 > 0) with maximum information gain. Ties are broken using F1 of the attribute in the query. This is based on prior work in goal-oriented dialog that attempts to estimate the information gain of a clarification question (Lee et al., 2018). In this setting, the agent asking questions needs to identify a target object among a set of candidate objects, and can ask clarification questions to help identify the target. Let C, Q_t and A_t be random variables corresponding to the target object, question in turn t and answer in turn t respectively, and c, q_t and a_t represent specific values of these variables. Then the information gain from asking question q_t, given previous questions $q_{1:t-1}$ and their answers $a_{1:t-1}$ is

$$\begin{aligned} & I[C, A_t; q_t, a_{1:t-1}, q_{1:t-1}] \\ = & \sum_{a_t} \sum_{c} p(c|a_{1:t-1}, q_{1:t-1}) p(a_t|c, q_t, a_{1:t-1}, q_{1:t-1}) \ln \frac{p(a_t|c, q_t, a_{1:t-1}, q_{1:t-1})}{p(a_t|q_t, a_{1:t-1}, q_{1:t-1})} \end{aligned}$$

$$\text{where } p(a_t|q_t, a_{1:t-1}, q_{1:t-1}) = \sum_{c} p(c|a_{1:t-1}, q_{1:t-1}) p(a_t|c, q_t, a_{1:t-1}, q_{1:t-1})$$

In our case, possible targets c correspond to possible images i. As in prior work (Lee et al., 2018), $p(c|a_{1:t-1}, q_{1:t-1})$ corresponds to the estimated likelihood of target c given the conversation history, which in our case is $b(i)$. We also make an additional assumption that the answer to question q_t depends only on the target image and not on prior questions and answers. Hence:

$$p(a_t|c, q_t, a_{1:t-1}, q_{1:t-1}) = p(a_t|c, q_t)$$

which in our case is $P(a_t|q_t, i)$. Since our questions q_t are attributes and the attribute classifier is expected to provide the probability that attribute q is true for image i, we get $P(1|q, i) = p_q(i)$ and $P(0|q, i) = 1 - p_q(i)$. In practice, we observe that the classifier does not produce well-calibrated probabilities despite the use of temperature correction (Guo et al., 2017), and we believe that this contributes to the poor performance of the static clarification policy.

Substituting these, we get information gain for question q, which we represent using $J(q)$, as:

$$J(q) = \sum_{i \in O_A^{te}} \sum_{a \in \{0,1\}} b(i) P(a|q,i) \ln \left(\frac{P(a|q,i)}{\sum_i b(i) P(a|q,i)} \right)$$

Since we observe in experiments that the static clarification policy does not perform well, we also use a special oracle clarification phase when collecting dialogs using the static policy to train the learned policy. The oracle policy simulates all possibel clarifications and selects the one that maximally increases the belief of the correct target image.

The static active learning policy has a fixed probability of choosing label queries and example queries. Uncertainty sampling is used to select the label query (w, i) with minimum $|p_w(i) - 0.5|$. An example query is chosen uniformly at random from the candidates. The decision policy initially chooses clarification if the information gain is above a minimum threshold, and the highest belief is below a confidence threshold. After a maximum number of clarifications, it chooses active learning until another threshold on the dialog length before guessing.

6.5 Experimental Setup

6.5.1 Dataset

To address a potential shopping application, we simulate dialogs using the iMaterialist Fashion Attribute dataset (Guo et al., 2019), consisting of images from the shopping website Wish[1] each annotated for a set of 228 attributes. We scraped product descriptions for the images in the train and validation splits of the dataset for which attribute annotations are publicly available. After removing products whose images or descriptions were unavailable, we had 648,288 images with associated product descriptions and annotations for the 228 attributes.

We create a new data split (as in chapter 5) to ensure that the learned dialog policy generalizes to attributes not seen during policy training. We divide the set of attributes into 4 subsets – *policy_pretrain*, *policy_train*, *policy_val* and *policy_test*. Using these, we divide the images into 4 subsets as follows:

- All images having a positive label for any of the attributes in *policy_test* subset form the *policy_test* set of images.
- Of the remaining images, the images with a positive label for any attribute in the *policy_val* form the *policy_val* set of images.
- Of the remaining images, the images with a positive label for any attribute in the *policy_train* form the *policy_train* set of images.
- The remaining images form the set of *policy_pretrain* images.

---

[1] `https://www.wish.com/`

This iterative procedure ensures that images in each of the *policy_train*, *policy_val* and *policy_test* result in the introduction of new attributes for which the classifier is not already trained. Each of these is then split into subsets *classifier_training* and *classifier_test* by a uniform 60-40 split.

The *policy_pretrain* data is used to pretrain the multi-class attribute classifier. We use its *classifier_training* subset of images for training and its *classifier_test* subset to tune hyperparameters. The *policy_train* data is then used to learn the dialog policy. The *policy_val* data is used to tune hyperparameters as well as choose between RL algorithms. Results are reported for *policy_test* data.

We wish to simulate dialogs as refinements of an initial retrieval based on the product description. For the description of each image in the current *classifier_test* subset, we rank all other images in this subset according to a simplified version of the score in equation 6.1 (details in Appendix 6.4.3). From the images which get ranked within the top 1000 for their corresponding description, we sample target images for each interaction. The active test set for the interaction consists of the top 1000 images as ranked for that description. We randomly sample 1000 images from the appropriate *classifier_training* subset to form the active training set.

In each interaction, the description of the target image is provided to the agent to start the interaction. The annotated attributes are used to answer queries from the system. This simulation procedure is similar to that used in chapter 5 but the answers to questions are less noisy in our simulation as all attributes are annotated as positive or negative for all images.

6.5.2 Experiment Phases

We run dialogs in batches of 100 and update the classifier and policies at the end of each batch. This is followed by repeating the retrieval step for all descriptions in the *classifier_test* subset before choosing target images for the next batch of dialogs. The experiment has the following phases:

- Classifier pretraining: We pretrain the classifier using annotated attribute labels for images in the *classifier_training* subset of the *policy_pretrain* set. This ensures that we have some reasonable clarifications at the start of dialog policy learning.
- Policy initialization: We initialize the dialog policy using experience collected using the baseline static policies (section 6.4.9) for the decision and active learning policies, and an oracle [2] to choose clarifications. This is done to speed up policy learning. The dialogs for this phase are sampled from the set of *policy_train* images.
- Policy training: This phase consists of training the policy using on-policy experience, with dialogs again sampled from the set of *policy_train* images.
- Policy testing: We reset the classifier to the state at the end of pretraining. This is done to ensure that any performance improvement seen during testing are due to queries made in the testing phase. This is needed both for fair comparison with the baseline and to confirm that the system can generalize to novel attributes not seen during any stage of training. Dialogs are sampled for

---

[2] The oracle tries each candidate clarification and returns the one that maximally increases the belief of the target image.

this from the *policy_val* set for hyperparameter tuning and from the *policy_test* set for reported results.

6.6 Results and Discussion

We initialize the policy with 4 batches of dialogs, followed by 4 batches of dialogs for the training phase, and 5 batches of dialogs in the testing phase. We compare the fully learned policy with hierarchical policies that consist of keeping one or more of the components static. We also compare the choice of Q-Learning or A3C (Mnih et al., 2016) as the policy learning algorithm for each learned policy. The results are presented in table 6.1. We evaluate policies on the fraction of successful episodes in the final test batch, and the average dialog length.

Ideally we would like the system to have a high dialog success rate while having as low a dialog length as possible. We observe that the using a learned policy for all three functions results in a significantly more successful dialog system (according to an unpaired Welch t-test with $p < 0.05$) than most conditions in which one or more of the policies are static. The exception is the case when the decision policy is static and the clarification and active learning policies are learned, in which case the difference is not statistically significant. Note that the mean success rate of the fully learned policy is still considerably higher, but due to the relatively small size of our batches (100 dialogs) and the variance in RL policies across test runs, we believe that it is more desirable to rely on measures of statistical significance such as the p-values from a T-test. The fully learned policy also uses significantly shorter dialogs than all conditions with a static decision policy. Some other conditions

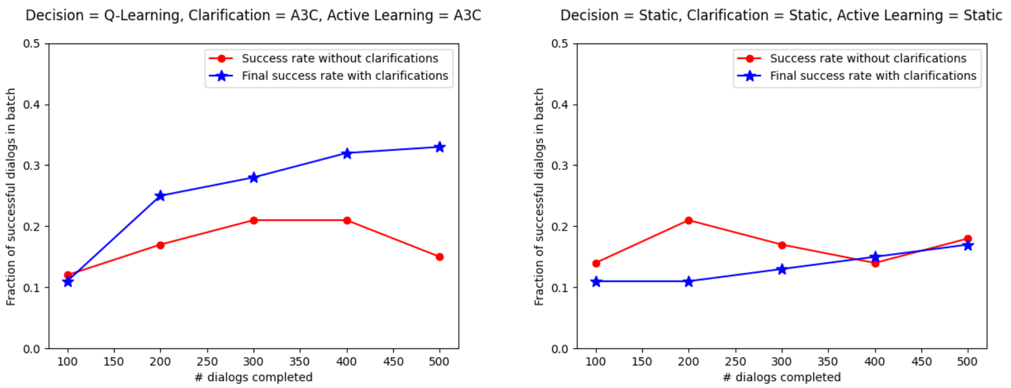
Table 6.1: Results from the final batch of the test phase. * indicates the conditions whose performance is comparable to the best condition (in bold).

Decision Policy Type	Clarification Policy Type	Active Learning Policy Type	Fraction of Successful Dialogs	Average Dialog Length
Q-Learning	Q-Learning	Q-Learning	0.12	15.45
Q-Learning	Q-Learning	A3C	0.18	16.96
Q-Learning	Q-Learning	Static	0.14	11.83
Q-Learning	A3C	Q-Learning	0.06	1.0
Q-Learning	A3C	A3C	**0.33**	9.4
Q-Learning	A3C	Static	0.15	14.16
Q-Learning	Static	Q-Learning	0.17	13.96
Q-Learning	Static	A3C	0.09	1.0
Q-Learning	Static	Static	0.17	3.81
A3C	Q-Learning	Q-Learning	0.09	20.0
A3C	Q-Learning	A3C	0.19	20.0
A3C	Q-Learning	Static	0.17	20.0
A3C	A3C	Q-Learning	0.13	20.0
A3C	A3C	A3C	0.09	20.0
A3C	A3C	Static	0.15	20.0
A3C	Static	Q-Learning	0.12	20.0
A3C	Static	A3C	0.13	20.0
A3C	Static	Static	0.16	20.0
Static	Q-Learning	Q-Learning	*0.29	20.0
Static	Q-Learning	A3C	*0.24	20.0
Static	Q-Learning	Static	0.1	20.0
Static	A3C	Q-Learning	*0.24	20.0
Static	A3C	A3C	*0.27	20.0
Static	A3C	Static	0.14	20.0
Static	Static	Q-Learning	0.15	20.0
Static	Static	A3C	0.16	20.0
Static	Static	Static	0.17	20.0

result in shorter dialogs, but these are unable to exploit the clarification and active learning actions enough to result in a success rate comparable to the fully learned policy.

Figure 6.3 plots the success rate across test batches, and the expected success rate if the system was forced to guess without clarification, for the fully learned, and fully static policies. We can evaluate the utility of clarifications by comparing these two success rates in each test batch. We can also examine the effect of active learning with or without clarification by examining the difference in the relevant metric between the first test batch where the classifier has not yet been improved with labels from any active learning queries, and the final test batch where the classifier has been improved using the labels acquired during previous test batches. We find that in the case of the fully static policy, there is no statistically significant improvement, either in the expected initial success rate without clarifications, or in the final success rate, between the first and last test batch. This suggests that neither the static active learning policy, nor its combination with the static clarification policy are capable of improving the system's performance.

However, in the case of the fully learned policy, we observe a statistically significant improvement in the final success rate, but not the initial success rate without clarifications. This suggests that while a learned active learning policy by itself is not sufficient to improve the system's success rate, the combination of learned active learning and clarification policies is sufficient to improve the system's success rate. We also observe that while the difference between the initial and final success rate is initially not significant, it increases across batches, and becomes significant in the last two batches. This suggests that the clarification policy by itself is also insufficient for improvement, and the combination of the two is required to improve the system's success rate.



(a)

(b)

Figure 6.3: Comparison of guess success rate with, and without clarifications across test batches.

We believe that the reason for the relatively poor performance of the static clarification and active learning policies is that the classifier is not sufficiently accurate, and does not produce well calibrated probabilities, due to the heavy imbalance in the dataset. We also believe that the poor performance of the learned decision policy is because individual features of the clarification and active learning policies are insufficient to score the goodness of these queries. On the other hand, the learned policies are able to learn to properly adjust for this miscalibration. We also believe that with more training dialogs or a different state-action representation, it may be possible to also learn a decision policy that outperforms the static decision policy.

