INCREMENTALITY FOR VISUAL REFERENCE RESOLUTION IN SPOKEN DIALOGUE SYSTEMS

by

Ramesh Radhakrishna Manuvinakurike

A Dissertation Presented to the
FACULTY OF THE USC GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(COMPUTER SCIENCE)

December 2019

Committee

Advisors: Kallirroi Georgila (Committee Chair), David DeVault

Committee: Paul Rosenbloom, Jonathan Gratch, Morteza Dehghani

Dedication

To my dad ...

Acknowledgments

I am indebted to my advisors Prof. David DeVault and Prof. Kallirroi Georgila. They've believed and put their faith in me all these years. This work wouldn't be possible without them. They are the best mentors one can ever get! This work wouldn't have been possible without inputs from my committee. I would especially like to thank Prof. Morteza Dehghani, Prof. Jonathan Gratch and Prof. Paul Rosenbloom for their valuable time and insights.

I am thankful to my wonderful family who never gave me a chance to feel lonely in this wonderful journey. I am thankful to my parents who shared my enthusiasm and happiness from every published paper. Their enthusiasm kept me going. My wife, Deepthi Karkada has been one of my biggest supporters and has cheered me on all these years. She has not only been a supporter but also a partner in crime, who contributed with important thoughts and insights. My sister, Shwetha Manuvinakurike was more eager to see me graduate than myself. I would love to thank my grandpa, grandma, aunts and uncles who have showered me with their love and support all through.

I had the pleasure of working with some of the best minds in the field from whom I got to learn a great deal. I had the pleasure to work with Maike Paetzel every year of my Ph.D. She is an amazing team mate. Working with David Schlangen, Casey Kennington, and Julian Hough during my summer internship at Bielefeld

University was one of the most educative experiences of my life. I am particularly thankful to Timothy Bickmore who helped me begin my career in research by accepting me as a student during my Masters at Northeastern University, Boston. I am also thankful to Sudarshan Iyengar, who motivated me to aspire to pursue a PhD. I was lucky to be a member of the Institute for Creative Technologies which provided me with the best environment for conducting research. I will always use it as a gold-standard baseline for comparing any research institute. I was lucky to get insightful inputs from David Traum, Ron Artstein, and Anton Leuski at the Natural Language Dialogue group.

I am extremely grateful to Trung Bui, Saurav Sahay, Lama Nachmann, Walter Chang, and Shachi Kumar with whom I got a chance to work and learn about research in industry.

Last but not least my co-graduate students who need special mentioning are Eli Pincus, Vinod Sharma, Rens Hoegen, Sayan Gosh, Yatin Wadhawan, Jacqueline Brixey, Christopher Weinberg, and Elnaz Nouri. They were fantastic! I also want to thank Lizsl De Leon for helping in various stages.

Contents

List of Tables

List of Figures

Abstract

There are several challenges involved in developing naturally and efficiently conversing Spoken Dialogue Systems (SDS). Fluid turn-taking is one such challenge. In traditional SDS the system has to wait until the user has finished speaking before starting to process the user's input in order to respond to it. Incremental dialogue processing is an important feature in SDS that can make them more efficient and natural compared to their non-incremental counterparts. Incrementality in SDS means that the processing of the user utterances (word-by-word) starts before the input is completely available and the system generates output increments as soon as possible, which in turn results in natural and efficient human-system interaction.

In this thesis, I first present models for incremental dialogue policy development and show that the incremental dialogue policy helps SDS perform better than their non-incremental counterparts and about as well as humans. Then I present an incremental dialogue policy developed using reinforcement learning. The work then provides models for incremental utterance segmentation and labeling in multiple domains. The work also applies transfer learning towards the development of an incremental dialogue policy. I show that an effective policy can be learned with fewer data samples using a transfer learning approach. Collecting data for the development of an SDS is an important step. In this work, I also develop a web-based crowd-sourcing framework for collecting spoken interaction data.

Chapter 1

Introduction

> *"Knowing is not enough; we must apply. Wishing is not enough; we must do."*
>
> – Johann Wolfgang Von Goethe, *Writer, Statesman*

1.1 Introduction

Recent advancements in Spoken Dialogue Systems (SDS) have been very exciting. These advancements can be largely attributed to the improvements in the performance (accuracy & speed) of components that make up a typical SDS pipeline such as speech recognition systems, and language understanding and generation modules. For instance, recent Automatic Speech Recognizers (ASR) outperform humans in the task of transcribing speech, i.e., converting speech to text (Xiong et al. (2018))[1]. Similar advancements driving the development of text-to-speech, language understanding, and language generation have helped achieve progress in the field of SDS development. As a result of these advancements, numerous SDS that serve as personal task assistants (e.g., Alexa, Siri, Google Home, Cortana, etc.), health assistants (e.g., DeVault et al. (2014); Stratou et al. (2015); Kenny et al. (2008); Bickmore et al. (2005); Manuvinakurike et al. (2014)), pedagogical agents (e.g., Litman & Silliman (2004); Graesser et al. (2005); Traum et al. (2015); DeVault et al. (2011b)), etc. have been developed in recent times. Despite this

[1] Using audio of high quality and non-spontaneous speech.

progress, SDS continue to score low on naturalness and lack scores of capabilities. This work aims to bridge this gap by providing methods to make SDS more natural and efficient.

The human dialogue processing capability is vastly superior compared to current state-of-the-art SDS. This superior human performance can be attributed to a variety of dialogue processing behaviors that are not typically exhibited by current SDS. For instance, humans generate back-channels, acknowledgments, laughter, and use a multitude of other non-verbal behaviors (e.g., head nods, gaze, etc.). Also, humans do not wait for their partner to finish speaking to build an understanding of what is being spoken. Human understanding happens incrementally word-by-word (or even sub-word) as and when the information is available (Marslen-Wilson (1973)). This equips humans with the capability to generate real-time feedback (e.g., uh-huh, hmm, head nods, etc.), interrupt, seek clarifications by asking questions, generate utterances to collaboratively complete their interlocutor's partial utterances, etc. Such conversational behaviors are part of what makes human conversation natural and efficient.

These types of behaviors constitute what we call 'incremental' behaviors in the context of SDS. These behaviors, referred to as 'incremental processing', are the focus of this work. In fact they are so important that not exhibiting them makes SDS seem significantly less natural and less efficient (e.g., see Aist et al. (2007b)). Developing such SDS that score high on naturalness and effectiveness is a very challenging task and an open research problem.

The goal of this work is multi-fold. First, I study incrementality, which is central to achieving the goal of natural and efficiently conversing SDS. We run experiments to demonstrate this in a live human study. A significant portion of this work is devoted towards modeling incrementality (mainly in the language understanding and dialogue management modules) and studying the importance of incrementality in SDS. Second, the work studies the research questions involved in building an incremental SDS. This involves an incremental SDS architecture and a data collection methodology. This incremental SDS must be quick and capable of processing the user input as soon as the user issues an utterance. Over the course of building an incremental SDS, numerous challenges and research questions emerge, mainly related to data collection and processing. Finally, I focus on extending the work in three directions: i) Addition of semantic processing (such as dialogue act segmentation and labeling) with a focus on incrementality. ii) Extending the work to other domains. This includes the application of incrementality to real-world problems and transferring an incremental dialogue policy learned in one domain to other related domains, iii) Multi-modal fusion of vision and language by using a visually grounded language understanding model called 'words as classifiers' (Kennington & Schlangen (2015)).

1.1.1 Claims & Contributions

The main claims & contributions of this work are:

1. **Incremental SDS are more efficient at performing a given task and are perceived by users as better than alternative architectures:** SDS with 'incremental' processing capabilities outperform their 'non-incremental' counterparts. With 'incremental' processing SDS achieve performance comparable to that of humans in a situated task. This is the first work in the literature that shows conclusively that incremental SDS perform better qualitatively and quantitatively for a given task compared to non-incremental SDS. This work specifically tests two hypotheses:

  - Incremental processing results in a better quantitative task performance than alternative non-incremental architectures and achieves performance comparable to humans.

  - SDS with incremental processing capabilities are qualitatively perceived by users as better than non-incremental SDS.

2. **Reinforcement learning approaches help learn a better dialogue policy:** Reinforcement Learning (RL) in the literature has yielded superior performance compared to other approaches for dialogue policy learning. However, it is not clear in the literature how much effort is involved in the development of baselines used for comparing against RL approaches. This is understandable as the focus is typically on the RL algorithms and their application to dialogue policy learning. Thus, the advantages of using RL in learning a policy are often unclear. In this work:

   - By comparing an RL policy with a strong baseline in an offline study (using real user dialogue data), we show that the RL approach learns a better policy than a carefully designed rule-based policy. In the literature, the baselines used for comparing against RL policies are often weak. In this work, we show that RL learns a better policy compared to a strong baseline that achieves near-human-level performance.

   - In an online user study we also show that the RL policy achieves better qualitative performance. This is also the first work in the literature that employs RL in support of developing a dialogue policy for an incremental reference resolution task.

3. **Transfer learning approaches help learn a better dialogue policy with fewer data points:** Using transfer learning, an incremental dialogue

policy learned in one domain (source domain) can converge faster when transferred to a new domain (target domain) and also achieve better task performance with fewer data points. This is the first work in the literature that utilizes transfer learning for learning incremental dialogue policies.

- With transfer learning a dialogue policy can be learned for a specific situated task in a target domain using information from a related source domain, and help the agent achieve better performance compared to the condition that does not employ transfer learning.

4. **Incremental segmentation and labeling in real-world SDS:** Incremental segmentation and dialogue act labeling help save time in human-system interaction by helping the SDS identify the user's intentions before completion of the user's utterance. This is the first work in the literature that applies incremental segmentation and dialogue act labeling to a real-world problem and shows that the savings achieved hold promise for including incremental policies in SDS. More specifically, this is the first work providing evidence that incremental segmentation and dialogue act labeling can be more efficient for the user and save time in accomplishing tasks in a real-world application. The hypotheses tested are:

    - Segmentation and dialogue act labeling can help develop more incremental dialogue systems without compromising task performance.
    - Segmentation and dialogue act labeling applied to incremental processing can be used in a real-world problem and help achieve time savings.
    - In the domains that involve grounding the meaning of words on visual features, the segmentation and labeling approach helps outperform baseline models that do not use segmentation and labeling.