We also examine the statics of different dialog acts taken by the best learned policy to qualitatively assess how it differs from the static policy. The results are shown in Figure 6.4. We observe that the average number of clarification questions

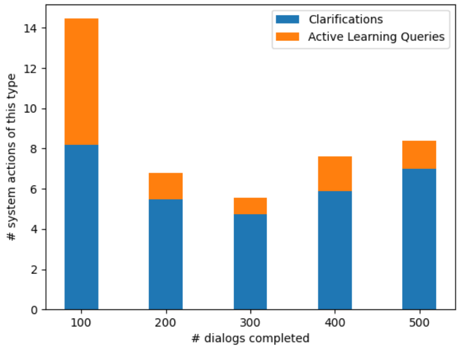


Figure 6.4: Dialog turns of each type taken by the best learned policy in the test phase.

remains relatively stable across test batches, whereas active learning questions are concentrated into the first test batch, with far fewer queries in later test batches.

6.7 Summary

We demonstrate how a combination of RL learned policies for choosing attribute-based clarification and active learning queries can be used to improve an interactive system that needs to retrieve images based on a natural language description, while encountering novel attributes at test time not seen during training. We demonstrate that in a challenging dataset where it is difficult to obtain an accurate attribute classifier, learned policies for choosing clarification and active learning queries outperform strong static baselines. We further show that in this challenging setup, a combination of learned clarification and active learning policies is necessary to obtain improvement over directly performing retrieval without interaction. In the future, we would like to evaluate whether our learned policy generalizes to other datasets which may have a different set of classifiers, and possibly different underlying classifiers. Since our policy representation depends on features based on evaluating classifiers, the general method is still applicable if the underlying classifiers change. However, further empirical work is needed to determine whether such generalization is effective in practice.

Chapter 7

Human Evaluation of Dialog Policy Learning for Joint Clarification and Active Learning Queries

In chapter 6, we trained a hierarchical dialog policy to jointly perform both clarification and active learning in the context of an interactive language-based image retrieval task. We demonstrated that jointly learning a dialog policy to perform both these functions improved an interactive system that needs to retrieve images based on a natural language description, while encountering novel attributes at test time not seen during training.

However, in chapter 6, we only evaluated the learned dialog policy using simulated dialogs. An important possibility to consider when training dialog systems in simulation is that the system may exploit imperfections in the simulation to achieve high performance that is not replicable in real interactions with humans.

In this chapter, we discuss a human evaluation we conducted using Amazon Mechanical Turk to compare our best learned policy with the static baseline.

7.1 Models

We use the same grounding model (section 6.4.2), baseline static policy (section 6.4.9), and learned policy (sections 6.4.4 – 6.4.8) from chapter 6. However, all experiments in this chapter use only the best combination of RL algorithms from the results in chapter 6 – A3C for choosing clarification and active learning queries, and Q-Learning for deciding between clarification, active learning and guessing. We retained most hyperparameters from chapter 6 but had to re-optimize the rewards using grid search due to changes in the experimental setup that will be discussed in section 7.3.

7.2 Experimental Design Changes

We had to make two important changes to the experimental setup when moving from a simulated environment to conducting interactions with real users on Amazon Mechanical Turk:

- **Sensitive content** – We found that the iMaterialist dataset contained many images (for example, images of undergarments and lingerie items) that may not be appropriate for use in experiments with human users without content warnings. We manually identified a list of 24 attribute labels that may be indicative of such images, and removed images have a positive value for any of these labels from the *policy_train*, *policy_val* and *policy_test* data splits (see section 6.5.1 for details of how the data splits were created).
- **Shorter Descriptions** – We found using pilot studies that workers on Amazon Mechanical Turk typically mentioned fewer of our annotated attributes than what is common in product descriptions. In this condition, both the original static and learned policies had very low success rates. Hence we switched to a simpler task setup described in section 7.3.

7.3 Modified Task Setup

We observed from pilot studies that user descriptions typically contained 1-3 attributes that were annotated in our dataset, and hence could be recognized by our classifier. This was very different from product descriptions that typically had more attributes specified. With this change, we observed that both the baseline static policy, and the learned policy from chapter 6 were rarely able to successfully retrieve the correct target images.

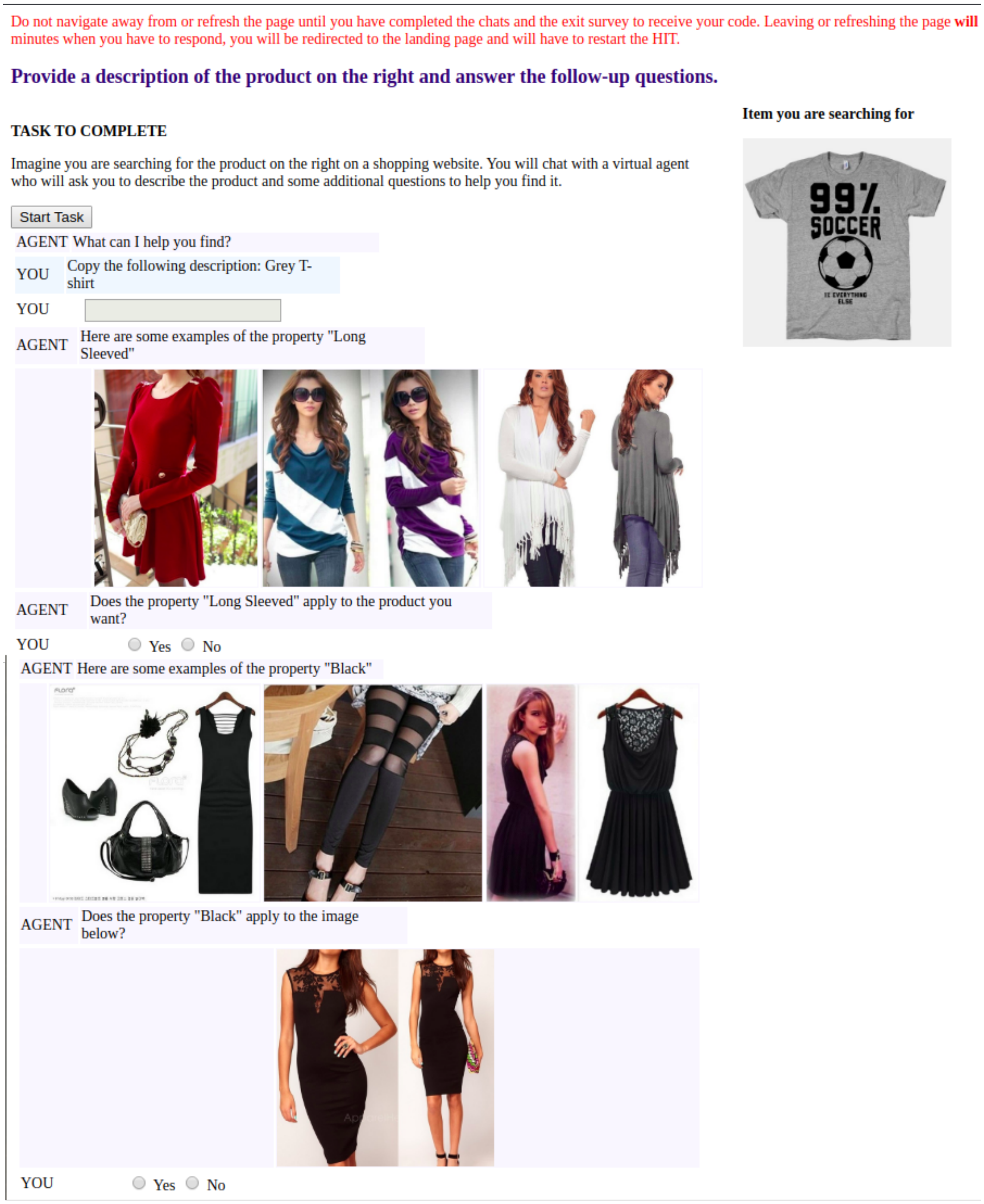
We first experimented with modifying the simulation setup from chapter 6, by randomly sampling one attribute from each product description to start the dialog, instead of providing all extracted attributes, to mimic the conditions on Amazon Mechanical Turk. In this modified simulation setup, we found that as we observed on Amazon Mechanical Turk, both the static and learned policies were rarely able to successfully retrieve the correct target images. We also tried learning a new policy in this setup, as in chapter 6, but the success rate of the learned policy was still too low to be useful for human evaluation.

Hence, we experimented with simpler task setups, and finally settled on the following setup. The active test set was created by randomly sampling 100 images from the relevant *classifier_test* subset of images, and only of these was randomly selected to be the target region. The active training set was created by randomly sampling 100 images from the relevant *classifier_train* subset of images. For training in simulation, we randomly sampled one attribute from those extracted from the product description of the target image (see section 6.5.1 for details of how we extracted attributes from product descriptions), and provided this as the initial description. In the new setup, we tuned with the reward structure to maximize the dialog success rate. We obtained best results when providing a reward of +25 for every correct guess, -25 for every incorrect guess, and -1 for all clarification and active learning queries.

We trained a new policy with 10 batches of initialization, 10 batches of training and 5 batches of testing in simulation (see section 6.5.2 for details about experiment phases). We then used the final policy, and classifiers at the end of the test phase in interactions with users on Amazon Mechanical Turk to evaluate how well the learned system transfers.

7.4 User Interface Challenges

We originally wanted to create an interface for workers that looked like a chat, but also enabled them to easily understand the task. As a result, we created a qualification task as shown in Figure 7.1. We provided an initial image and an example description to prime users to focus on the types of attributes our system used. Users had to copy the description, following which they would get the clarification query shown, and later the active learning query shown. After answering the active learning query, they would get a completion code that could be provided on Amazon Mechanical Turk to obtain payment. Our goal was to choose relatively unambiguous clarification and active learning queries for the qualification task, and only allow users to provided the answers we expected to both queries to participate in our main experiment.



Do not navigate away from or refresh the page until you have completed the chats and the exit survey to receive your code. Leaving or refreshing the page **will**
minutes when you have to respond, you will be redirected to the landing page and will have to restart the HIT.

Provide a description of the product on the right and answer the follow-up questions.

Item you are searching for

TASK TO COMPLETE

Imagine you are searching for the product on the right on a shopping website. You will chat with a virtual agent who will ask you to describe the product and some additional questions to help you find it.

AGENT What can I help you find?

YOU Copy the following description: Grey T-shirt

YOU

AGENT Here are some examples of the property "Long Sleeved"

AGENT Does the property "Long Sleeved" apply to the product you want?

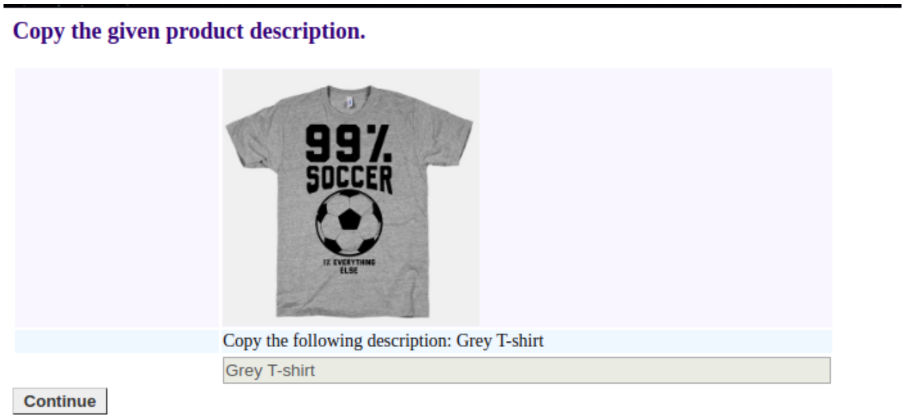
YOU ○ Yes ○ No

AGENT Here are some examples of the property "Black"

AGENT Does the property "Black" apply to the image below?

YOU ○ Yes ○ No

Figure 7.1: Initial Qualification Task Interface on Amazon Mechanical Turk.

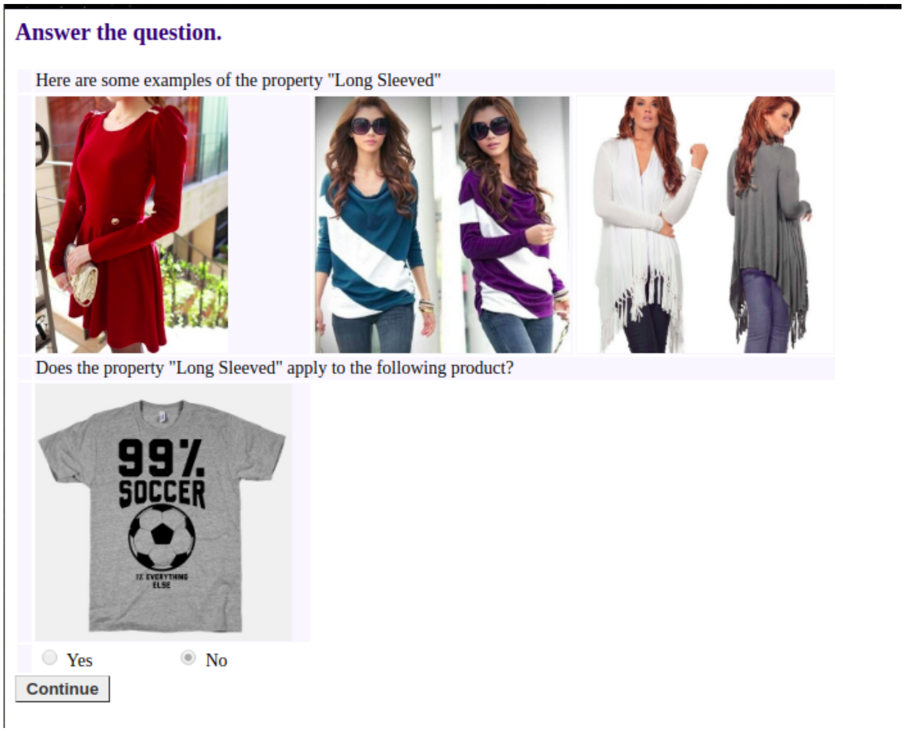


Copy the given product description.