5. **Crowd-sourcing offers a cheaper and better alternative for building spoken dialogue corpora:** Data is very important for the development of an SDS. Crowd-sourcing serves as an effective platform for collecting spoken dialogue data despite the users not owning quality audio equipment. The disadvantage incurred due to the reduced quality of data is compensated by the diversity and larger user participation. This is the first work in the literature that develops a web-framework in support of collecting crowd-sourced spoken dialogue corpora for building incremental SDS. The hypotheses tested are:

    - Crowd-sourcing spoken dialogue data collection is cheaper and faster than traditional in-lab methods. This is not trivial as preparing the data for SDS development requires multiple steps, i) collect, ii) clean, iii) transcribe, iv) annotate, and each of these steps presents a unique challenge.
    - Data collected using crowd-sourcing is comparable in quality with data collected in the lab.

The outline of this work is discussed in detail in Section 1.2 which is summarized in Figure 1.1.

1.2 Thesis Outline

In this section, I will discuss the outline for the rest of the work.

Related work In Chapter 2, I begin with discussing a typical general SDS architecture. The general SDS architecture is a pipeline model that takes user speech as input and generates the system response as output. It has been argued in the literature (e.g., Allen et al. (2001); Schlangen & Skantze (2011)) that the standard

SDS pipeline needs modification for accommodating truly incremental SDS. I discuss these incremental SDS architectures that have been proposed for accommodating incremental dialogue processing. In this work, I adopt incrementality in a standard SDS pipeline architecture with high-frequency message-passing to show that SDS can be made incremental while still being based on the general pipeline architecture. Chapter 2 also discusses past works on incrementality in the context of SDS. With a few examples, I also motivate the importance of including incrementality in SDS.

Data Data collection and annotation are important steps in the development process of SDS. In order to build an incremental SDS, it is necessary to have data consisting of human conversations recorded with accurate timing information (e.g., the ASR needs to provide timing information along with the recognized words as noted in Baumann et al. (2017)), transcribed and annotated with a relevant annotation scheme. It has become clear in the literature that collecting human-human conversations is a necessary first step in the development of SDS (Lasecki et al. (2013)) as this data can be used for training the components of the SDS pipeline. The common practice is to collect conversations between humans in a controlled lab setting with high-quality recording equipment. Such in-lab methods are beneficial as the quality of the data collected in such settings can be controlled well. Audio data collected in such settings have high fidelity (low background noise) as the quality of the equipment used to record the audio can be controlled. Additionally, there can be detailed delivery of instructions from the research staff to the recruited subjects. However, collecting a large dataset of hundreds (if not thousands) of conversations is often a massive undertaking requiring large amounts of time and money. These experiments also suffer a setback as the subjects that are recruited for the experiments are demographically constrained. The subjects are brought to the lab from the area around the location of the experiment for collecting the data, and this results in data of lower demographic diversity. In recent times, crowd-sourcing has become extremely popular as it addresses the diversity problem and at the same time provides a large pool of participants. Web-based crowd platforms allow parallelizing the process of data collection which allows collecting a large corpus in a short period of time. Crowd-sourcing is popular for collecting data and is being used extensively in many applications such as the development of ASR models (Panayotov et al. (2015)), computer vision (Lin et al. (2014)), etc., and has helped achieve tremendous progress. However, collecting real-time conversational data using crowd-sourcing remains a significant challenge as the infrastructure required for such data collection is engineering intensive. Furthermore, the quality of the environment and the equipment are hard to control. It is also not clear how the dialogues collected will compare with the tried and tested methods of in-lab data collection. After data collection, the data need to be curated and annotated so that they can be used to train machine learning models.

In Chapter 3, I present an SDS for a visual reference resolution application and the steps involved in building it. This includes the domain description, data collection, and software architecture of the required tools for data collection and the SDS. In this chapter, I discuss the architecture and software developed which could be useful for developing similar systems in the future. The trade-offs involved in data collection for SDS in the lab and using crowd-sourcing are not clear. In this chapter through an experiment, we compare the difference in the quality of the data. We also discuss the software architecture of the SDS developed for the crowd-sourcing environment, and run experiments using crowd-workers and detail the advantages of using the crowd-sourcing paradigm. The discussions presented in this chapter are covered in Manuvinakurike et al. (2015) and Manuvinakurike & DeVault (2015).

Incrementality In Chapter 4, I discuss incrementality in SDS developed for a game called RDG-Image (Rapid Dialogue Game - Images). The SDS developed is named *Eve*. I discuss the algorithms involved in training *Eve*'s components such as language understanding and dialogue management. We compare the different versions of *Eve*'s dialogue policy with a varying degree of incrementality and show the advantages of developing an SDS that is highly incremental. *Eve*'s performance is measured quantitatively and qualitatively in a study with human users conducted using crowd-sourcing. We achieve near-human-level performance in a dialogue task and thereby show the advantage of developing a highly incremental SDS. The discussions presented in this chapter are covered in Paetzel et al. (2015) and Manuvinakurike et al. (2015).

In Chapter 5, we extend *Eve* by using the reinforcement learning (RL) framework for learning an incremental dialogue policy. In the literature of dialogue systems when comparing the performance of an RL dialogue policy against a baseline, the effort spent on the development of the baseline is minimal. This is understandable as the focus of the research is often not on the baseline but rather on the RL algorithms under study. In this work, we compare the performance of an RL-based *Eve* against a very strong baseline *Eve* (based on carefully designed rules) which performs as well as humans, and against human performance. This discussion is covered in Manuvinakurike et al. (2017).

The application of RL also opens up an avenue for exploring how this work can be applied to other related domains using transfer learning, which will be studied as well. We utilize a pre-existing resource that does not consist of any dialogues in our target domain, and show that transferring the incremental dialogue policy learned in another source domain to our target domain helps the agent outperform a similar agent not using the transfer learning methods. In Chapter 5, we also show that by using transfer learning we can train models faster in our target domain with fewer data points.

Semantics Human language involves complex semantics usage. In Chapter 6, we study methods to equip SDS with incremental language understanding capabilities by including incremental segmentation and dialogue act detection modules. Incremental segmentation and dialogue act detection correspond to classifying the dialogue segments in the user utterance and identifying the dialogue act of the corresponding segment. We explore incremental segmentation and dialogue act detection in SDS for two different domains involving different challenges and complexities. The domains covered in this chapter can also be found in Paetzel et al. (2014), Manuvinakurike & DeVault (2015), Manuvirakurike et al. (2018), and Manuvinakurike et al. (2018a). We extend the work to a real-world problem and show the importance of segmentation and dialogue act detection. We also show the importance of including such capability in SDS by measuring the *savings* in time and the task performance. In this work, we focus on the understanding capabilities of SDS. We utilize state-of-the-art deep learning techniques for building dialogue act labelers and show that such methods can outperform traditional classifiers. This discussion is covered in Manuvinakurike et al. (2016, 2018b).

Another direction explored in this work is to include vision capabilities in SDS. When the language understanding module makes a reference resolution decision, we utilize basic vision features and combine this information with language information. We show that segmentation and labeling are important in making decisions about complex visual domains and help improve the performance of the agent in a reference resolution task. This is dealt with in Chapter 7.

Figure 1.1 shows the outline of the work.

1.2.1 Relation to Previous Publications

The framework for crowd-sourcing spoken dialogue data and the qualitative and quantitative comparisons with in-lab collected data has been published in Manuvinakurike & DeVault (2015). The extension of the framework to support dialogue data collection with an incremental agent and the architecture of the agent is published in Manuvinakurike et al. (2015). The development of the dialogue policy based on Carefully Designed Rules (CDR) and the study with humans users comparing incremental versus non-incremental architectures are published in Paetzel et al. (2015). The development of the RL-based policy and the offline comparison (using real user data) with the CDR policy are published in Manuvinakurike et al. (2017). The extension of the work with incremental segmentation and dialogue act detection is published in Manuvinakurike et al. (2016). The application of incremental segmentation and dialogue act detection to a complex image editing domain is published in Manuvinakurike et al. (2018a), Manuvirakurike et al. (2018), and Manuvinakurike et al. (2018b). Finally, the models for incremental segmentation and dialogue act detection for fine-grained understanding using basic visual features have been published in Manuvinakurike et al. (2016) and Zarrieß et al. (2016).