Copy the following description: Grey T-shirt

Grey T-shirt

(a) Step 1: Copy an example description.



Answer the question.

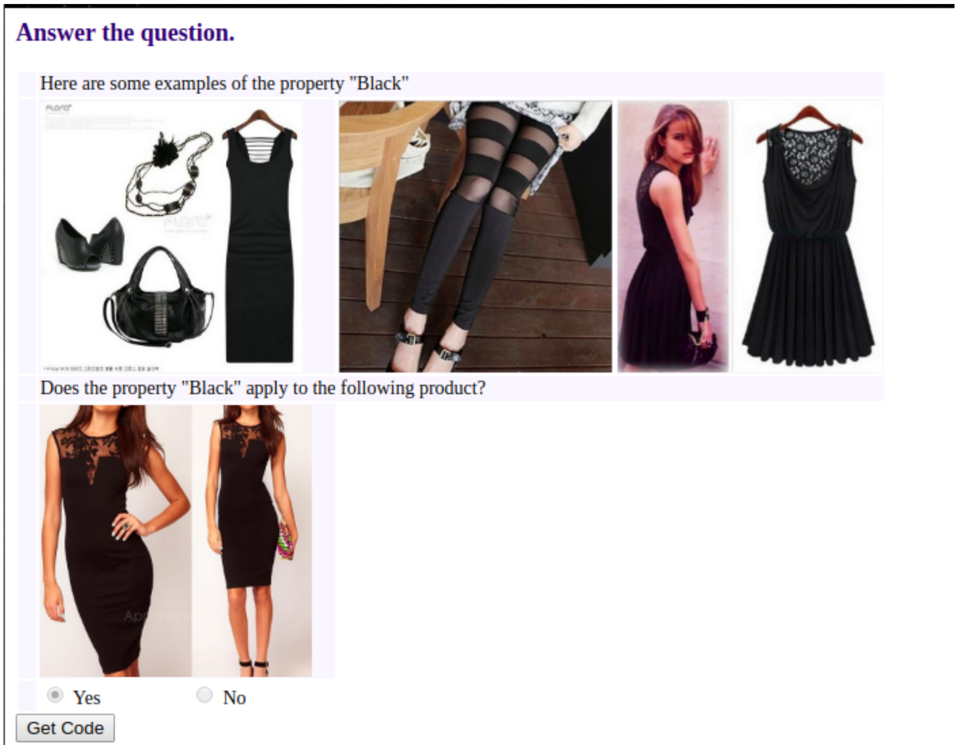
Here are some examples of the property "Long Sleeved"

Does the property "Long Sleeved" apply to the following product?

○ Yes ● No

(b) Step 2: Clarification Question.

Figure 7.2: Final qualification interface for Amazon Mechanical Turk *(Continued on next page)*.



Answer the question.

Here are some examples of the property "Black"

Does the property "Black" apply to the following product?

● Yes ○ No

(c) Step 3: Active Learning Question.

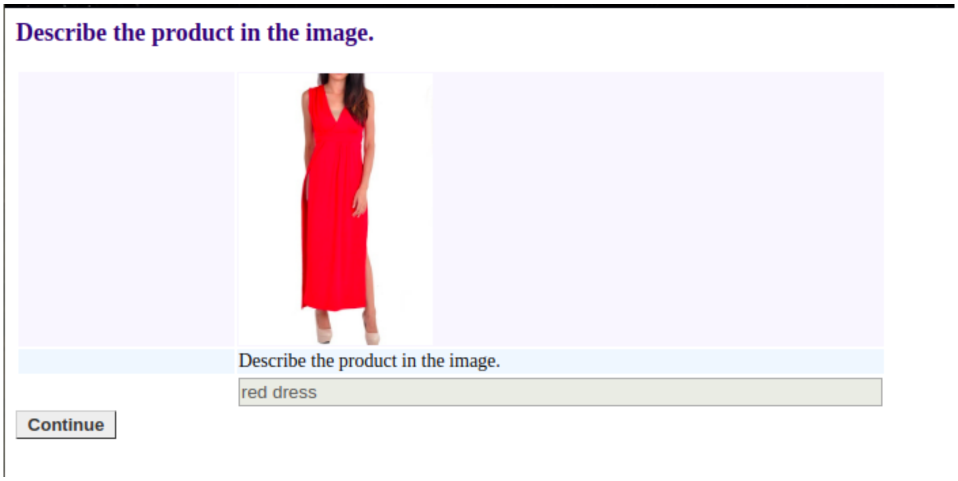
Figure 7.2: Final qualification interface for Amazon Mechanical Turk.

We provided manually chosen example images for each attribute because while some attributes such as colors may be known to all workers, other such as “Argyle” or “Herringbone” may not be familiar to the average worker, but we believe can be identified given examples.

We required workers to have completed at least 1000 HITs and have at least a 95% approval rate on their previous HITs. However, in a pilot study of 40 workers, we found that even with these qualifications, only 3 workers correctly copied the description and answered the questions as we expected. 16 out of 40 workers did not even follow the instruction to copy the description. However, all 40 workers correctly copied their completion codes.

Hence, we changed the interface to provide one question per page, and allow the worker to move to the next page only after they had provided an answer to the question on the current page. Hence, the above task was divided into 3 pages as shown in Figure 7.2. We then found using another pilot study of 20 workers with the same qualifications that this time, all 20 workers completed all verification tasks correctly. Hence, we decided to use a similar interface for our main dialog experiments as well – providing each question in a new page with question specific instructions. While this does not allow us to study whether workers view this interaction as a natural conversation, we wished to study whether we could at least replicate our results about dialog success. Further work is needed to develop an appropriate user interface for this task that allows workers to interact in a more dialog-like fashion, while also not getting confused by the amount of information provided.

For the final experiment, we added a page at the start that showed the target image and asked users to describe the target image (Figure 7.3). This was followed by pages similar to figures 7.2b and 7.2c as relevant, based on the dialog system's responses. Finally when the dialog system made a guess, we provided users with their original description, and asked them whether the image matched this description. We also asked users to fill out an exit survey rating the questions on a Likert scale of 0-4.



Describe the product in the image.

Describe the product in the image.

red dress

Figure 7.3: Final dialog task on Amazon Mechanical Turk – description page.

7.5 Results and Discussion

We obtained a new learned policy with the new experimental setup with 10 batches of initialization, 10 batches of training and 5 batches of testing in simulation. We then used the final policy, and classifiers at the end of the test phase in interactions with users on Amazon Mechanical Turk to evaluate how well the learned system transfers to human users. We present the results in table 7.1. We report the fraction of successful dialogs (in which the system guesses the correct target image), and average dialog length.

Firstly, we observe that in this condition, the improvement in the learned policy is considerably higher than the static policy. However, its average dialog length does not decrease. Qualitatively, we observe that in the test phase, the learned policy initially has a low rate of clarification, and high rate of active learning, which is reversed as the dialogs progress. We also observe that clarification is significantly

Table 7.1: Results of the static and new learned policies at the end of the test phase in simulation and in interactions on Amazon Mechanical Turk. Bold indicates a statistically significant improvement over the baseline ($p < 0.05$) and italic indicates trending significance ($p <= 0.1$) according to an unpaired Welch t-test

Policy	Simulation – Fraction of Successful Dialogs	Simulation – Average Dialog Length	AMT – Fraction of Successful Dialogs	AMT – Average Dialog Length
Static	0.23	20.0	0.06	19.16
Learned	**0.65**	20.0	*0.16*	18.86

more beneficial in this setting compared to that used in chapter 6.

We also observe a considerable drop in success rate for both the static and learned policy, in human interactions. However, the learned policy remains more successful than the static policy, trending towards significance ($p <= 0.1$) according to an unpaired Welch t-test.

We speculate that the drop in performance may be because of differences between the labels in the original dataset, and those provided by workers, making our classifiers less effective. This may be due to a couple of reasons:

- The labelling of clothing attributes may be somewhat subjective, or may be affected by the display settings used by workers. For example, although we expected workers to describe the image in figure 7.3 as “red”, a surprisingly high fraction of workers described it as “orange”.
- The dataset includes attributes about clothing material such as “satin” and “velvet” that may be harder for workers to label using only images.

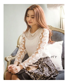
We had expected fewer such differences since the original attributes of the dataset were also labeled by crowdworkers. However, those workers were also provided the product description in addition to the image to perform the labeling, which may have enabled them to label more difficult attributes that are less easily recognizable for human users with just images. In future, we can experiment with simulations that introduce noise into the annotated labels when answering clarification labels to attempt to train a dialog system that generalizes better to real world conditions. However, we speculate that this is likely to require more accurate classifiers than those used in our current experiments, so that a correct prediction on a single consistently labeled attribute can counteract the effects of variance in labels of other attributes.

A sample interaction is included in table 7.2. While the agent is eventually successful, we can see that the interaction is fairly long. In practical applications, we would need to provide a mechanism to end the dialog earlier if necessary. We also observe that some of the questions can appear strange to human users due to category violations – for example asking whether the properties “Heels” and “Sneakers” apply to a T-shirt. In an embodied system, facial feedback can potentially be used to identify such incongruous questions and provide a negative reward for them, so that the dialog policy can learn to avoid such questions.

7.6 Summary

In this chapter we discuss many challenges that we faced when trying to test our dialog policy from chapter 6, that was trained in simulation, on interactions with human users on Amazon Mechanical Turk. We found that workers found the system confusing when it was presented as a dialog system, and instead had to be presented with individual questions on separate web pages. We also had to simplify the experimental setup to adapt to the differences between descriptions provided by workers, and product descriptions from the original dataset. Despite these simplifications, we observed a significant drop in performance of dialog policies in human evaluation, compared to simulation. However, we found that our fully learned dialog policy transferred modestly better to human evaluation than the static baseline.

Table 7.2: A sample interaction from Amazon Mechanical Turk.

Agent	What can I help you find?
User	Black Off-the-Shoulder Dress
Agent	Here are some examples of the property ”Female”
Agent	Does the property ”Female” apply to the product you want?
User	Yes
Agent	Here are some examples of the property ”Round Neck”
Agent	Does the property ”Round Neck” apply to the product you want?
User	No
Agent	Here are some examples of the property ”Lace” 
Agent	Does the property ”Lace” apply to the product you want?

Continued on next page

Table 7.2: A sample interaction from Amazon Mechanical Turk.

User	No
Agent	Here are some examples of the property ”Teal”
Agent	Does the property ”Teal” apply to the product you want?
User	No
Agent	Here are some examples of the property ”Paisley”
Agent	Does the property ”Paisley” apply to the product you want?
User	No
Agent	Here are some examples of the property ”Long Sleeved”
Agent	Does the property ”Long Sleeved” apply to the product you want?
User	No
Agent	Here are some examples of the property ”Quilted”
Agent	Does the property ”Quilted” apply to the product you want?
User	No
Agent	Here are some examples of the property ”Spaghetti Straps”
Agent	Does the property ”Spaghetti Straps” apply to the product you want?

Continued on next page

Table 7.2: A sample interaction from Amazon Mechanical Turk.

User	No
Agent	Here are some examples of the property ”Tunic”
Agent	Does the property ”Tunic” apply to the product you want?
User	No
Agent	Here are some examples of the property ”Pin Stripes”
Agent	Does the property ”Pin Stripes” apply to the image below?
User	No
Agent	Here are some examples of the property ”Pink”
Agent	Does the property ”Pink” apply to the image below?
User	No
Agent	Here are some examples of the property ”Organza”
Agent	Does the property ”Organza” apply to the image below?
User	No

Continued on next page

Table 7.2: A sample interaction from Amazon Mechanical Turk.

Agent	Here are some examples of the property ”Heels”
Agent	Does the property ”Heels” apply to the image below?
User	No
Agent	Here are some examples of the property ”Sneakers”
Agent	Does the property ”Sneakers” apply to the image below?
User	No
Agent	Here are some examples of the property ”Galaxy”
Agent	Does the property ”Galaxy” apply to the image below?
User	No
Agent	Here are some examples of the property ”Checkered”
Agent	Does the property ”Checkered” apply to the product you want?
User	No

Continued on next page

Table 7.2: A sample interaction from Amazon Mechanical Turk.

Agent	Here are some examples of the property ”Printed”
Agent	Does the property ”Printed” apply to the product you want?
User	No
Agent	Here are some examples of the property ”Shoe Accessories”
Agent	Does the property ”Shoe Accessories” apply to the image below?
User	No
Agent	Here are some examples of the property ”Satin”
Agent	Does the property ”Satin” apply to the image below?
User	No
Agent	Is this the image you were looking for?
User	Yes

Chapter 8

Clarification in Language Grounding Models Based on Joint Embeddings

Our prior work in chapters 4, 5 and 6 is motivated by enabling robots to robustly understand natural language descriptions of objects, and use clarifications and opportunistic active learning to improve or augment this ability. In those works, we have used binary or multiclass classifiers for perceptual concepts to perform grounding. However, if a system is performing lifelong learning, it can potentially be exposed to a very large number of concepts. In this situation, it is challenging to identify when different words should be grouped into a single concept, and it becomes computationally difficult to maintain classifiers for all concepts. In this chapter, we attempt to extend our prior work on clarifications (chapter 6) to a grounding model based on joint embeddings of language and images.

8.1 Motivation

For many grounded language understanding tasks such as understanding natural language descriptions of objects, a robot would need to be able to perceptually ground a large number of words used to denote properties of objects. When we require such a system to perform lifelong learning, we would ideally not want to fix the set of perceptual concepts to be learned ahead of time. We would prefer the system to identify these from interactions.

In our work so far, we have used binary or multiclass classifiers to perceptually ground such words. However, as the number of perceptual predicates to be learned increases, there are significant memory requirements for storing binary classifiers, or for performing operations in a sufficiently large multiclass classifier. Also, while the use of a multiclass classifier in chapter 6 allows the model to learn correlations between labels, it requires the set of labels, or at least its size, to be pre-defined. Further, both binary and multiclass classifiers need hard boundaries between different categories, to decide what constitues each class. When these are used as part of a grounding system, the system needs a mechanism to decide whether two different words, for example "*red*" and "*scarlet*", are to be treated as two separate concepts or a single one.

Also, there may be many correlations between such concepts. For example, "*red*" and "*scarlet*" are very similar colors, "*red*" and "*blue*" are both colors but less similar in meaning otherwise, "*red*" and "*apple*" are not similar in meaning but many apples are red. While the use of a multiclass classifier in chapter 6 allows the model to learn such correlations between labels, it still requires the model to make a decision about whether to treat "*red*" and "*scarlet*" as referring to the same or different classes. In our prior work, we have either used a fixed set of perceptual concepts to be learned (chapter 6) or used simple heuristics such as stemming to decide whether two words correspond to the same perceptual concept (chapter 5).

In this chapter, we attempt to extend our prior work on clarifications (chapter 6) to a grounding model based on joint embeddings of language and images.

Word embeddings have been shown to capture many useful similarity and relatedness properties (Mikolov et al., 2013), which provide an alternative method to capture label correlations. These are shown to be enhanced by training multimodal vectors (Lazaridou et al., 2015). There are also works on learning networks to score how well a natural language description or caption applies to an image (Hu et al., 2016; Xiao et al., 2017; Wang et al., 2016), some of which test the effectiveness of these methods for retrieving images based on captions. Such a model is also often used in downstream tasks where such grounding is implicitly required (Zhang et al., 2018a)

Further, once a joint embedding space is learned, a number of fast nearest neighbour algorithms (Johnson et al., 2019) can be used to efficiently find words that describe images, or vice versa, even as the number of words and images in question becomes large. This efficiency is desirable in systems that need to interact in real time with users.

Our original goal was to replicate the results in chapter 6 with a language grounding model that is more capable of adapting to an open ended vocabulary. However, we find that estimated information gain, as well as other statistics from the grounding model are not informative about whether a clarification query is likely to increase the belief of the correct target object.

8.2 Task Setup

We consider an interactive task for learning to ground natural language descriptions of objects, similar to that of chapter 6. Given a set of candidate images and a natural-language description, the goal of the system is to identify the image being referred to. Before trying to identify the target image, the system can ask attribute-based clarification questions that can better improve its success at identifying the correct image. In this work, we do not include active learning questions, although that would be desirable in the future. Also, in contrast to chapter 6, we do not maintain a fixed set of attributes for which perceptual classifiers are to be learned. More similar to chapter 5, the system builds a list of perceptual predicates from previous descriptions of images, and any of these can be queried in a clarification question.

Since we do not have active learning queries, we only have an active test set. The system is presented with the active test set and a description of the target object. The system can then optionally ask yes/no clarification questions about whether a word is applicable to the target object. Clarification questions are typically expected to use words not present in the description but are somehow related to the images in the active test set, as they are expected to reduce the search space for guessing.

The interaction ends when the system makes a guess of which object is the target, and is said to be successful if it correctly guesses the target objects. The goal of the task is to maximize the number of successful guesses, while also keeping dialogs as short as possible.

Similar to chapter 5, we simulate dialogs using the Visual Genome dataset (Krishna et al., 2017). More details about the dataset are included in sections 5.3 and 5.5.1. We use annotated descriptions of image regions to provide the target descriptions in simulated dialogs, and use the expanded list of words associated with a region (section 5.3) to answer clarification queries.

8.3 Methodology

8.3.1 Grounding Model

We use a grounding model which learns to project the images, and words from descriptions, represented as vectors, into a joint space in which projected images are close to projections of words that apply to them and vice versa.

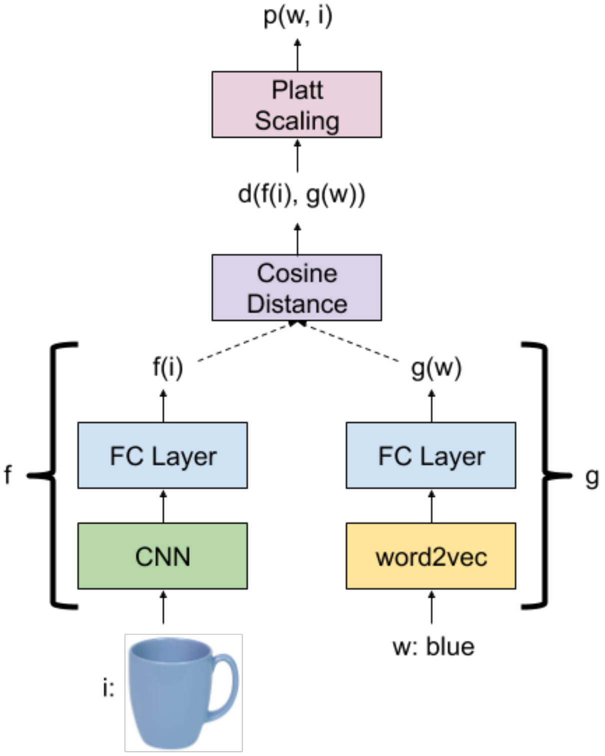


Figure 8.1: Grounding model that projects words and images to a joint space, and uses Platt scaling to convert the distances in the space to probabilities.

The model is illustrated in Figure 8.1. We extract feature representations for images from the penultimate layer of the VGG network (Simonyan and Zisserman, 2014), and encode words using word2vec (Mikolov et al., 2013). The vector representation of each modality is then projected using a corresponding fully connected layer to a common dimensionality.

Consider an image i with the associated description $\mathbf{w} = (w_1, w_2, \dots w_n)$. Let f and g be the learned projection functions for projecting images and words respectively into the joint space. Given another random word w', we would like to ensure that in the learned space, the distance from the projection of image i to that of the random word w' is greater than its distance to the projections of the words in the description, $w, \in, \mathbf{w}$. That is, given an appropriate distance function d in the learned space, we would like to ensure that:

$$d(f(i), g(w')) \geq d(f(i), g(w)) \; \forall \; w \in \mathbf{w} \tag{8.1}$$

Also, we would like to ensure for any word in the description, $w, \in, \mathbf{w}$, the distance between its projection in the learned space to that of any random image i' is greater than the distance to the image i that the words describe. That is,

$$d(f(i'), g(w)) \geq d(f(i), g(w)) \; \forall \; w \in \mathbf{w} \tag{8.2}$$

In this work, f and g project the words and images to the same dimension and we use cosine distance as the distance function. We can capture the above constraints using a ranking loss (Wang et al., 2016) as follows:

$$\begin{aligned}\mathcal{L}_r &= max(0, d(f(i), g(w)) - d(f(i), g(w')) + m) \\ &+ max(0, d(f(i), g(w)) - d(f(i'), g(w)) + m) \\ &+ \ldots \end{aligned} \tag{8.3}$$

where m is a margin. We initially train the grounding model using paired images and descriptions, and using this trained model for interactive retrieval. During training, given a positive example (i, w), we sample a random image as the negative example i' and use constraint 8.1. We also experimented with sampling words from descriptions of other random images to obtain a negative example that could be used with 8.2 but found that the final model performed better with only image-based negative examples.

It would be possible using distances in this joint space to identify the image that best matches a description, for example by choosing the image with minimum distance to words in the description. However, for identifying good clarification questions, it is helpful to calibrate the distances into probabilities. To do this, we use the following mechanism inspired by Platt scaling (Platt, 1999). Let $p(w, i)$ be the probability that w is applicable to i, which is estimated as follows:

$$p(w, i) = \sigma(-A(w) * d(f(i), g(w)) + B(w))$$

where

$$\begin{aligned} A(w) &= \sigma(\theta_A^T w) \quad (8.4) \\ B(w) &= ReLU(\theta_B^T w) \quad (8.5) \end{aligned}$$

To train parameters θ_A and θ_B we add an additional log loss term:

$$\begin{aligned} \mathcal{L}_l &= \sum_w \sum_i - \{(y(w,i) \log(p(w,i)) \\ &+ (1 - y(w,i)) \log(1 - p(w,i)))\} \end{aligned} \quad (8.6)$$

where $y(w,i) = 1$ if word w applies to image i and $y(w,i) = 0$ if word w does not apply to image i. The total loss is then a linear combination of the two loss terms:

$$\mathcal{L} = \alpha \mathcal{L}_l + \mathcal{L}_r \quad (8.7)$$

where the weight α is tuned as a hyperparameter. Our best results used $\alpha = 10$.

For grounding, given images $\mathbf{I} = \{i_1, i_2 \ldots i_m\}$, and a description $\mathbf{W} = (w_1, w_2, \ldots w_n)$. Let the clarifications obtained during the dialog be

$$\mathbf{C} = \{(q_1, a_1), (q_2, a_2) \ldots (q_k, a_k)\}$$

where $q_1, q_2 \ldots q_k$ are words and $a_1, a_2, \ldots a_k \in \{0, 1\}$. Then, the set of words that are known to apply to the target image are $\mathbf{W}^+ = \{w_1, w_2, \ldots w_n\} \cup \{q_i : a_i = 1, (q_i, a_i) \in \mathbf{C}\}$ and the set of words known *not* to apply to the image are

$\mathbf{W}^- = \{q_i : a_i = 0, (q_i, a_i) \in \mathbf{C}\}$ Then, for each image, we can calculate the belief that it is the target as follows:

$$b(i) = \frac{1}{Z}\Pi_{w\in\mathbf{W}^+}p(w,i)\Pi_{w\in\mathbf{W}^-}(1 - p(w,i)) \tag{8.8}$$

where Z is a normalization factor. When guessing, we choose the image with maximum belief $b(i)$.

8.3.2 Identifying and Scoring Clarifications

A clarification can also involve any word from the vocabulary. As candidates, we first identify the set of words closer to at least one candidate image than a distance threshold. We explore the following ways of fixing the distance threshold:

- Max-in-context threshold – Closest distance between any candidate image and any word in the current description – This indicates that the word is very likely to apply more strongly to that image than any word in the description to the target.
- Average Threshold – Average distance between images and words in their descriptions, as observed during pretraining of the grounding model. This gives a rough estimate of how close a word is on average to an image it applies to.

We then estimate the information gain J of each candidate q as follows, similar to chapter 6

$$J(q) = \sum_{i} \sum_{a \in \{0,1\}} b(i) P(a|q,i) log \left(\frac{P(a|q,i)}{P(a|q)} \right)$$

where $P(1|q,i) = p(q,i)$ and $P(0|q,i) = 1 - p(q,i)$ and $P(a|q) = \sum_i b(i)P(q,i)$.

8.4 Experimental Setup

We simulate dialogs using the Visual Genome dataset (Krishna et al., 2017), as in chapter 5. More details about the dataset are included in sections 5.3 and 5.5.1. However, we modify the data split because we need a pretrained grounding model to be able to generate useful clarification questions even in the early stages of policy learning. From the work in chapter 5, we obtain a list of words relevant to at least 1000 regions each. We divide these randomly into 4 splits which we call *policy_pretrain*, *policy_train*, *policy_val* and *policy_test*. Regions containing words in the *policy_test* set form the set of *policy_test* regions. Of the remaining regions, those containing words in the *policy_val* set form the set of *policy_val* regions. Of the remaining regions, those containing words in the *policy_train* set form the set of *policy_train* regions. The remaining regions form the *policy_pretrain* set of regions. This iterative procedure ensures that regions in each of the *policy_train*, *policy_val* and *policy_test* sets contain related words not seen in the previous stages. Each of these is then split into subsets *classifier_training* and *classifier_test* by a uniform 60-40 split.