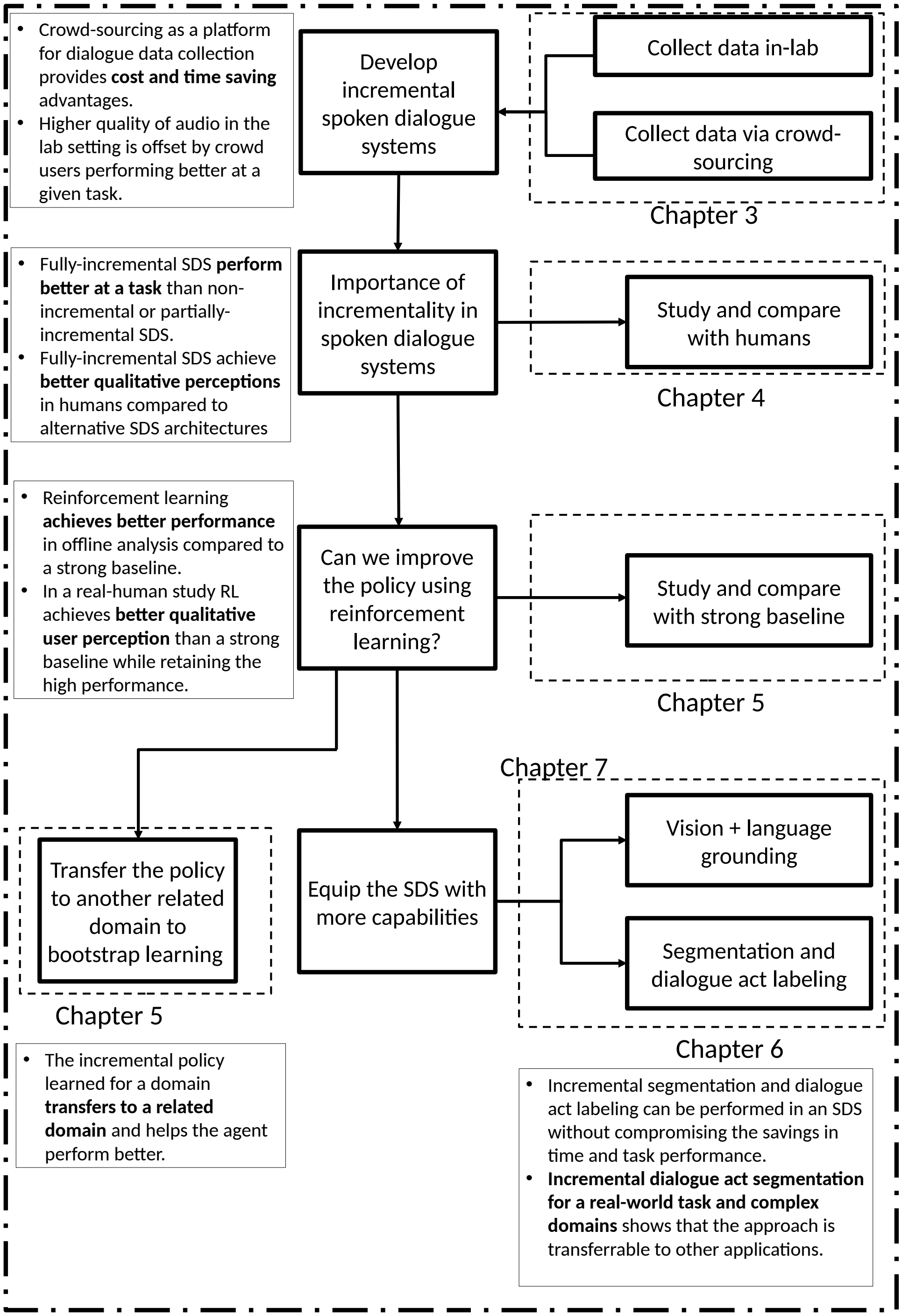

• Crowd-sourcing as a platform for dialogue data collection provides cost and time saving advantages.
• Higher quality of audio in the lab setting is offset by crowd users performing better at a given task.
Develop incremental spoken dialogue systems
Collect data in-lab
Collect data via crowd-sourcing
Chapter 3
• Fully-incremental SDS perform better at a task than non-incremental or partially-incremental SDS.
• Fully-incremental SDS achieve better qualitative perceptions in humans compared to alternative SDS architectures
Importance of incrementality in spoken dialogue systems
Study and compare with humans
Chapter 4
• Reinforcement learning achieves better performance in offline analysis compared to a strong baseline.
• In a real-human study RL achieves better qualitative user perception than a strong baseline while retaining the high performance.
Can we improve the policy using reinforcement learning?
Study and compare with strong baseline
Chapter 5
Chapter 7
Transfer the policy to another related domain to bootstrap learning
Equip the SDS with more capabilities
Vision + language grounding
Segmentation and dialogue act labeling
Chapter 5
Chapter 6
• The incremental policy learned for a domain transfers to a related domain and helps the agent perform better.
• Incremental segmentation and dialogue act labeling can be performed in an SDS without compromising the savings in time and task performance.
• Incremental dialogue act segmentation for a real-world task and complex domains shows that the approach is transferrable to other applications.

Figure 1.1: Outline of this work and sequence in which topics are covered. The contributions made in each section are highlighted and covered in the corresponding chapters.

Chapter 2

Spoken Dialogue Systems & Incrementality

In this chapter, I discuss the two centerpieces of this work: i) Spoken Dialogue Systems (SDS) architecture, ii) incrementality.

The outline for the chapter is as follows: In Section 2.1, I discuss the components of a typical SDS pipeline. The SDS processes the speech input from the user and generates a relevant spoken response. To enable this, the SDS needs to convert speech to text, understand the user utterance, process, and generate the response for the user. We discuss the components which typically enable SDS in achieving this. In Section 2.2, I discuss incremental speech processing (referred to as 'incrementality') in SDS. Incrementality in SDS impacts the traditional SDS processing pipeline as the relevant decisions need to be made with low latency. The advantages of incremental SDS are explained with a few examples in Section 2.2.3, which motivate the domains used in this work.

2.1 Spoken Dialogue System Architecture

A generic standard SDS pipeline architecture is discussed in this section. We look at the components of such an SDS pipeline and their functionalities. The SDS takes speech input from the user and generates speech output for the user with discrete processing functions performed by the intermediate components. Figure 2.1 shows the components of a typical SDS pipeline.

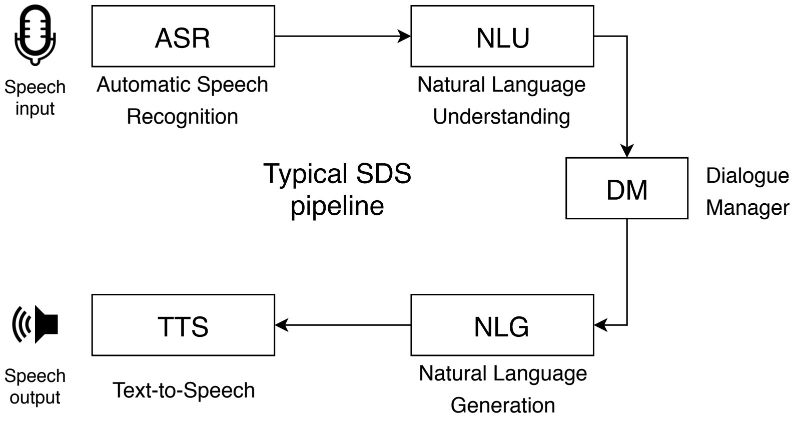


Figure 2.1: The pipeline of a typical SDS.

Now, I discuss the components of the SDS pipeline and briefly describe the functions with a few examples.

2.1.1 Automatic Speech Recognizer

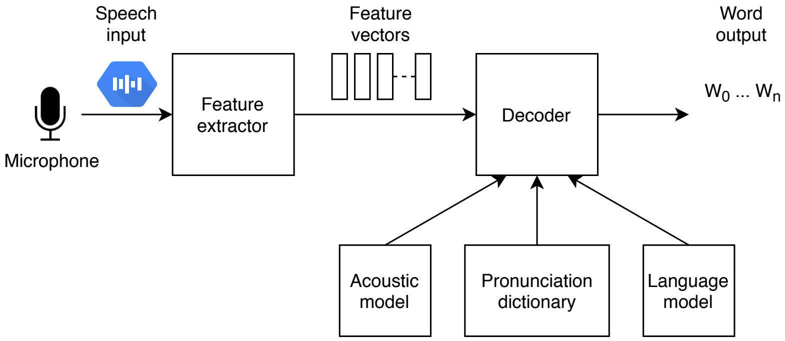


Figure 2.2: The modules of a typical HMM-based ASR (Gales et al. (2008)).

The Automatic Speech Recognizer (ASR) converts the speech input to a text output called transcription. Examples of such ASR include Kaldi (Povey et al. (2011)), PocketSphinx (Huggins-Daines et al. (2006)), Google speech recognizer[1], Microsoft Cortana[2], etc. ASR in recent times have achieved low WER ($\sim 5\%$) (Word Error Rate) which is a measure of how well the ASR converts the audio input to correct words.

HMM (Hidden Markov Model) based speech recognizers such as Sphinx, etc. have achieved low WER in recent times. Such ASR takes audio input that is typically sampled between 16KHz - 44.1KHz by standard microphones. These audio samples are down-sampled to standard 16KHz that are then input to audio feature extractors. The feature extractors encode the audio samples as MFCC features (Mel-Frequency Cepstral Coefficients). These encoded samples are then decoded to the text output that is made up of the most likely sequence of words by the decoder module. This decoder module utilizes i) an acoustic model, ii) a pronunciation dictionary, and iii) a language model to produce the word sequence output. The acoustic model and the language model determine the likelihood of the sequence of words. The acoustic model converts the audio features into a phoneme (basic units of sounds) sequence which is then converted to word output using the pronunciation dictionary. The language model is then used to model the most probable sequence of words by searching through the likely possible sequences of words and then limit the list of possible hypotheses using pruning. Figure 2.2 shows the architecture of a typical HMM-based ASR. Typically the ASR produces n-best text hypotheses (transcriptions) of the input audio samples. A detailed description of such an ASR can be found in Gales et al. (2008).

[1] https://cloud.google.com/speech-to-text/

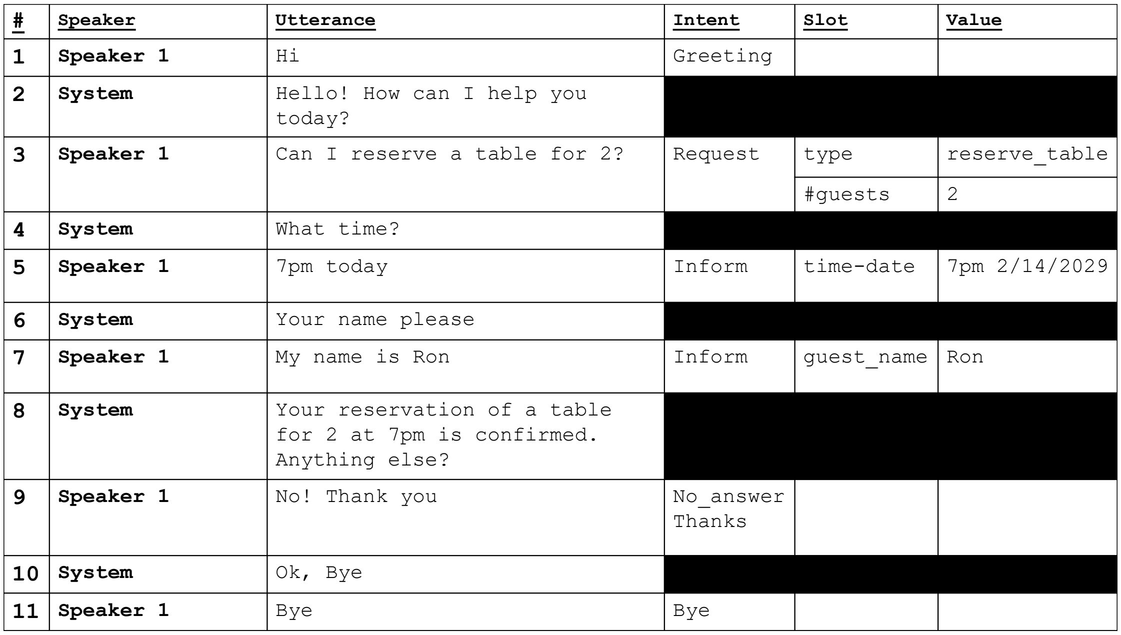
[2] https://azure.microsoft.com/en-us/services/cognitive-services/speech-to-text/

In recent times the best performing ASR have been built using deep learning models for acoustic and language modeling (Xiong et al. (2018); Panayotov et al. (2015)) and achieve a WER as low as 5% on non-spontaneous dictation based speech. The WER on spontaneous conversation is much higher[3].

End-to-end speech recognition (Graves et al. (2013)) has also received much attention because it promises conversion of speech to text with simpler modeling. Such systems convert the input speech signal into sequences of words without explicit acoustic models, pronunciation dictionary, and language models. Such systems have also achieved appreciably low WER in recent times (Amodei et al. (2016)). ASR continue making progress in achieving lower WER.