The set of *policy_pretrain* regions is used to train the grounding model. We use its *classifier_training* subset of regions for training and its *classifier_test* subset to tune hyperparameters of the grounding model. We intended to use the remaining splits for dialog policy learning and evaluation.

We train the grounding model using paired regions and descriptions from regions in the *classifier_training* subset of the *policy_pretrain* set of regions. For negative examples, we randomly sample other images in the subset. We then evaluate the usefulness of clarifications on the *policy_train* set of regions as follows.

We sample 1000 dialog contexts to evaluate the usefulness of clarifications. We compare two types of dialog setups -

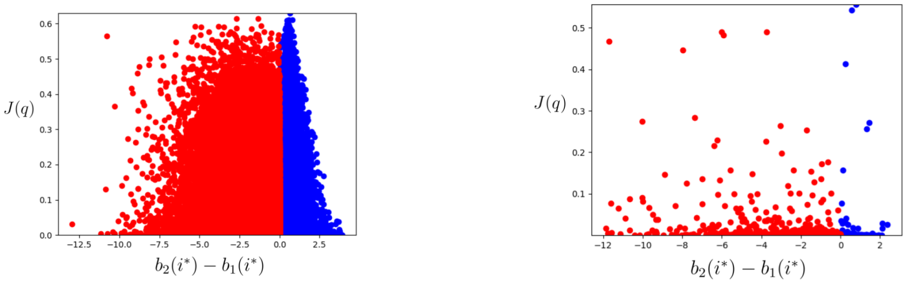
- Random candidate images - Each dialog context has an active test set consisting of 20 regions sampled from the *classifier_test* of the *policy_train* set. One of these is randomly selected to be the target region.
- Nearest neighbor candidate images - We sample target regions uniformly at random from the *classifier_test* of the *policy_train* set, and provide as the active test set the target image with its 19 nearest neighbors.

We provide the system with the annotated caption of the chosen region as an initial description and obtain an initial belief $b_1(i)$, according to equation 8.8, for each image i in the active test set. Then, for each type of threshold in section 8.3.2, we obtain all words within the threshold, and obtain a new belief $b_2(i)$, for all candidate images, after using this word q as a clarification. For a clarification query q to be useful, asking the query needs to increase the belief in the correct target image. For our estimate of information gain, $J(q)$, to be a useful metric to identify good clarification questions, a high value of $J(q)$ should correspond to an increase in the belief of the target image, that is $b_2(i^*) - b_1(i^*) > 0$.

To determine whether our metric is useful, we plot the difference in belief of the target image i^*, $b_2(i^*) - b_1(i^*)$ on the x-axis, against the information gain of the query $J(q)$ on the y-axis. We use different colors to separate useful queries q that increase the belief of the target image, that is, $b_2(i^*) - b_1(i^*) > 0$, and those that are not useful and decrease the belief of the target image, that is, $b_2(i^*) - b_1(i^*) < 0$. Ideally we would like to identify an information gain threshold j^* such that for queried words q where $J(q) > j^*$, the belief of the target image increases, that is $b_2(i^*) - b_1(i^*) > 0$. That is, we would like to observe a separation between the two groups of queries along the y-axis. We wish to use the plots to verify whether such a separation exists.

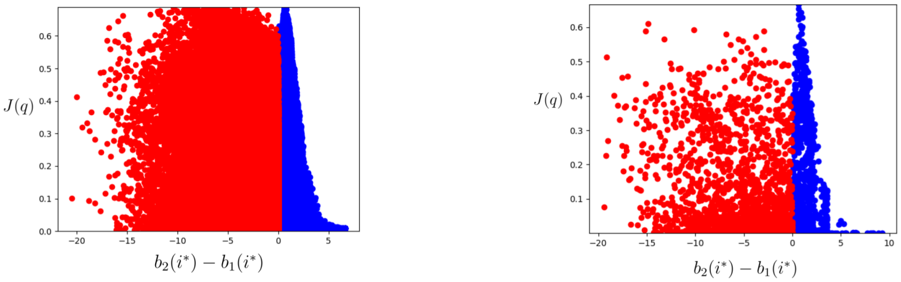
8.5 Results and Discussion

The results are presented in figure 8.2. Queries that are useful are plotted in blue, and those that are not useful are plotted in red. We observe that for the most part, the two groups of queries are not separated along the y-axis. This suggests that our estimate of information gain is not a good indicator of whether a clarification query is likely to improve the belief of the correct target image. For the combination of random candidate images and the average threshold, it does appear that at higher information gain, there are fewer clarification queries that will decrease the belief of the correct target image. However, when we compared retrieval success, with and without clarification queries chosen in this condition to maximize information gain, we did not obtain an improvement in overall retrieval success.



(a) Nearest neighbor candidate images
Max-in-context threshold

(b) Nearest neighbor candidate images
Average threshold



(c) Random candidate images
Max-in-context threshold

(d) Random candidate images
Average threshold

Figure 8.2: Information gain vs difference in belief of target image $b_2(i^*) - b_1(i^*)$. Words which raise the belief of the target image are plotted in blue and others are plotted red.

We have a couple of hypotheses about why clarification questions based on information gain do not necessarily increase the belief of the correct target image -

- We use a list of manually annotated objects and attributes to answer clarification queries. We assume that any word not present in an expanded version of this list is not relevant to the image region. However, it is very possible that such lists are incomplete, and our heuristics for expanding the list do not capture all possible synonyms or other words closely related in meaning. Thus it is possible that especially when we give an answer of “no” to a clarification query, it may be wrong.

- Although we treat the values $p(i, w)$ from our grounding model as probabilities for the purposes of calculating information gain, these are not necessarily calibrated, and model is not constrained to produce values that correspond to a true probability distribution. It is only constrained to ensure that each individual value is between 0 and 1. However, we would additionally need $\sum_i \sum_w p(i, w)$ to be 1 for this to be a true probability distribution, which is difficult to implement in models such as joint embeddings as the summation would be intractable.

8.6 Summary

In this chapter, we attempt to extend the clarification mechanism introduced in chapter 6 to a joint embedding based grounding model, that can more naturally adapt to an open vocabulary. This required us to design a method to convert distances obtained from such a learned space into calibrated probabilities to estimate information gain. However, we find our estimate of information gain of clarification queries is not a good indicator of whether these clarification queries can increase the success rate of retrieving images based on natural language descriptions, and further work is needed to find an alternative. We believe that the problem of estimating information gain for clarifications in models other than those based on classification is an interesting direction for future work as such models show increasingly better performance in language grounding tasks.

Chapter 9

Future Work

In this thesis, we discuss ways in which dialog interaction with users can be used to enable lifelong learning in grounded natural language understanding systems, motivated by service robot applications. In this chapter we discuss future directions that can help improve such human-robot dialog systems, and to make lifelong learning in such systems more practically useful. A subset of these have been presented in our position paper, Padmakumar and Mooney (2020a).

9.1 Better Language Grounding Models

In most of our work (chapters 4, 5, 6), we have used fairly simple models of natural language understanding that are not designed to handle multi-word expressions or more complex compositionality such as part-of relations and negation. In recent years, context sensitive word and sentence embeddings pretrained on very large amounts of text have been shown to be more effective in a number of tasks including question answering, textual entailment, semantic role labeling, coreference resolution, named entity extraction, and sentiment analysis (Peters et al., 2018; Devlin et al., 2019; Liu et al., 2019). There have been shown to be useful for natural language understanding in dialog systems (Chen et al., 2019b), and can probably be used to predict object attributes more effectively than the heuristics used in our work.

More recently, multimodal transformers pretrained on large datsets of paired images and captions have been shown to perform well on a number of language and vision tasks, including retrieving images based on natural language descriptions (Lu et al., 2019; Tan and Bansal, 2019). The success of these models goes a long way to make them more practically useful, but can be a challenge when developing active learning methods because it is difficult to see a performance improvement on them from a few examples. Further work is needed to extend our work on clarification and opportunistic active learning to such models.

9.2 Multimodal Language Grounding

Robots can sense the world through modalities other than vision, for example sound through a microphone or by manipulating objects with an arm. Prior work has shown that when people are allowed to handle objects before describing them, they may use non-visual predicates such as *heavy* or *rattling* to describe them. Using multimodal features has been shown to allow a robot to better ground object descriptions, and are essential when non-visual predicates are used (Thomason et al., 2016).

Deep learning has been very effective in visual classification (Simonyan and Zisserman, 2014), and even some audio classification tasks (Hershey et al., 2017) using large labelled datasets. However, it is difficult to obtain the same improvements with other types of data that requires physical manipulation of objects. Such manipulation is time consuming, and requires a lot of robots to parallelize. It is also difficult to share data between different types of robots due to differences in available sensors. Developing ensembling methods to effectively combine deep learning based visual classifiers with more sample efficient classifiers for other modalities such as haptic data is likely to improve the performance of grounded language understanding systems on physical robots.

Another type of multimodality available to dialog systems on embodied robots is gesture and gaze. Humans often use gesture and gaze to direct attention and using this information can potentially reduce ambiguity in natural language understanding (Whitney et al., 2017). Robots can also make use of gaze and gestures to facilitate turn taking and reduce ambiguity (Kontogiorgos et al., 2019). This can be used both as an alternative mechanism to ask clarification questions and also to make the dialog more natural.

9.3 Active Learning in Practice

In our work on opportunistic active learning (chapters 4, 5, 6), we set up our test phase so that new concepts are presented which require active learning for the model to adapt to them. However, in real world scenarios, the model is likely to be trained on a reasonable amount of annotated data, and the system has to trade off model improvement with a mix of tasks that may or may not require model improvement. In this scenario, it is more difficult to demonstrate the benefits of active learning.

Further, our work so far, along with most work on active learning, has been restricted to binary or multiclass classifiers. However, for language grounding in particular, and many other applications, other types of models such as joint embeddings or multimodal transformers may be preferable. Further work needs to be done to identify appropriate active learning queries for such models.

Recent work has also highlighted some other limitations of active learning in real-world settings – with benefits not generalizing reliably across models and tasks, changing deployed models, as well as the model size, training data requirements and stochasticity of deep learning models (Lowell et al., 2019; Koshorek et al., 2019). This suggests that new types of active learning methods may be required with deep learning models.

User frustration is another potential concern for frameworks such as opportunistic active learning. This restricts the number of queries that the system may ask per dialog session, which may require active learning methods to be combined with other semi-supervised learning techniques to scale up to the data requirements of deep learning models. The spread of queries across dialogs in batches can also be improved using extensions of batch-mode active learning techniques (Brinker, 2003; Guo and Schuurmans, 2008) to deep learning methods (Ash et al., 2020). In order to reduce user frustration due to active learning questions, another possibility would be to look into methods to implicitly embed active learning queries into system responses. For example, an interactive search and retrieval system that allows a user to refine search results can combine a mix of search results known to be relevant, with one or two results that it is uncertain about. Whether or not the user selects these can provide a weak, noisy label.

In dialog systems where the user interacts via speech rather than text, additional cues can be used to decide when active learning queries are potentially inappropriate. For example, prosodic cues can be used to identify whether users are stressed or frustrated (Devillers et al., 2003), and the dialog policy can be designed to avoid active learning questions when such reactions are detected. Other sources of input such as facial expressions (Schuller et al., 2011) or gestures can also be used to improve such predictions (Busso et al., 2008). Prosody may also be useful for detecting sarcasm (Tepperman et al., 2006; Rakov and Rosenberg, 2013) or other forms of misuse or intentional wrong answers from users, to avoid corrupting the collected labelled data.

Depending on the types of active-learning questions involved, the systems may also need to demonstrate some sort of few-shot learning to keep users interested in assisting the learning process. For example, children typically can learn to identify colors or common objects such as fruits with very few examples. For such tasks, users are likely to expect a similar rate of learning from the system. In contrast, more error may be tolerated in a system learning something less tangible such as a person's food preferences for the purpose of recommending restaurants.

9.4 Sim2Real Issues in Dialog Systems for Lifelong Learning

Simulation is an important component of research in dialog systems (Schatzmann et al., 2006; Gür et al., 2018) as well as robotics (Eppner et al., 2019). In particular when using data hungry methods such as deep learning and reinforcement learning, it is desirable to train systems in simulation as it is time consuming and expensive to obtain enough training data in the real world.

Simulation in dialog systems is commonly used for dialog policy learning (Gür et al., 2018) but also for training end to end dialog systems (Wen et al., 2017). Simulations for training end to end dialog systems are typically built using a corpus of human-human dialogs (Budzianowski et al., 2018; De Vries et al., 2017; Thomason et al., 2019). It is challenging to extend this for dialogs where the system needs to perform active learning because if the concepts the system is trying to learn are well know to human users, it is difficult for humans to ask questions that would be good active learning questions (Yu et al., 2017b). Some work solves this problem by replacing the words denoting these concepts with words in a synthetic language (Yu et al., 2017b). However this idea is not easily adapted if the active learning queries need to obtain per-example labels for a concept, for example, asking whether an image contains a person or not. Other works make use of additional information available in the dataset, such as annotated attributes of objects in our case (chapters 5, 6 and 8) or a known navigation graph (Chi et al., 2020; Nguyen and III, 2019) that can potentially provide answers to any queries the system asks. However, as we observed in chapter 7, these simulations do not necessarily accurately reflect the behaviour of human users interact with the dialog system, and hence may not transfer well.

Simulation environments that require extensive extra annotation can also be expensive to build, especially if these have to be task specific. If the dialog system has to make use of an existing annotated dataset, this restricts the set of information gathering actions to those that can be answered by the available annotations.

This challenge becomes more significant for dialog systems that are intended to be a part of an embodied system. Most current work on dialog for embodied systems is done entirely in simulation (Thomason et al., 2019; Nguyen and III, 2019), particularly if the work involves the learning of dialog policies. This is also true of our work (chapters 5 and 6). In addition to the difficulty of designing a simulation environment, further experiments are needed to evaluate whether existing systems perform comparably when implemented on real robots. In this case, besides differences in interaction, there is also the possibility of increased sensing errors or improper execution of actions by the robot to be handled. Additionally, as in chapter 7, further work in human robot interaction may be required to ensure that the interaction is natural and not confusing to users. It is likely that additional challenge problems are likely to be identified as more attempts are made to transfer dialog systems from simulated to real physical environments.

Further, most current lifelong learning systems are specific to the task and data involved (Chen and Liu, 2018). More work is needed to identify general purpose models or techniques that can be used to enable lifelong learning across tasks and data types.

9.5 Learning From Demonstration With Dialog

Learning from demonstration is a paradigm in which an agent learns a policy to perform a task from example trajectories, or demonstrations, instead of through experience and numerical rewards (Argall et al., 2009). Demonstrations can be obtained in a variety of ways, for example having an expert teleoperate a robot to perform a task, or having the robot watch a person perform the task, or watch a video of the task being performed. Learning from demonstration is important for general purpose robots as they may need to perform tasks they have not been specifically trained to do, and it is easier even for expert users to provide a demonstration of the task than to program it.

Most existing work on learning from demonstration is not interactive. However, some methods assume that an expert is watching the robot as it learns and can provide expert guidance on what action to take when required, to handle unforeseen scenarios (Ross et al., 2011). A simple way to augment demonstrations when they are done in an interactive setup, would be for the human to provide feedback as the robot performs actions, which can then be used for reinforcement learning (Knox and Stone, 2008; Warnell et al., 2018).

Another way to augment demonstrations would be with simultaneous natural language instructions that highlight which parts of a demonstration are salient (Goyal et al., 2019). For example, when demonstrating how to set a table, the human demonstrator can say that they are placing a spoon to the right of a plate, so that the agent knows that this is the salient information, not the distance of the spoon from the plate. A language interaction accompanying a demonstration can also allow the agent to ask clarification questions. For example, after the above demonstration, the robot may ask what to do if there is another object already present to the right of the plate. Also, a robot may request for a demonstration during an interaction. For example, if the person asks the robot to set the table, the robot may ask to see a demonstration of this if it does not know how to perform the task, or may ask for a demonstration of a specific step such as how to handle a vase which is not normally present.

Chapter 10

Conclusions

In this thesis, we present work that is aimed at designing dialog systems that ground language in perception and leverage interactions with users to perform lifelong learning. This research combines ideas and techniques from natural language processing, computer vision, robotics, dialog systems, reinforcement learning and active learning. The following is a brief summary of the individual contributions included:

- We combine existing work on learning a dialog policy from dialog interactions, and improving a semantic parser using weak supervision obtained by leveraging the structure of clarification dialogs. We demonstrate that simultaneous improvement of the dialog policy and semantic parser is beneficial compared to improving either component alone, and that updates from the semantic parser need to be seen by the dialog policy through interactions to improve overall dialog performance (Chapter 3).
- We propose the framework of opportunistic active learning to integrate active learning questions at test time to enable a robot to identify new perceptual concepts and build classifiers for them during operation. We demonstrate that a robot performing opportunistic active learning is more successful than a baseline agent at using knowledge obtained from test-time queries at improving its ability to retrieve objects based on natural language descriptions,

and that people find such a robot more fun and usable (Chapter 4).

- We demonstrate how to formulate an opportunistic active learning problem as a reinforcement learning problem, and learn a policy that can effectively trade-off opportunistic active learning queries against task completion. We evaluated this approach on the task of grounded object retrieval from natural language descriptions and learn a policy that retrieves the correct object in a larger fraction of dialogs than a previously proposed static baseline, while also lowering average dialog length (Chapter 5).
- We train a hierarchical dialog policy to jointly perform *both* clarification and active learning in the context of an interactive language-based image retrieval task, and demonstrate that jointly learning dialog policies for clarification and active learning is more effective than the use of static dialog policies for one or both of these functions (Chapter 6).
- We compare the learned and static dialog policies from chapter 6 in interactions with human users, and find that the learned policy transfers modestly better than the static policy. We also uncover issues related to the differences between simulated and real interactions, as well as the importance of a good user interface design for dialog systems.
- We attempt to extend the clarification mechanism introduced in chapter 6 to a joint embedding based grounding model, that can more naturally adapt to an open vocabulary. However, we find our estimate of information gain of clarification queries is not a good indicator of whether these clarification queries can increase the success rate of retrieving images based on natural

language descriptions (Chapter 8).

In our work, we focus on how clarification and active learning questions can be used to improve natural language understanding and enable a robot to perform lifelong learning to improve its ability to understand and ground language in perception. We believe that lifelong learning has the potential to be extremely beneficial in service robot applications, and dialog systems provide a good mechanism to obtain labelled training data necessary to perform lifelong learning. However, considerable improvement is still necessary to realize these benefits. We outline some possible future directions for such improvement in chapter 9.

References

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and Top-down Attention for Image Captioning and Visual Question Answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 6, 2018.

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A Survey of Robot Learning from Demonstration. *Robotics and Autonomous Systems (RAS)*, 57(5):469–483, 2009.

Yoav Artzi and Luke Zettlemoyer. Bootstrapping Semantic Parsers from Conversations. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 421–432, 2011.

Yoav Artzi and Luke Zettlemoyer. Weakly Supervised Learning of Semantic Parsers for Mapping Instructions to Actions. *Transactions of the Association for Computational Linguistics (TACL)*, 1(1):49–62, 2013.

Dilip Arumugam, Siddharth Karamcheti, Nakul Gopalan, Edward C Williams, Mina Rhee, Lawson LS Wong, and Stefanie Tellex. Grounding Natural Language Instructions to Semantic Goal Representations for Abstraction and Generalization. *Autonomous Robots*, 2018.

Jordan T Ash, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

Philip Bachman, Alessandro Sordoni, and Adam Trischler. Learning Algorithms for Active Learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pages 301–310, Sydney, Australia, 06–11 Aug 2017. PMLR.

Michele Banko and Oren Etzioni. Strategies for Lifelong Knowledge Extraction from the Web. In *Proceedings of the 4th International Conference on Knowledge Capture*, 2007.

Emanuele Bastianelli, Danilo Croce, Andrea Vanzo, Roberto Basili, and Daniele Nardi. A Discriminative Approach to Grounded Spoken Language Understanding in Interactive Robotics. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2747–2753. AAAI Press, 2016.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic Parsing on Freebase from Question-Answer Pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013.

Indrani Bhattacharya, Arkabandhu Chowdhury, and Vikas C Raykar. Multimodal Dialog for Browsing Large Visual Catalogs Using Exploration-Exploitation Paradigm in a Joint Embedding Space. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, pages 187–191, 2019.

Yonatan Bisk, Deniz Yuret, and Daniel Marcu. Natural Language Communication with Robots. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 751–761, 2016.

Valts Blukis, Dipendra Misra, Ross A. Knepper, and Yoav Artzi. Mapping Navigation Instructions to Continuous Control Actions with Position Visitation Prediction. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2018.

Antoine Bordes and Jason Weston. Learning end-to-end goal-oriented dialog. *Computing Research Repository*, arXiv:1605.07683, 2016.

Klaus Brinker. Incorporating Diversity in Active Learning with Support Vector Machines. In *Proceedings of the 20th International Conference on Machine Learning (ICML 2003)*, 2003.

Klaus Brinker. On Active Learning in Multi-label Classification. In *From Data and Information Analysis to Knowledge Engineering*, pages 206–213. Springer-Verlag, 2006.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Andrea Burns, Reuben Tan, Kate Saenko, Stan Sclaroff, and Bryan A Plummer. Language Features Matter: Effective Language Representations for Vision-Language Tasks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 7474–7483, 2019.

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N Chang, Sungbok Lee, and Shrikanth S Narayanan. IEMOCAP: Interactive Emotional Dyadic Motion Capture Database. *Language Resources and Evaluation*, 42(4), 2008.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka, and Tom M Mitchell. Toward an Architecture for Never-Ending Language Learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2010.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Alessandra Cervone, Giuliano Tortoreto, Stefano Mezza, Enrico Gambi, Giuseppe Riccardi, et al. Roving Mind: A Balancing Act Between Open–domain and Engaging Dialogue Systems. In *Alexa Prize Proceedings*, 2017.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from RGB-D Data in Indoor Environments. In *3D Vision*, 2017.

Shobhit Chaurasia and Raymond J. Mooney. Dialog for Language to Code. In *Proceedings of the 8th International Joint Conference on Natural Language Processing (IJCNLP-17)*, pages 175–180, Taipei, Taiwan, November 2017.

Zhiyuan Chen and Bing Liu. Lifelong Machine Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018.

David L Chen and Raymond J Mooney. Learning to Interpret Natural Language Navigation Instructions from Observations. In *Proceedings of the Twenty Fifth AAAI Conference on Artificial Intelligence (AAAI)*, 2011.

Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. NEIL: Extracting Visual Knowledge from Web Data. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2013.

Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. Touchdown: Natural Language Navigation and Spatial Reasoning in Visual Street Environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12538–12547, 2019.

Qian Chen, Zhu Zhuo, and Wen Wang. BERT for Joint Intent Classification and Slot Filling. *Computing Research Repository*, 1902.10909, 2019.

Ta-Chung Chi, Mihail Eric, Seokhwan Kim, Minmin Shen, and Dilek Hakkanitur. Just Ask: An Interactive Learning Framework for Vision and Language Navigation. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. Visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 326 – 335, 2017.

Abhishek Das, Satwik Kottur, José MF Moura, Stefan Lee, and Dhruv Batra. Learning Cooperative Visual Dialog Agents with Deep Reinforcement Learning. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2970–2979. IEEE, 2017.

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied Question Answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Lucie Daubigney, Matthieu Geist, Senthilkumar Chandramohan, and Olivier Pietquin. A Comprehensive Reinforcement Learning Framework for Dialogue Management Optimization. *Journal of Selected Topics in Signal Processing*, 6(8):891–902, 2012.

Harm De Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron Courville. GuessWhat?! Visual Object Discovery Through Multi-Modal Dialogue. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Laurence Devillers, Ioana Vasilescu, and Catherine Mathon. Prosodic Cues for Perceptual Emotion Detection in Task-Oriented Human-Human Corpus. In *Proceedings of the 15th International Congress of Phonetic Sciences (ICPhS)*, 2003.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2019.

Haris Dindo and Daniele Zambuto. A Probabilistic Approach to Learning a Visually Grounded Language Model through Human-Robot Interaction. In *International Conference on Intelligent Robots and Systems*, pages 760–796, Taipei, Taiwan, 2010. IEEE.

Layla El Asri, Romain Laroche, and Olivier Pietquin. Compact and Interpretable Dialogue State Representation with Genetic Sparse Distributed Memory. In *Proceedings of the 7th International Workshop on Spoken Dialogue Systems (IWSDS)*, 2016.

Yaakov Engel, Shie Mannor, and Ron Meir. Reinforcement Learning with Gaussian Processes. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 201–208, 2005.

Clemens Eppner, Arsalan Mousavian, and Dieter Fox. A Billion Ways to Grasp: An Evaluation of Grasp Sampling Schemes on a Dense, Physics-based Grasp Data Set. *Computing Research Repository*, arXiv:1912.05604, 2019.

Rui Fang, Changsong Liu, Lanbo She, and Joyce Y Chai. Towards Situated Dialogue: Revisiting Referring Expression Generation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 392–402, 2013.

Rui Fang, Malcolm Doering, and Joyce Y Chai. Collaborative Models for Referring Expression Generation in Situated Dialogue. In *Proceedings of the Twenty Eighth AAAI Conference on Artificial Intelligence (AAAI)*, pages 1544–1550, 2014.

Meng Fang, Yuan Li, and Trevor Cohn. Learning How to Active Learn: A Deep Reinforcement Learning Approach. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. ACL, 2017.

Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing Objects by their Attributes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1778–1785. IEEE, 2009.

Geli Fei, Shuai Wang, and Bing Liu. Learning Cumulatively to Become More Knowledgeable. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1565–1574, 2016.

Fangxiang Feng, Xiaojie Wang, Ruifan Li, and Ibrar Ahmad. Correspondence Autoencoders for Cross-Modal Retrieval. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 12(1s):26, 2015.