Many standard out-of-the-box ASR are available (Google, Microsoft, Watson, Alexa, etc.) which operate on the cloud and provide online decoding functionalities (i.e., provide text output before the complete utterance is spoken). Such services along with standalone ASR toolkits such as Kaldi (Povey et al. (2011)) and PocketSphinx (Huggins-Daines et al. (2006)) are commonplace in an SDS pipeline.

The main factors that play an important role while choosing an ASR for building an incremental SDS are low WER, a capability to perform live online decoding (convert speech to text incrementally), and low latency. In this work, we adopt the standalone Kaldi ASR and Google cloud ASR for the development of incremental SDS as they both satisfy the requirements for building incremental SDS. This work does not claim any novel contributions towards ASR research and architecture.



#	Speaker	Utterance	Intent	Slot	Value
1	**Speaker 1**	Hi	Greeting		
2	**System**	Hello! How can I help you today?			
3	**Speaker 1**	Can I reserve a table for 2?	Request	type	reserve_table
				#guests	2
4	**System**	What time?			
5	**Speaker 1**	7pm today	Inform	time-date	7pm 2/14/2029
6	**System**	Your name please			
7	**Speaker 1**	My name is Ron	Inform	guest_name	Ron
8	**System**	Your reservation of a table for 2 at 7pm is confirmed. Anything else?			
9	**Speaker 1**	No! Thank you	No_answer Thanks		
10	**System**	Ok, Bye			
11	**Speaker 1**	Bye	Bye		

Figure 2.3: An example showing how NLU operates. The user input is classified to intents, slots, and values.

2.1.2 Natural Language Understanding

The next component in the SDS pipeline is the NLU (Natural Language Understanding) module. The text output from the ASR is input to the NLU module, which infers the intentions and meanings of the user speech. The standard NLU tasks involve tree-based parsing, word-level sequence tagging, and utterance classification (intents, slots-values, dialogue acts, speech acts, etc.). The function of the NLU varies depending on the application of the SDS. Generally, the NLU converts the input text into higher level abstract representations. These representations are either specific to the SDS application (e.g., Lemon et al. (2006); Williams et al. (2015), DeVault et al. (2011b)), or general such as dialogue acts (e.g., Core & Allen (1997); Bunt et al. (2010); Bunt (2009)), or both. The primary function of

[3] A recent analysis of WER on conversational speech can be found in Baumann et al. (2017) and Morbini et al. (2013).

the NLU module is to convert the diverse user utterances into a standardized input representation that can be interpreted by the SDS. I will demonstrate one such application through an example.

In the example shown in Figure 2.3, a conversation between a speaker (Speaker 1) and the SDS (System) performing a restaurant table reservation task is demonstrated. The NLU in this example converts the utterances from the speaker (line #1,3,5,7,9,11) into intents, slots, and values representations. The intents are higher-level labels that specify types of user intentions. The users could intend to perform a task by choosing to generate numerous possible utterances, for example, the 'greeting' intent could be achieved using the utterances, 'hi' , 'hello', 'good morning', etc. The slots and values (Figure 2.3) are the task-specific information provided by the users to accomplish their goals, for example, # of guests, date, time, etc. In the example shown in Figure 2.3, the system can perform the task of booking the table only when it has all the required information (type of request, # of guests, date-time, and guest name). When the user issues a request (line # 3, *Can I reserve a table for 2?*), the system knows that the speaker intends to perform the task of *reserving* the table for *2 guests*. The task of the NLU in this example is to identify the user intents, slots, and values. It is important to note that the intents, slots, and values are not the only possible design alternatives for the NLU module. In DeVault & Traum (2012), the output of the NLU is in the form of an Attribute-Value matrix (AVM) which is a semantic frame consisting of a set of attributes and values derived from an ontology designed for the domain. They are also accompanied by the confidence score in the AVM.

In an SDS where the understanding of the user input is based on visual signals as well, the NLU needs to accommodate such vision input. Figure 2.4 shows such an example of a vision-based SDS. In this example, the user input consists of two parts, a *statement* and a *yes/no question*. In order to answer the question, the system needs to build an understanding of the image along with the user utterance. The information from the language and the visual modalities is combined, and the output is sent to the dialogue manager. Such SDS which require both spoken and visual input (e.g., Allen et al. (2001), Gorniak & Roy (2004)) have gained attention in recent times. Understanding of the words and the visual scene (objects, user gaze, head-nods, etc.) in such SDS is necessary. For example, in Kousidis et al. (2014) head-nods are used as an input to the dialogue manager to infer the user state. In recent times, this method of composition of information from the modalities (vision and language) has been done using deep learning approaches such as CNNs (Convolutional Neural Networks) and LSTMs (Long Short Term Memory Networks) (Antol et al. (2015)). In this work, we focus on such applications combining vision and speech.

The NLU produces the output in a format that is then utilized by the dialogue manager/policy. We will now see the function of the dialogue manager/policy.

2.1.3 Dialogue Manager

The Dialogue Manager (DM), also called action manager, policy, etc., takes the output generated by the NLU module and generates the action (often a response) that needs to be generated by the SDS. The DM utilizes the output of the NLU (semantic frame, intents-slot-value, dialogue acts, etc.) and in many cases also utilizes application specific database functionalities, and generates an output in the semantic form that is then utilized by the NLG (Natural Language Generation) module to produce the natural language text output.

Figure 2.5 shows the operation of a sample dialogue manager which is an extension of the example introduced in Figure 2.3. In this example, we zoom into

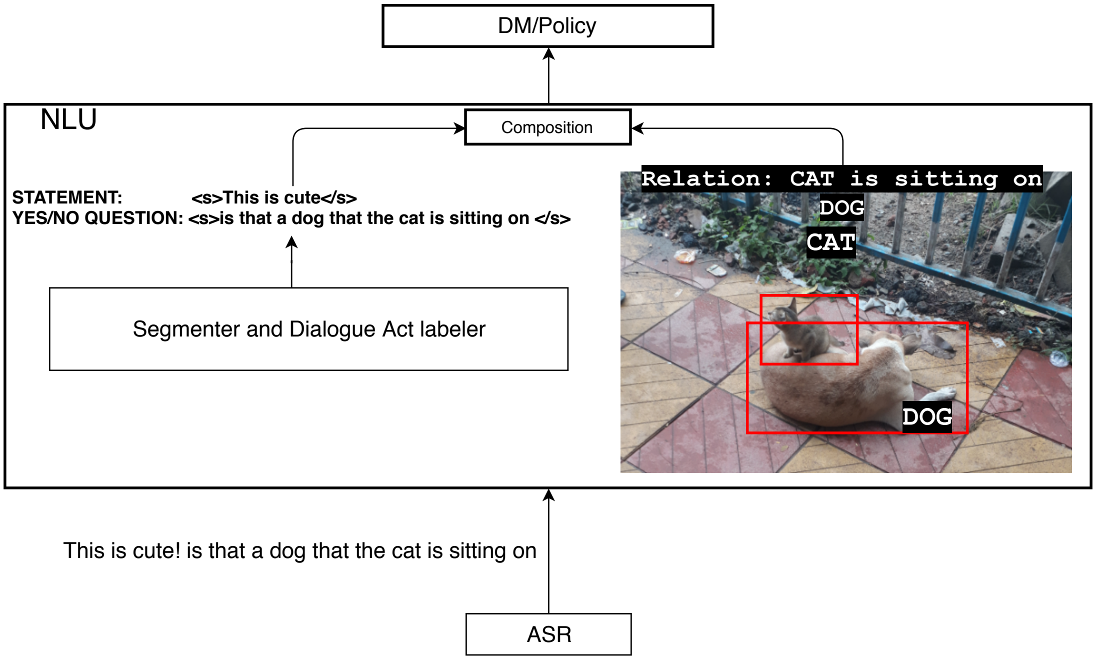
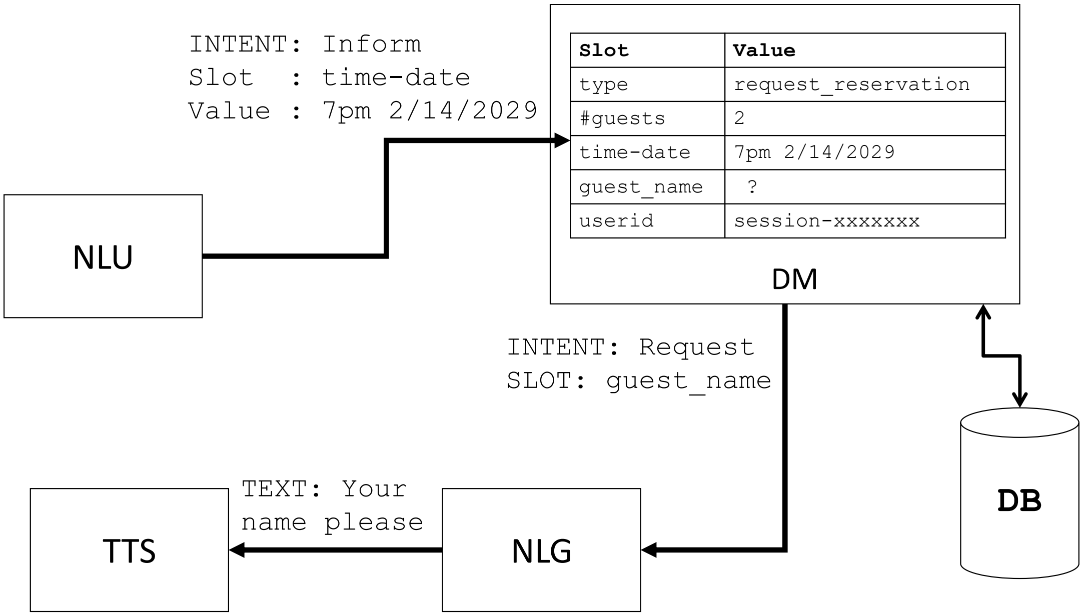

DM/Policy
NLU
Composition
STATEMENT: <s>This is cute</s>
YES/NO QUESTION: <s>is that a dog that the cat is sitting on </s>
Relation: CAT is sitting on
DOG
CAT
DOG
Segmenter and Dialogue Act labeler
This is cute! is that a dog that the cat is sitting on
ASR

Figure 2.4: An NLU architecture for a visual question-answering application.

the DM operation utterances indicated by #5 (Speaker 1: 7pm today) and #6 (System: Your name please) in Figure 2.3. Upon the speaker issuing the utterance #5 (7 pm today), the NLU converts the utterance into the semantic frame indicated in Figure 2.5 which consists of intent (inform), slot (time-date), and value (7 pm 2/14/2029). The system at this point figures out that it needs the *guest_name* from the user to complete the reservation. This *Request* is now sent to the NLG module which converts it to a natural language output. The DM, in order to accomplish this task, maintains the state information, which consists of information presented by the user until that point in the conversation. The DM also needs to track the state information of the user across a dialogue session. This task is called dialogue state tracking, and such systems are often called Information State Update dialogue systems (e.g., Georgila et al. (2005)).