Maxwell Forbes and Yejin Choi. Verb Physics: Relative Physical Knowledge of Actions and Objects. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 266–276, 2017.

Qiaozi Gao, Malcolm Doering, Shaohua Yang, and Joyce Chai. Physical Causality of Action Verbs in Grounded Language Understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 1814–1824, 2016.

Qiaozi Gao, Shaohua Yang, Joyce Chai, and Lucy Vanderwende. What action causes this? Towards Naive Physical Action-Effect Prediction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 934–945, 2018.

Milica Gašić and Steve Young. Gaussian Processes for POMDP-Based Dialogue Manager Optimization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):28–40, 2014.

Matthieu Geist and Olivier Pietquin. Kalman Temporal Differences. *Journal of Artificial Intelligence Research (JAIR)*, 39(1):483–532, 2010.

Prasoon Goyal, Scott Niekum, and Raymond J. Mooney. Using Natural Language for Reward Shaping in Reinforcement Learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

Sergio Guadarrama, Erik Rodner, Kate Saenko, Ning Zhang, Ryan Farrell, Jeff Donahue, and Trevor Darrell. Open-Vocabulary Object Retrieval. *Robotics: Science and Systems (RSS)*, 2(5):6, 2014.

Sergio Guadarrama, Erik Rodner, Kate Saenko, and Trevor Darrell. Understanding Object Descriptions in Robotics by Open-Vocabulary Object Retrieval and Detection. *The International Journal of Robotics Research (IJRR)*, 35(1-3):265–280, 2016.

Yuhong Guo and Dale Schuurmans. Discriminative Batch Mode Active Learning. In *Advances in Neural Information Processing Systems*, 2008.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On Calibration of Modern Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, page 1321–1330. JMLR.org, 2017.

Xiaoxiao Guo, Hui Wu, Yu Cheng, Steven Rennie, Gerald Tesauro, and Rogerio Feris. Dialog-based Interactive Image Retrieval. In *Advances in Neural Information Processing Systems*, pages 678–688, 2018.

Sheng Guo, Weilin Huang, Xiao Zhang, Prasanna Srikhanta, Yin Cui, Yuan Li, Hartwig Adam, Matthew R Scott, and Serge Belongie. The iMaterialist Fashion Attribute Dataset. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

Izzeddin Gür, Dilek Hakkani-Tür, Gokhan Tür, and Pararth Shah. User Modeling for Task oriented Dialogues. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 900–906. IEEE, 2018.

Stevan Harnad. The Symbol Grounding Problem. *Physica D: Nonlinear Phenomena*, 42(1-3):335–346, 1990.

David Harwath, Adria Recasens, Dídac Surís, Galen Chuang, Antonio Torralba, and James Glass. Jointly Discovering Visual Objects and Spoken Words from Raw Sensory Input. In *Proceedings of the European conference on computer vision (ECCV)*, pages 649–665, 2018.

Hado V. Hasselt. Double Q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621, 2010.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. The ATIS Spoken Language Systems Pilot Corpus. In *Proceedings of the Workshop on Speech and Natural Language*, HLT '90, pages 96–101, Stroudsburg, PA, USA, 1990. Association for Computational Linguistics.

Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. CNN Architectures for Large-Scale Audio Classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 131–135. IEEE, 2017.

Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. Natural Language Object Retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4555–4564, 2016.

Ronghang Hu, Marcus Rohrbach, Jacob Andreas, Trevor Darrell, and Kate Saenko. Modeling Relationships in Referential Expressions with Compositional Modular Networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 4418–4427. IEEE, 2017.

Huang Hu, Xianchao Wu, Bingfeng Luo, Chongyang Tao, Can Xu, Wei Wu, and Zhan Chen. Playing 20 Question Game with Policy-Based Reinforcement Learning. In *Proccedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Ronghang Hu, Jacob Andreas, Trevor Darrell, and Kate Saenko. Explainable Neural Computation Via Stack Neural Module Networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 53–69, 2018.

J. Johnson, M. Douze, and H. Jégou. Billion-scale Similarity Search with GPUs. *IEEE Transactions on Big Data*, pages 1–1, 2019.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artificial intelligence*, 1998.

Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. Globally Coherent Text Generation with Neural Checklist Models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 329–339, 2016.

W Bradley Knox and Peter Stone. TAMER: Training an Agent Manually via Evaluative Reinforcement. In *2008 7th IEEE International Conference on Development and Learning*, pages 292–297. IEEE, 2008.

Thomas Kollar, Jayant Krishnamurthy, and Grant Strimel. Toward Interactive Grounded Language Acquisition. In *Robotics: Science and Systems (RSS)*, 2013.

Thomas Kollar, Vittorio Perera, Daniele Nardi, and Manuela Veloso. Learning Environmental Knowledge from Task-Based Human-Robot Dialog. In *Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4304–4309, 2013.

Dimosthenis Kontogiorgos, Gabriel Skantze, André Tiago Abelho Pereira, and Joakim Gustafson. The Effects of Embodiment and Social Eye-Gaze in Conversational Agents. In *41st Annual Meeting of the Cognitive Science (CogSci), Montreal July 24th–Saturday July 27th, 2019*, 2019.

Omri Koshorek, Gabriel Stanovsky, Yichu Zhou, Vivek Srikumar, and Jonathan Berant. On the Limits of Learning to Actively Learn Semantic Representations. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, 2019.

Satwik Kottur, Ramakrishna Vedantam, José MF Moura, and Devi Parikh. Visual word2vec (vis-w2v): Learning Visually Grounded Word Embeddings Using Abstract Scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4985–4994, 2016.

Adriana Kovashka, Devi Parikh, and Kristen Grauman. WhittleSearch: Image Search with Relative Attribute Feedback. In *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2973–2980. IEEE, 2012.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations. *International Journal of Computer Vision*, 123(1):32–73, feb 2017.

Johannes Kulick, Marc Toussaint, Tobias Lang, and Manuel Lopes. Active Learning for Teaching a Robot Grounded Relational Symbols. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJACI)*, pages 1451–1457. AAAI Press, 2013.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. Scaling Semantic Parsers with On-the-fly Ontology Matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013.

Stanislao Lauria, Guido Bugmann, Theocharis Kyriacou, and Ewan Klein. Mobile Robot Programming Using Natural Language. *Robotics and Autonomous Systems (RAS)*, 38(3-4):171–181, 2002.

Angeliki Lazaridou, Elia Bruni, and Marco Baroni. Is this a Wampimuk? Cross-Modal Mapping Between Distributional Semantics and the Visual World. In

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL), volume 1, pages 1403–1414, 2014.

Angeliki Lazaridou, Marco Baroni, et al. Combining Language and Vision with a Multimodal Skip-gram Model. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 153–163, 2015.

Sang-Woo Lee, Yu-Jung Heo, and Byoung-Tak Zhang. Answerer in Questioner's Mind for Goal-oriented Visual Dialogue. In *Visually-Grounded Interaction and Language Workshop, NeurIPS*, 2018.

David D. Lewis and William A. Gale. A Sequential Algorithm for Training Text Classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval SIGIR '94*, pages 3–12. Springer London, 1994.

Xin Li and Yuhong Guo. Active Learning with Multi-Label SVM Classification. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pages 1479–1485, 2013.

Xachan Li, Lei Wang, and Eric Sang. Multi-label SVM Active Learning for Image Classification. In *2004 International Conference on Image Processing, 2004. ICIP'04*, volume 4, pages 2207–2210. IEEE, 2004.

Yining Li, Chen Huang, Xiaoou Tang, and Chen Change Loy. Learning to Disambiguate by Asking Discriminative Questions. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3419–3428, 2017.

Changsong Liu, Rui Fang, Lanbo She, and Joyce Chai. Modeling Collaborative Referring for Situated Referential Grounding. In *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 78–86, 2013.

Changsong Liu, Shaohua Yang, Sari Saba-Sadiya, Nishant Shukla, Yunzhong He, Song-Chun Zhu, and Joyce Chai. Jointly Learning Grounded Task Structures from Language Instruction and Visual Demonstration. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1482–1492, 2016.

Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations. In

Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pages 1096–1104, 2016.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *Computing Research Repository*, 1907.11692, 2019.

David Lowell, Zachary C. Lipton, and Byron C. Wallace. Practical Obstacles to Deploying Active Learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks. In *Advances in Neural Information Processing Systems*, pages 13–23, 2019.

Corey Lynch and Pierre Sermanet. Grounding Language in Play. *Computing Research Repository*, arXiv:2005.07648, 2020.

Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. Self-Monitoring Navigation Agent via Auxiliary Progress Estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox. Learning to Parse Natural Language Commands to a Robot Control System. In *Experimental Robotics*, pages 403–415. Springer, 2013.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. Listen, Attend, and Walk: Neural Mapping of Navigational Instructions to Action Sequences. In *Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, 2016.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. Investigation of Recurrent-Neural-Network Architectures and Learning Methods for Spoken Language Understanding. In *Proceedings of the 14th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2013.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

Dipendra Misra, John Langford, and Yoav Artzi. Mapping Instructions and Visual Observations to Actions with Reinforcement Learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1015–1026. Association for Computational Linguistics, 2017.

Ishan Misra, Abhinav Gupta, and Martial Hebert. From Red Wine to Red Tomato: Composition with Context. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 6, 2017.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. In *International Conference on Machine Learning (ICML)*, pages 1928–1937, 2016.

Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. Multi-Domain Dialog State Tracking Using Recurrent Neural Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2015.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. *Computing Research Repository*, arXiv:2003.04960, 2020.

Chahab Nastar, Matthias Mitschke, and Christophe Meilhac. Efficient Query Refinement for Image Retrieval. In *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 547–552. IEEE, 1998.

Khanh Nguyen and Hal Daume III. Help, Anna! Visual Navigation with Natural Multimodal Assistance via Retrospective Curiosity-Encouraging Imitation Learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Aishwarya Padmakumar and Raymond J. Mooney. Dialog as a Vehicle for Lifelong Learning. In *Position Paper Track at the SIGDIAL Special Session on Physically Situated Dialogue (RoboDial 2.0)*, July 2020.

Aishwarya Padmakumar and Raymond J. Mooney. Dialog Policy Learning for Joint Clarification and Active Learning Queries. *Computing Research Repository*, arXiv:2006.05456, June 2020.

Aishwarya Padmakumar, Jesse Thomason, and Raymond J. Mooney. Integrated Learning of Dialog Strategies and Semantic Parsing. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 547–557, Valencia, Spain, April 2017.

Aishwarya Padmakumar, Peter Stone, and Raymond J. Mooney. Learning a Policy for Opportunistic Active Learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Brussels, Belgium, November 2018.

Natalie Parde, Adam Hair, Michalis Papakostas, Konstantinos Tsiakas, Maria Dagioglou, Vangelis Karkaletsis, and Rodney D. Nielsen. Grounding the Meaning of Words through Vision and Interactive Gameplay. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1895–1901, Buenos Aires, Argentina, 2015.

Chris Paxton, Yonatan Bisk, Jesse Thomason, Arunkumar Byravan, and Dieter Foxl. Prospection: Interpretable Plans from Language by Predicting the Future. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6942–6948. IEEE, 2019.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, volume 1, pages 2227–2237, 2018.

Olivier Pietquin, Matthieu Geist, Senthilkumar Chandramohan, and Hervé Frezza-Buet. Sample-Efficient Batch Reinforcement Learning for Dialogue Management Optimization. *ACM Transactions on Speech and Language Processing*, 7(3):1–21, may 2011.

John Platt. Probabilistic Outputs for Support Vector Machines and Comparison to Regularized Likelihood Methods. *Advances in Large Margin Classifiers*, 1999.

Shaowei Png, Joelle Pineau, and Brahim Chaib-Draa. Building Adaptive Dialogue Systems Via Bayes-Adaptive POMDPs. *IEEE Journal of Selected Topics in Signal Processing*, 6(8):917–927, 2012.

Guo-Jun Qi, Xian-Sheng Hua, Yong Rui, Jinhui Tang, and Hong-Jiang Zhang. Two-Dimensional Multilabel Active Learning with an Efficient Online Adaptation Model for Image Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(10):1880–1897, oct 2009.

Rachel Rakov and Andrew Rosenberg. ”Sure, I Did The Right Thing”: A System for Sarcasm Detection in Speech. In *Proceedings of the 14th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2013.

Stephane Ross, Brahim Chaib-draa, and Joelle Pineau. Bayes-Adaptive POMDPs. In *Advances in Neural Information Processing Systems*, pages 1225–1232, 2008.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 627–635, 2011.

Nicholas Roy and Andrew McCallum. Toward Optimal Active Learning Through Sampling Estimation of Error Reduction. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*, pages 441–448, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, apr 2015.

Paul Ruvolo and Eric Eaton. ELLA: An Efficient Lifelong Learning Algorithm. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013.

Amrita Saha, Mitesh M Khapra, and Karthik Sankaranarayanan. Towards Building Large Scale Multimodal Domain-Aware Conversation Systems. In *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. A Survey of Statistical User Simulation Techniques for Reinforcement-Learning of Dialogue Management Strategies. *Knowledge Engineering Review*, 21(2):97–126, 2006.

Björn Schuller, Michel Valstar, Florian Eyben, Gary McKeown, Roddy Cowie, and Maja Pantic. AVEC 2011–The First International Audio/Visual Emotion Challenge. In *International Conference on Affective Computing and Intelligent Interaction*. Springer, 2011.

Thomas Scialom and Jacopo Staiano. Ask to Learn: A Study on Curiosity-driven Question Generation. *Computing Research Repository*, arXiv:1911.03350, 2019.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, volume 16, pages 3776–3784, 2016.

Burr Settles and Mark Craven. An Analysis of Active Learning Strategies for Sequence Labeling Tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. ACL, 2008.

Burr Settles. Active Learning Literature Survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010.

H. S. Seung, M. Opper, and H. Sompolinsky. Query by Committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory (COLT)*, pages 287–294, New York, NY, USA, 1992. ACM.

Manali Sharma and Mustafa Bilgic. Evidence-based Uncertainty Sampling for Active Learning. *Data Mining and Knowledge Discovery*, 31(1):164–202, April 2016.

Lanbo She and Joyce Chai. Interactive Learning of Grounded Verb Semantics towards Human-Robot Communication. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 1634–1644, 2017.

Lanbo She, Shaohua Yang, Yu Cheng, Yunyi Jia, Joyce Chai, and Ning Xi. Back to the Blocks World: Learning New Actions through Situated Human-Robot Dialogue. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 89–97, 2014.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10740–10749, 2020.

Carina Silberer and Mirella Lapata. Grounded Models of Semantic Representation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in*

Natural Language Processing and Computational Natural Language Learning (EMNLP), pages 1423–1433. Association for Computational Linguistics, 2012.

Carina Silberer and Mirella Lapata. Learning Grounded Meaning Representations with Autoencoders. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 721–732, 2014.

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-scale Image Recognition. *Computing Research Repository*, arXiv:1409.1556, 2014.

Jivko Sinapov, Priyanka Khante, Maxwell Svetlik, and Peter Stone. Learning to Order Objects using Haptic and Proprioceptive Exploratory Behaviors. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.

Mohan Singh, Eoin Curran, and Pádraig Cunningham. Active Learning for Multilabel Image Annotation. In *Proceedings of the 19th Irish Conference on Artificial Intelligence and Cognitive Science*, pages 173–182, 2009.

Mark Steedman and Jason Baldridge. Combinatory Categorial Grammar. *Non-Transformational Syntax: Formal and Explicit Models of Grammar*, pages 181–224, 2011.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. Learning to Map Context-Dependent Sentences to Executable Formal Queries. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 2238–2249. Association for Computational Linguistics, 2018.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement Learning: An Introduction*. MIT press, 1998.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

Hao Tan and Mohit Bansal. LXMERT: Learning Cross-Modality Encoder Representations from Transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint*

Conference on Natural Language Processing (EMNLP-IJCNLP), pages 5103–5114, 2019.

Hao Tan, Licheng Yu, and Mohit Bansal. Learning to Navigate Unseen Environments: Back Translation with Environmental Dropout. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2610–2621, 2019.

Stefanie Tellex, Ross A. Knepper, Adrian Li, Nicholas Roy, and Daniela Rus. Asking for Help Using Inverse Semantics. In *Proceedings of the 2016 Robotics: Science and Systems Conference (RSS)*, 2014.

Joseph Tepperman, David Traum, and Shrikanth Narayanan. ”Yeah Right”: Sarcasm Recognition for Spoken Dialogue Systems. In *Proceedings of the Ninth International Conference on Spoken Language Processing (ICSLP)*, 2006.

Christian Theobalt, Johan Bos, Tim Chapman, Arturo Espinosa-Romero, Mark Fraser, Gillian Hayes, Ewan Klein, Tetsushi Oka, and Richard Reeve. Talking to Godot: Dialogue with a mobile robot. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 2, pages 1338–1343. IEEE, 2002.

Jesse Thomason, Shiqi Zhang, Raymond Mooney, and Peter Stone. Learning to Interpret Natural Language Commands through Human-Robot Dialog. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1923–1929, 2015.

Jesse Thomason, Jivko Sinapov, Maxwell Svetlik, Peter Stone, and Raymond Mooney. Learning Multi-Modal Grounded Linguistic Semantics by Playing “I Spy”. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3477–3483, July 2016.

Jesse Thomason, Aishwarya Padmakumar, Jivko Sinapov, Justin Hart, Peter Stone, and Raymond J. Mooney. Opportunistic Active Learning for Grounding Natural Language Descriptions. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *In Proceedings of the 1st Annual Conference on Robot Learning (CoRL-17)*, pages 67–76, Mountain View, California, November 2017. PMLR.

Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. Vision-and-Dialog Navigation. In *Proceedings of the 2019 Conference on Robot Learning (CoRL)*, 2019.

Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. Vision-and-Dialog Navigation. In *Proceedings of the 4th Conference on Robot Learning (CoRL)*, pages 394–406, 2020.

Jesse Thomason. *Continually Improving Grounded Natural Language Understanding through Human-Robot Dialog*. PhD thesis, Department of Computer Science, The University of Texas at Austin, April 2018.

Blaise Thomson and Steve Young. Bayesian Update of Dialogue State: A POMDP Framework for Spoken Dialogue Systems. *Computer Speech and Language*, 24(4):562–588, 2010.

Sebastian Thrun and Tom M Mitchell. Lifelong Robot Learning. *Robotics and Autonomous Systems (RAS)*, 15(1-2):25–46, 1995.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5 - RMSProp: Divide the Gradient by a Running Average of its Recent Magnitude. *COURSERA: Neural networks for machine learning*, 2012.

Kinh Tieu and Paul Viola. Boosting Image Retrieval. *International Journal of Computer Vision*, 56(1-2):17–36, 2004.

Deepak Vasisht, Andreas Damianou, Manik Varma, and Ashish Kapoor. Active Learning for Sparse Bayesian Multilabel Classification. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD'14*, pages 472–481. ACM Press, 2014.

Adam Vogel, Karthik Raghunathan, and Dan Jurafsky. Eye Spy: Improving Vision through Dialog. In *Proceedings of the Twenty Fourth AAAI Conference on Artificial Intelligence (AAAI)*, pages 175–176, 2010.

Liwei Wang, Yin Li, and Svetlana Lazebnik. Learning Deep Structure-Preserving Image-Text Embeddings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5005–5013, 2016.

Peng Wang, Qi Wu, Jiewei Cao, Chunhua Shen, Lianli Gao, and Anton van den Hengel. Neighbourhood Watch: Referring Expression Comprehension Via Language-Guided Graph Attention Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1960–1968, 2019.

Garrett Warnell, Nicholas Waytowich, Vernon Lawhern, and Peter Stone. Deep TAMER: Interactive Agent Shaping in High-Dimensional State Spaces. In *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems. In *Proceedings of the 2015 Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2015.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems. In *Proceedings of the 2015 Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2015.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. A Network-based End-to-End Trainable Task-oriented Dialogue System. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, volume 1, pages 438–449, 2017.

David Whitney, Eric Rosen, James MacGlashan, Lawson LS Wong, and Stefanie Tellex. Reducing Errors in Object-Fetching Interactions Through Social Feedback. In *International Conference on Robotics and Automation (ICRA)*, 2017.

Jason D Williams and Geoffrey Zweig. End-to-end LSTM-based Dialog Control Optimized with Supervised and Reinforcement Learning. *Computing Research Repository*, arXiv:1606.01269, 2016.

Edward C Williams, Mina Rhee, Nakul Gopalan, and Stefanie Tellex. Learning to Parse Natural Language to Grounded Reward Functions with Weak Supervision. In *AAAI Fall Symposium on Natural Communication for Human-Robot Collaboration*, 2017.

Ronald J. Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. In *Reinforcement Learning*, pages 5–32. Springer US, 1992.

Yuk Wah Wong and Raymond J. Mooney. Learning Synchronous Grammars for Semantic Parsing with Lambda Calculus. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 960–967, 2007.

Mark Woodward and Chelsea Finn. Active one-shot learning. *Computing Research Repository*, arXiv:1702.06559, 2017.

Chien-Sheng Wu, Andrea Madotto, Zhaojiang Lin, Peng Xu, and Pascale Fung. Getting To Know You: User Attribute Extraction from Dialogues. In *Proceedings of the 1st Workshop on NLP for Conversational AI at ACL 2019*, 2019.

Fanyi Xiao, Leonid Sigal, and Yong Jae Lee. Weakly-Supervised Visual Grounding of Phrases With Linguistic Structures. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5945–5954, 2017.

Claudia Yan, Dipendra Kumar Misra, Andrew Bennett, Aaron Walsman, Yonatan Bisk, and Yoav Artzi. CHALET: Cornell House Agent Learning Environment. *Computing Research Repository*, arXiv:1801.07357, 2018.

Tsung-Yen Yang and Karthik Narasimham. Robust and Interpretable Grounding of Spatial References with Relation Networks. *Computing Research Repository*, arXiv:2005.00696, 2020.

Bishan Yang, Jian-Tao Sun, Tengjiao Wang, and Zheng Chen. Effective Multi-label Active Learning for Text Classification. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD'09*. ACM Press, 2009.

Shaohua Yang, Qiaozi Gao, Changsong Liu, Caiming Xiong, Song-Chun Zhu, and Joyce Y Chai. Grounded Semantic Role Labeling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 149–159, 2016.

Linwei Ye, Mrigank Rochan, Zhi Liu, and Yang Wang. Cross-Modal Self-Attention Network for Referring Image Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10502–10511, 2019.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. The Hidden Information State Model: A Practical Framework for POMDP-based Spoken Dialogue Management. *Computer Speech and Language*, 24(2):150–174, 2010.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. POMDP-based Statistical Spoken Dialog Systems: A Review. *Proceedings of the IEEE*, 101(5):1160–1179, 2013.

Yanchao Yu, Arash Eshghi, and Oliver Lemon. Training an Adaptive Dialogue Policy for Interactive Learning of Visually Grounded Word Meanings. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, page 339, 2016.

Yanchao Yu, Arash Eshghi, and Oliver Lemon. Learning How to Learn: An Adaptive Dialogue Agent for Incrementally Learning Visually Grounded Word Meanings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

Yanchao Yu, Arash Eshghi, Gregory Mills, and Oliver Lemon. The BURCHAK corpus: a Challenge Data Set for Interactive Learning of Visually Grounded Word Meanings. In *Proceedings of the 6th Workshop on Vision and Language at EACL 2017*, 2017.

Yuyin Sun and D. Fox. NEOL: Toward Never-Ending Object Learning for Robots. In *Procedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

Luke Zettlemoyer and Michael Collins. Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*, 2005.

Shiqi Zhang and Peter Stone. CORPP: Commonsense Reasoning and Probabilistic Planning, as Applied to Dialog with a Mobile Robot. In *Proceedings of the 29th Conference on Artificial Intelligence (AAAI)*, 2015.

Yu Zhang and Qiang Yang. A Survey on Multi-Task Learning. *Computing Research Repository*, arXiv:1707.08114, 2017.

Jiaping Zhang, Tiancheng Zhao, and Zhou Yu. Multimodal Hierarchical Reinforcement Learning Policy for Task-Oriented Visual Dialog. In *Proceedings of the 19th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 140–150, 2018.

Junjie Zhang, Qi Wu, Chunhua Shen, Jian Zhang, Jianfeng Lu, and Anton Van Den Hengel. Goal-oriented Visual Question Generation via Intermediate Rewards. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 186–201, 2018.

Yan Zhu, Shaoting Zhang, and Dimitris Metaxas. Interactive Reinforcement Learning for Object Grounding via Self-talking. *Computing Research Repository*, arXiv:1712.00576, 2017.

Wang Zhu, Hexiang Hu, Jiacheng Chen, Zhiwei Deng, Vihan Jain, Eugene Ie, and Fei Sha. BabyWalk: Going Farther in Vision-and-Language Navigation by Taking Baby Steps. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

Vita

Aishwarya Padmakumar was brought up in Bangalore, India where she completed her schooling at National Public School, Indiranagar. She went on to pursue an undergraduate degree at the Indian Institue of Technology, Madras in Chennai, India and graduated in May 2015 with a B.Tech. (Hons.) in the Department of Computer Science. As a part of this degree, she completed a senior thesis titled “Improving Aggregate Diversity in Recommender Systems” supervised by Prof Balaraman Ravindran. During this period, she also completed summer internships at SAP Labs, Google, and Adobe Research in Bangalore, India.

She enrolled in the PhD program in the Department of Computer Science at the University of Texas at Austin in August 2015. Her research work during her PhD has been published at top venues including EACL 2017, CoRL 2017 and EMNLP 2018. She has also been a reviewer for conferences such ACL, EMNLP, CoRL and AAAI in the last two years. During her PhD, she has also completed summer internships at Facebook, Menlo Park and Google, Mountain View.

Permanent Address: aishwarya@utexas.edu

This dissertation was typeset with LaTeX 2_ε[1] by the author.

---

[1] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay, James A. Bednar, Ayman El-Khashab and Dan Garette. The author used a template released by Dan Garette with small modifications to account for changes in formatting requirements.