The other task of the DM is to request additional information from the user. As in Figure 2.5, the DM issues a request *guest_name* information which is then transferred to the NLG module. When the SDS has enough information, it queries/updates the database (DB) with this information. As with the NLU, the design of the DM is specific to the application.



Slot	Value
type	request_reservation
#guests	2
time-date	7pm 2/14/2029
guest_name	?
userid	session-xxxxxxx

Figure 2.5: The operation of an example DM.

2.1.4 Natural Language Generation

The Natural Language Generation (NLG) module takes as input the output of the DM and converts it to a natural language response. Figure 2.5 shows an example of NLG operation where the DM output is converted to a natural language text output. A common NLG method is to employ a template-based approach using a set of templates to convert the frame output by the DM into text. For instance, in this example, each output from the DM is mapped to a specific natural language sentence. Such methods are often criticized as repetitive and not scalable. Since each DM output frame is mapped to a template, the generations produced by the SDS are often repetitive. A large number of possible output frames by the DM makes it challenging to generate templates in the case of open-domain SDS. For this reason, in recent times, data-based approaches to utterance generation have yielded promising results. For instance, Wen et al. (2015) use LSTMs (Hochreiter & Schmidhuber (1997)) for generating language. They show that the utterances generated with this method are rated favorably by human raters. We adopt template-based methods for NLG in this work. There are numerous challenges in developing an NLG module and several NLG approaches that have been proposed in the literature are covered in detail in Stent & Bangalore (2014).

2.1.5 Text-To-Speech

The natural language text response generated by the NLG module is converted into audio output by the Text-To-Speech (TTS) module. Two of the most important factors, guiding the decision criteria for selecting a TTS module and a specific synthetic voice, are the naturalness and intelligibility of the synthetic voice. A common approach to generating audio is to use pre-recorded human speech or an out-of-the-box TTS such as CereProc[4]. It is, however, important to note that the modulations of the voices and diverse tones generated by humans is an important aspect of the TTS module. The problem of choosing the right voice for an SDS is not trivial, and more detailed analysis is in Georgila et al. (2012). The audio generated is then played back to the user.

[4] https://www.cereproc.com/

2.1.6 Limitations

In the SDS architecture described, it is important to note the ‘pipeline’ nature of the SDS, i.e., each component depends on the input from the module that precedes it in the pipeline. This often imposes a strict-turn based limitation, especially if the ASR does not support online decoding. The strict nature of turn-taking is often undesirable in an incremental SDS (Allen et al. (2001)) which needs to revise the understanding and thus the generation mid-utterance. In the next section (Section 2.2) we discuss the incrementality and an SDS architecture that accommodate incremental processing.

2.2 Incrementality

In this section, I motivate incremental processing by describing the mechanisms observed in human language processing. I then discuss previous incremental dialogue processing work carried out in the SDS literature. I also discuss examples which motivate the domains covered in this work and the impact of incrementality on the SDS pipeline.

2.2.1 Humans Process Language Incrementally

It is true that humans do not wait for the end of a sentence to process language. The information presented (as a word or sounds) at a given point in time is integrated with the information that has been already presented to build an understanding *incrementally*. Humans also generate language incrementally often planning sentences along the way. This mechanism of comprehending and generating language incrementally is performed seamlessly by humans (Marslen-Wilson (1973)). *Incremental* processing aids efficient language processing in humans by helping understanding and generation and thus saving time. We also continuously make predictions for sounds, words, form, syntax, and semantics while processing language (Tanenhaus et al. (1995), Sedivy et al. (1999)). Recent evidence indicates that we not only use the information presented but also predict the future context (words, syntax, and semantics) and integrate this knowledge in our processing to make sense of what is being spoken and generate relevant responses (Salverda et al. (2014), Maess et al. (2016), Pickering & Gambi (2018)). Incremental processing of language is not trivial but rather involves complex planning and optimization.

Incremental processing is particularly useful when holding conversations. It allows humans to achieve fluid turn-taking, provide acknowledgments, generate interruptions, generate back-channels and other non-verbal cues, provide concurrent feedback (uh-huh, yeah), generate collaborative utterance completion, etc. midway through an utterance. This behavior helps achieve a state of mutual understanding (grounding) faster, among many other advantages. Incremental behavior also enables humans to interrupt and ask questions when in doubt and provide real-time feedback (head-nods, back-channels, etc.) when following the information being presented.

2.2.2 Incremental Processing in SDS

One compelling high-level motivation for systems developers to incorporate such incremental processing into their systems is in order to reduce system response latency (Skantze & Schlangen (2009)). As the incremental systems by design process the user utterance at units smaller than an utterance of a typical user turn, they possess the advantage of understanding and responding at a faster rate. Figure 2.6 shows the operation mechanism of incremental SDS and non-incremental SDS.

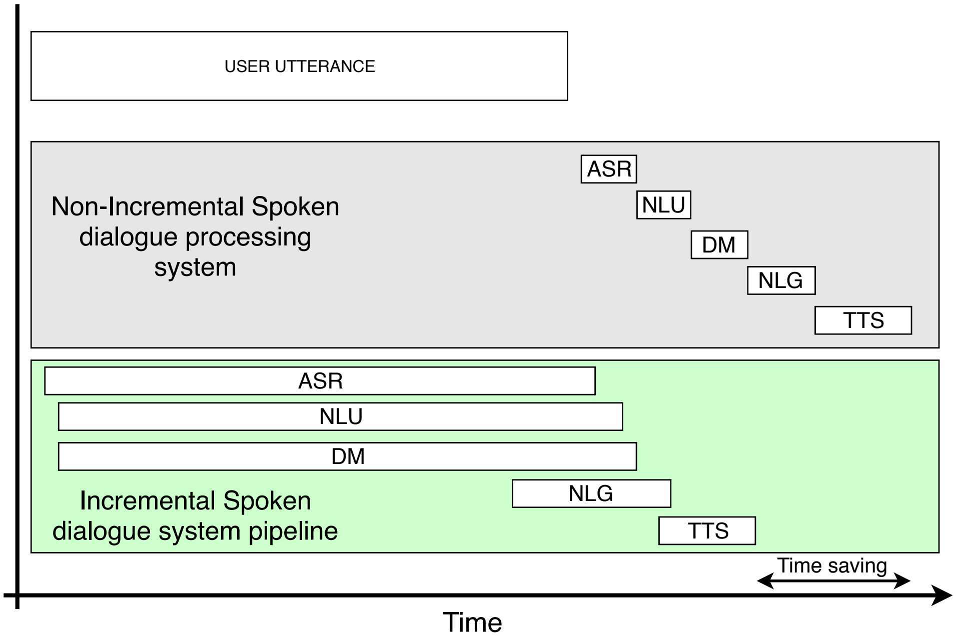

USER UTTERANCE
Non-Incremental Spoken dialogue processing system
ASR
NLU
DM
NLG
TTS
ASR
NLU
DM
NLG
TTS
Incremental Spoken dialogue system pipeline
Time saving
Time

Figure 2.6: The latency reduction advantage of using an incremental processing pipeline vs. a non-incremental pipeline. The time savings achieved by the SDS operation are shown in the figure. In a non-incremental SDS the ASR, NLU, DM, NLG, and TTS modules operate in a pipeline manner. This results in increased processing time due to the non-parallel operation. In a more incremental SDS the modules begin processing as soon as the inputs are available thus reducing the response time of the SDS. This comes at a cost of high performance processing overhead required to achieve this functionality.

Recent studies have also demonstrated user preference of incremental systems over their non-incremental counterparts (Skantze & Schlangen (2009); Aist et al. (2007a)), shown positive effects of incrementality on user ratings of system efficiency and politeness (Skantze & Hjalmarsson (2010)), and even shown increases in the fluency of user speech when appropriate incremental feedback is provided (Gratch et al. (2006)). Incremental systems have also been shown to be perceived as comparatively more intelligent (de Kok et al. (2015)) compared to their non-incremental counterparts.

Equipping SDS with such incremental capabilities is very challenging and has been an active topic of research (Aist et al. (2007a); Selfridge et al. (2013); Hastie et al. (2013); Baumann & Schlangen (2013); Buß & Schlangen (2010); Dethlefs et al. (2012); Selfridge et al. (2012); DeVault et al. (2011b); Skantze & Schlangen (2009); Schlangen et al. (2009)). Incremental processing in SDS differs from the typical turn-based processing on multiple levels. The functions performed by an incremental SDS happen at a smaller time scale compared to a non-incremental turn-based SDS. This means that the mechanism of operation in the SDS needs to undergo modifications, which can be challenging. I will discuss i) architecture choices involved in the development of an incremental SDS, ii) the choices and sub-problems involved in the development of the NLU, and iii) the challenges in the development of the dialogue manager/policy and the NLG.

Architecture Choices

The changes required in the architecture of an SDS are a necessary consideration. The premise for the changes in incremental SDS are:

i) The system needs to start processing the utterance as soon as the users begin their speech. The ASR needs to perform online real-time incremental decoding of the speech input, i.e., process the utterances in chunks without dependence on the future speech signals to produce the text output. The NLU (parser, classifier) needs to begin processing the user utterance before the user completes the sentence. The NLU needs to generate the output using partial hypotheses from the ASR (called partials). This can often be difficult to achieve due to the instability of the ASR output while performing online decoding. For instance, the incremental ASR hypotheses for the words 'green dog' in a sequence are, 'grey', 'grey in', 'green dog'.

These hypotheses are generated every few milliseconds. Such instabilities have to be accounted for by an SDS.

ii) Incremental systems should plan and begin generation of the output, often before the user has completed the utterance when the confidence is sufficient. If the new information requires changes to the currently being spoken utterance, mechanisms for taking back the information that has been uttered (such as repairs, self-interrupt) need to be supported by the architecture. In multimodal SDS, integration of information from different modalities needs to be supported by the architecture.

Allen et al. (2001) propose an architecture that supports i) and ii) as shown in Figure 2.7. In this architecture, the interpretation manager, behavioral agent, and the generation manager operate asynchronously. The interpretation manager takes the speech input and generates speech acts as the output, which are utilized by the behavioral agent to generate the appropriate behaviors. The generation manager plans the responses that the system should generate. The discourse context performs the job of providing the generation manager with the speech acts from the user, which can be utilized to convert to text output. This architecture, however, does not explicitly provide the mechanism for information composition incrementally using the past utterances. However, utilizing an Information State Update model (as in Matheson et al. (2000)) could help achieve this goal.

Schlangen & Skantze (2011) provide a detailed buffer-based incremental architecture. In this general abstract model, the SDS is viewed as a network of processing modules with information flowing between these modules. Each of these processing modules is composed of a processor and two buffers (a left one and a right one). The processor receives input from the left buffer, processes it, and then forwards the output to the right buffer. The next step is to forward the contents of the right

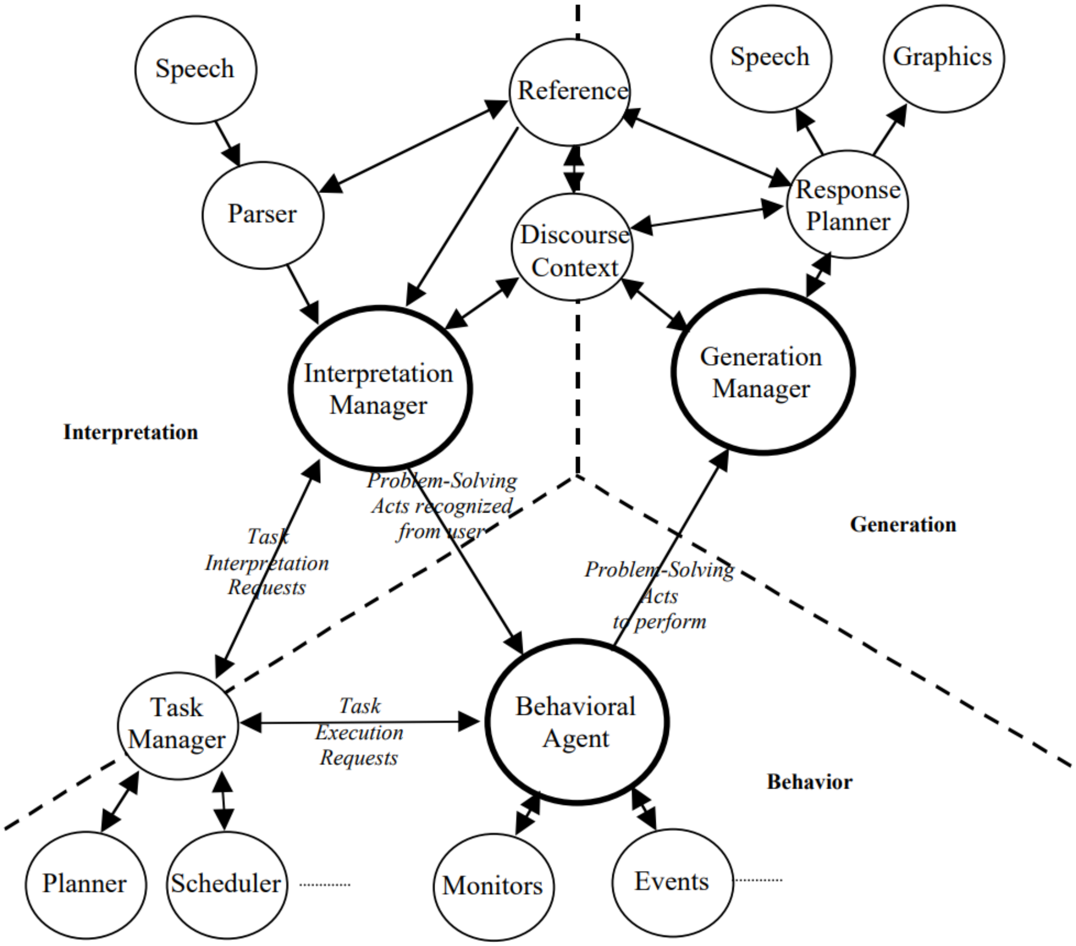


Figure 2.7: The architecture of Allen et al. (2001) for an SDS that supports incremental processing.

buffer to the left buffer of the next module. Skantze & Hjalmarsson (2010) provide an implementation of the model in an SDS.

In both of these models proposed for incremental processing, the typical SDS pipeline needs to be redesigned completely. Rather than completely redesigning their architectures, system builders may be able to gain some of the advantages of incrementality, such as reduced response latencies, by incorporating incremental processing in select system modules such as ASR or NLU. Such an architecture

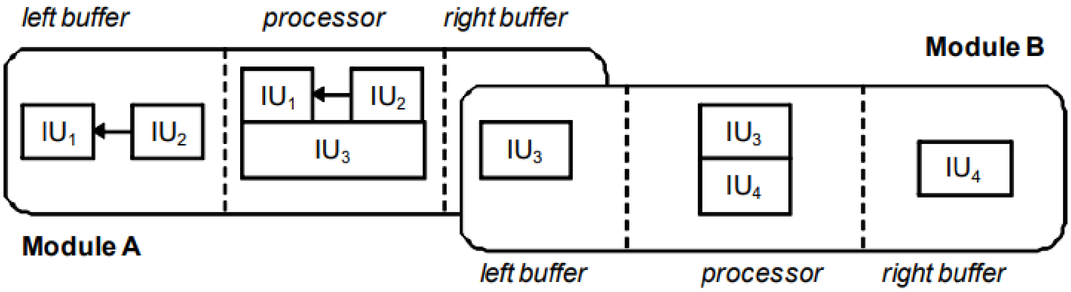


Figure 2.8: The architecture of Skantze & Hjalmarsson (2010) for an SDS that supports incremental processing using the buffer paradigm (Schlangen & Skantze (2011)).

utilizes the standard SDS pipeline and incorporates incremental processing capabilities in the SDS components. In Chapter 3, we describe in detail the architecture followed in this work, which supports the development of low latency SDS without redesigning the SDS pipeline completely.

Now I will discuss works that have studied various incremental language processing phenomena and tackle the sub-problems involved.

Incremental Language Processing

An incremental SDS needs to process user speech at a sub-utterance level, including the language understanding module, the dialogue manager, and the language generation module. The aspects of incrementality that have been studied in SDS[5] are

- Natural language understanding:
  - parsing user utterances into a semantic form (e.g., DeVault & Stone (2003); Kruijff et al. (2007); DeVault et al. (2011b); Eshghi & Lemon (2014))

[5] Does not include the ASR and TTS modules as they are not the main focus of this work.

  - disfluency detection (e.g., Hough et al. (2015))
  - segmentation and labeling (e.g., Nakano et al. (1999); Petukhova & Bunt (2011); Hough & Schlangen (2017))
  - grounding (e.g., Visser et al. (2014))
- Dialogue management & generation:
  - turn-taking (e.g., Skantze & Hjalmarsson (2010); DeVault et al. (2011a); Selfridge et al. (2013); Kousidis et al. (2014); Ghigi et al. (2014); Khouzaimi et al. (2016))
  - natural language generation (e.g., Kelleher & Kruijff (2006); Skantze & Hjalmarsson (2010); Rieser et al. (2014))

Now I will discuss these in detail.

Natural Language Understanding

The incremental NLU module needs to be capable of generating the output representation using the partial text hypotheses from the ASR. Many approaches have been developed to support this functionality. I discuss two such popular approaches for developing incremental NLU.

Parsing & semantic representation. In incremental SDS, as in DeVault et al. (2011a), the ASR partial inputs are used to predict the AVM (attribute-value-matrix) representation of the user utterance. The matrix contains a set of attributes and values and the confidence scores assigned to this matrix, which are used by the DM to take a specific action. The DM utilizes such a matrix input to generate the next action.

Another method for representing the user input is using dialogue acts. For a long time, dialogue acts (e.g., Bunt (2009); Bunt et al. (2010)) have been an important tool for understanding user utterances in SDS. They have also been another important mode of representing user intentions in incremental SDS. Incrementally detecting dialogue acts in the user utterance has been studied in Petukhova & Bunt (2011). In SDS, such dialogue acts are often accompanied by intents and slot-values Williams (2011) which the DM utilizes to take the next action.

TTR (Type Theory with Records) has also been studied in recent times in the context of incremental dialogue processing (Hough & Purver (2014); Eshghi & Lemon (2014); Zarrieß et al. (2017)). However, no single best representation for NLU in SDS can be claimed.

Segmentation and dialogue act labeling. To allow users to speak naturally to SDS, the NLU module needs to be operating incrementally. It has been understood for some time that this ultimately requires a system to be able to automatically segment the user's speech into meaningful units in real-time while the user speaks (Nakano et al. (1999)). Still, most current systems use relatively limited and straightforward approaches to this segmentation problem. For example, in many systems, it is assumed that pauses in the user's speech can be used to determine the segmentation, often by treating each detected pause as indicating a dialogue act (DA) boundary (Komatani et al. (2015)). In such systems, each inter-pausal unit (IPU) or *speech segment* identified by a Voice Activity Detection (VAD) component is fed to an understanding component for interpretation.

While easily implemented, such a pause-based design has several problems. First, a substantial number of spoken dialogue acts contain internal pauses (Bell et al. (2001); Komatani et al. (2015)), as in *I need a car in... 10 minutes.* Using simple pause length thresholds to join certain speech segments together for interpretation is not a very effective remedy for this problem (Nakano et al. (1999); Ferrer et al. (2003)). More sophisticated approaches train algorithms to join speech across pauses (Komatani et al. (2015)) or decide which pauses constitute ends of an utterance that should trigger interpretation (e.g., Raux & Eskenazi (2008); Ferrer et al. (2003)). This addresses the problem of DA-internal pauses, but it does not address the second problem with pause-based designs, which is that it is also common for a continuous segment of user speech to include multiple DAs *without* intervening pauses, as in *Sure that's fine can you call when you get to the gate?* The third problem is that waiting for a pause to occur before interpreting earlier speech may increase latency and erode the user experience (Skantze & Schlangen (2009)). Together, these problems suggest the need for an incremental dialogue act segmentation capability in which a continuous stream of captured user speech, including the intermittent pauses therein, is incrementally segmented into appropriate DA units for interpretation.

Dialogue management & natural language generation

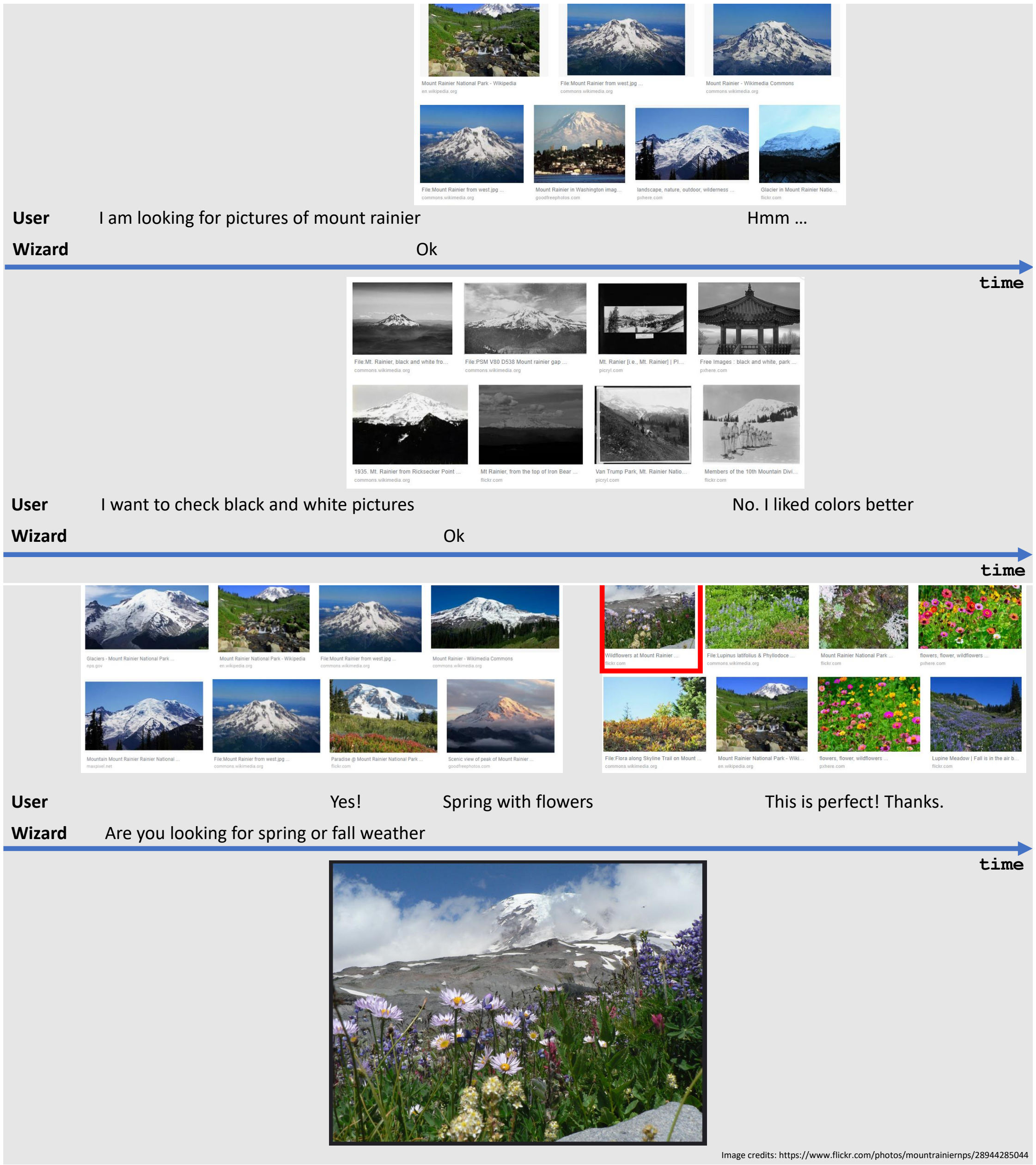
The DM in incremental SDS needs to play an active role in deciding when to intervene in the dialogue. Since the SDS can generate real-time interruptions and feedback, designing the DM and NLG is a challenging task. The DM needs to actively decide when to intervene in the dialogue without compromising on the quality of the dialogue. An overly eager DM interrupting the user is undesirable in an SDS and so is a DM which is too slow to respond (much like the present day non-incremental SDS). This is because over-generating utterances can be annoying. In DeVault et al. (2011a), the DM is a classifier which actively decides when to intervene. The classifier utilizes the confidence scores in the NLU frames along with other information from the NLU to make a decision. Reinforcement Learning (RL) is another method that has been used to improve the responsiveness of the SDS and increase task success (Selfridge & Heeman (2010); Dethlefs et al. (2012); Khouzaimi et al. (2015) Dethlefs et al. (2016); Kim et al. (2014)) in the DM and NLG modules. For example, RL has been used to learn policies about when the system should interrupt the user (barge-in), stay silent, or generate back-channels. Reinforcement learning has shown promise for DM and will be actively used in this work along with the classifiers, which will be covered in the later chapters.

2.2.3 Motivating Examples

In this section, we look at a few hypothetical examples highlighting the importance of incrementality, and its impact on the SDS processing pipeline. I also introduce an alternative SDS pipeline that accommodates building such applications. Through these examples, I motivate the domains covered in this work and highlight the important research questions that will be the focus of this work.

Example 1: Conversational image search

First we look at a hypothetical incremental conversational image search example. A user converses with a hypothetical image search SDS which collaborates with the user to retrieve the relevant images. Speech processing is faster than traditional typing (Ruan et al. (2016)), and there are a growing number of users who use voice for search (Schalkwyk et al. (2010)). Such voice-based interfaces are more prevalent in mobile devices. Such a conversational image search system can also find application in medical domains (Kalpathy-Cramer et al. (2011)). With additional incremental processing by the conversational SDS, the system can generate quicker



User I am looking for pictures of mount rainier

Wizard Ok

User Hmm …

User I want to check black and white pictures

Wizard Ok

User No. I liked colors better

Wizard Are you looking for spring or fall weather

User Yes!

User Spring with flowers

User This is perfect! Thanks.

Image credits: https://www.flickr.com/photos/mountrainiernps/28944285044

Figure 2.9: A hypothetical conversation between a user and an SDS which can perform conversational image search. In this example the user requests pictures of Mount Rainier. The user and the SDS converse until an image has been found.

responses, provide feedback, and even help users achieve the goal of retrieving the correct image faster.

In this example (shown in Figure 2.9) the user is searching for an image. The user begins the conversation by issuing an image search request (“I am looking for pictures of mount rainier”). The SDS responds and displays the pictures using an image retrieval engine. The user takes some time to look at the results and modifies the query (“I want to check black and white pictures”), which yields different image results. The user, upon checking the images, reverts the decision and intends to have the previous set of images back. At this point, the SDS decides to assist the user by issuing a disjunctive question (“Are you looking for spring or fall weather?”) providing the options of selecting either Spring or Fall weather pictures. The user likes the recommendation and clarifies before the system has finished speaking that they want to see the spring pictures of Mount Rainier with flowers (“Spring with flowers”). The system processes the request and displays the new set of pictures. The user, at this point, likes the first displayed picture and ends the dialogue exchange by thanking the SDS.

There are advantages in dialogue. The benefit of using dialogue in such a scenario is that the system can not only track the user’s past search for easier retrieval (“No, I liked colors better”) in the future but also recommend the search results relevant to the current results (‘Spring or Fall’). This type of system is usually implemented with a state tracking module in the SDS pipeline.

Advantages of incrementality. The benefit of incremental processing in this example is that the user is shown the relevant images more quickly, as soon as their preference is recognized by the system (e.g., “I want to check black and white pictures”, “Spring with flowers”). Figure 2.10 shows the advantage of processing the user query incrementally where the system can display the pictures to the user faster. Along with the quicker retrieval, the user satisfaction could also be much higher as observed with incremental processing for other domains. It is also important to understand the semantics of the user utterances (e.g., no answers, image query, back-channels, etc.). For instance, in the example, when the user responds "No. I liked colors better", the intent was to revert the search to the old images. Also, note that the same utterance is made up of two different intents. The incremental model can identify the intents quickly before the user finishes their utterance.

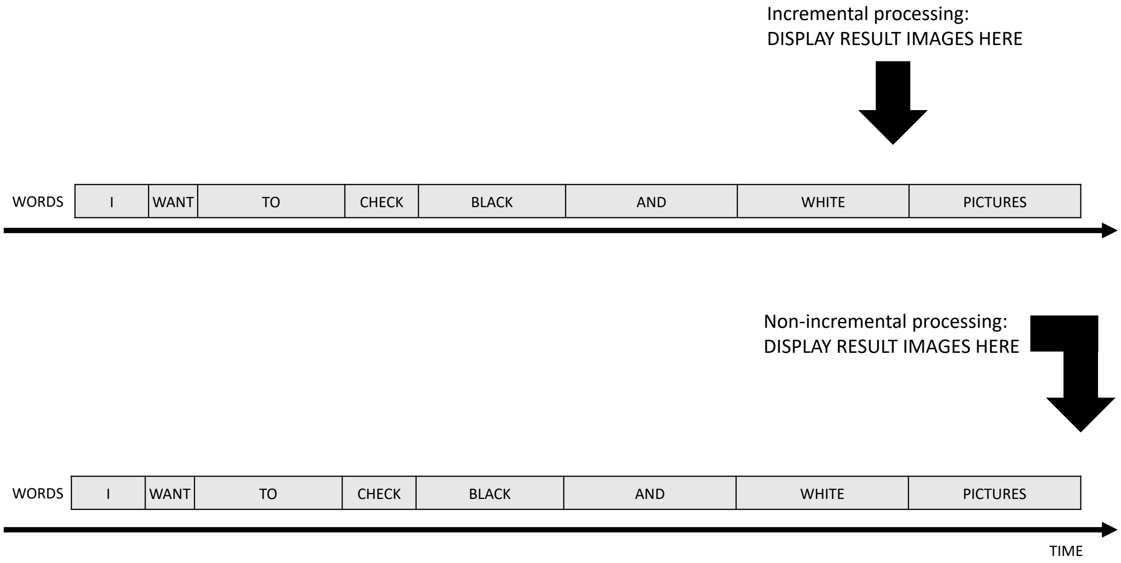


Figure 2.10: Time saving in incremental dialogue processing (Example 1).

We will use conversation image search as motivation in Rapid Dialogue Game (RDG) domains and as a testbed for incremental processing and other sub-problems.

Example 2: Conversational image editing

In the previous example, we introduced a scenario showing the advantage of incremental image search. In this example, we look at another case called conversational image editing, which involves more diverse and complex semantics.

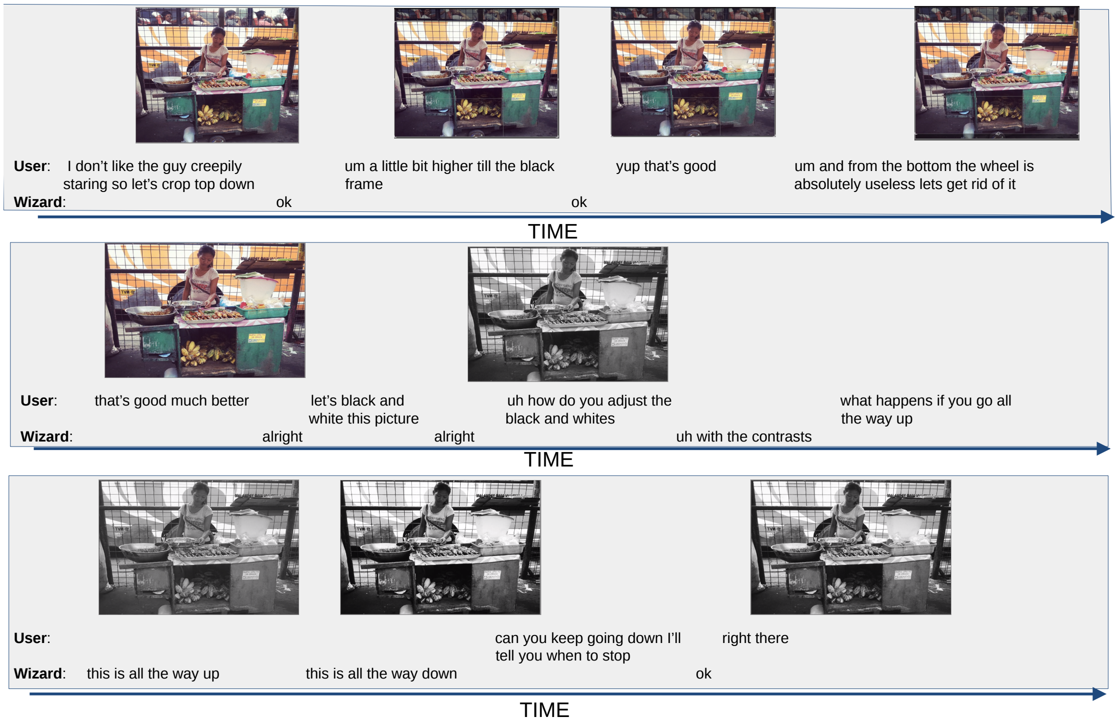
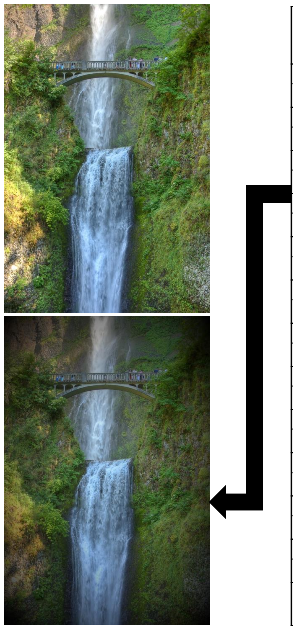

User: I don't like the guy creepily staring so let's crop top down
um a little bit higher till the black frame
yup that's good
um and from the bottom the wheel is absolutely useless lets get rid of it
Wizard: ok
ok
TIME
User: that's good much better
let's black and white this picture
uh how do you adjust the black and whites
what happens if you go all the way up
Wizard: alright
alright
uh with the contrasts
TIME
User: can you keep going down I'll tell you when to stop
right there
Wizard: this is all the way up
this is all the way down
ok
TIME

Figure 2.11: In this example the user and the SDS collaborate to perform the task of image editing. The user requests edits and the SDS interprets and understands the requests, and performs the edits on the image.

In this example, the user converses with the system to collaboratively perform the task of image editing. Image editing using tools such as Adobe Photoshop, Lightroom, Gimp, etc. has become popular in recent times. Image editing is a difficult task requiring a steep learning curve. The users can take assistance from a smart SDS to perform the task of image editing collaboratively. In this example, we imagine an SDS performing the task of conversational image editing taking commands from the user, recommending to the user what changes to perform, and assisting the user to achieve the best edits. The image editing task is very incremental as the users perform various small edits and often revert and update the edits using short utterances. However, the semantics of user requests are complex, which requires fine-grained understanding models.



		Non-Incremental		Incremental	
		Intent	Parameter	Intent	Parameter
t_0	add				
t_1	a			IMAGE_EDIT	ADJUST
t_2	vignette				VIGNETTE
t_3	to				
t_4	highlight				
t_5	better				
t_6		IMAGE_EDIT	ADJUST, VIGNETTE		
t_7	I				
t_8	think				
t_9	that's				
t_{10}	good			LIKE_EDIT	
t_{11}	enough				
		LIKE_EDIT			

Figure 2.12: Time saving in incremental dialogue processing (Example 2).

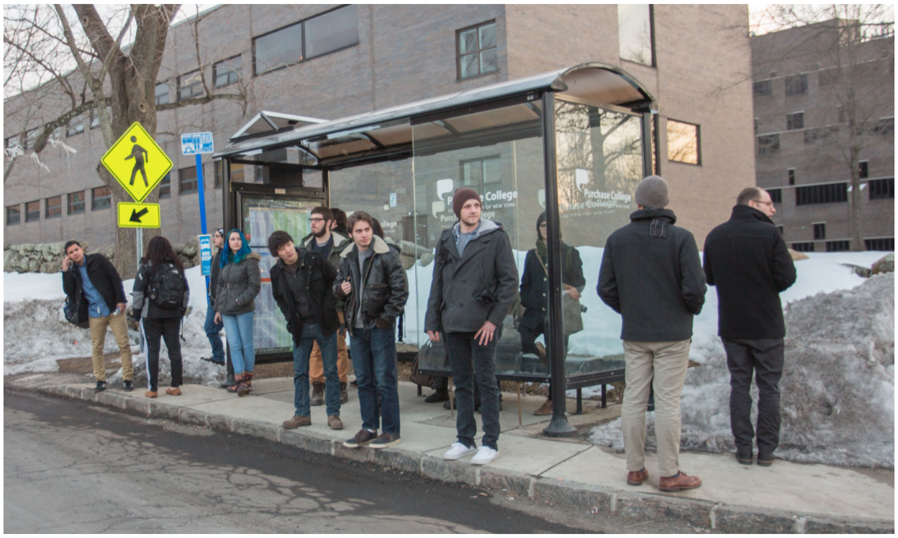
Figure 2.11 shows an example conversation between a user and a hypothetical SDS performing the image editing. In the first utterance, “I don’t like the guy creepily staring so let’s crop top down”, the first part of the utterance, “I don’t like the guy creepily staring” provides the user motivation to crop the photo to a point until which the cropping needs to be done. The action that the user wants to take is only known (cropping) upon hearing the utterance “let’s crop top down”. Upon observing the changes performed on the image by the SDS the user issues a command which provides additional information about the point up to which the edits (“um a little bit higher till the black frame”) need to be performed. When the user is happy with the edits, they issue the utterance “yup that’s good” which indicates that the user is happy with the edit. The conversation goes on with the user proceeding to perform other edits such as changing colors, contrasts, and modifying the edits incrementally. Incremental processing is especially important in the cases where the user keeps providing real-time feedback (“right there”, “a little more", etc.) as in this case the inability to perform incremental processing in the SDS will not only result in slow responses but incorrect edits. The diverse list of image editing commands needs to be understood by the system along with the list of attribute values (for example, command: cropping, attribute: position, from_top, until_blackframe). In this work (Chapter 6), we develop such an incremental language processing module and present an annotation schema, which helps achieve the task of conversational image editing.

Figure 2.12 shows the additional time saving that can be achieved using incremental processing. Performing real-time understanding helps save time and thus can achieve higher naturalness perception and better efficiency.

Example 3: Hypothetical autonomous car

In this example, I show how such a method can be applied to another real-world incremental SDS. Let's imagine a hypothetical conversation in a not so distant future with an autonomous driving taxi[6]. In this example, a human is on their way to work on a winter day. The human usually takes the bus to work. Today the bus is late. So, the human decides to summon an autonomous taxi using their favorite taxi-hailing app from the bus stop. The taxi is approaching the user, but there are a lot of people at the bus stop. The taxi is confused. The user also sees on their app that the taxi is near but is not sure which car it is. So, the taxi places a call to the human (referred to as Rider), and they have a conversation as shown in Figure 2.13. In this example, as in the other two examples, the conversation proceeds naturally, and it is critical to understand the user utterance and perform the action in real-time while performing semantic understanding.

[6] This example is about spoken dialogue and not about the mechanics or ethics of autonomous cars. The "taxi" in the example below is to be seen in the context of SDS.



ID	Speaker	Utterance
U1	Taxi:	Hi Rider. I am here. Which one are you?
U2	Rider:	Hi! I am the one with the black bag, blue shirt holding a phone. . . .
U3	Taxi:	Are you standing next to the sign post?
U4	Rider:	Yes.
U5	Taxi:	Cool. I see you.
U6:	Rider:	Awesome. Which one are you?
U7:	Taxi:	I am the white Prius with number plate THESIS2019.
U8	Rider:	Cool. I see you now.

Figure 2.13: A hypothetical conversation between an autonomous taxi and a human (Rider). '. . . ' means that the Rider is continuing to speak while being interrupted by the Taxi.

In the next chapter, we look at the problem of data collection and pre-processing that supports the building of an SDS.

Chapter 3

Crowd-Sourcing Dialogue Data Collection

> *"Data is a precious thing and will last longer than the systems themselves."*
>
> – Tim Berners-Lee, *Inventor of WWW*

Data collection and preparation is an integral step that precedes building an SDS. In this chapter, I discuss the data collection and preparation for building an SDS, and also show the methodology adopted in building corpora in support of developing an incremental SDS.

3.1 Introduction

The problem of data collection is often viewed as an engineering obstacle rather than a problem of research inquiry. While it is true that data collection involves overcoming numerous engineering and logistic hurdles, the process of data collection posits interesting research questions.

1. A typical crowd-sourcing task involves assigning labels to a given image or text, transcribing audio, performing a web search, and answering demographic questions. There are also chat-based crowd-sourcing tasks that involve users role-playing or playing a game. However, crowd-sourcing spoken dialogue data adds to the complexity of the task by requiring that users provide voice input through a

piece of audio recording equipment (microphone). The question that arises is ‘can we crowd-source dialogue data collection where the majority of users might not have high-quality audio recording equipment?’

2. It is also important to compare the quality of data collected from such remote participants recruited through crowd-sourcing to the traditional in-lab data collection approach. Users in crowd-sourcing environments generally do not have access to high-quality audio equipment. In lab environments typically audio recording microphones with high fidelity are present. While, it is clear that the audio collected in-lab from high-quality equipment is better, it is not often clear how much better the audio quality is compared to the crowd-collected audio data.

3. In traditional lab-based approaches, a research assistant is present to explain the rules and answer questions about the task to the study participant. In a crowd-based approach, this instruction channel is not present, resulting in reliance on users following the instructions presented at the beginning of the task. It is important also to study if the participants in crowd-sourcing environments follow the instructions and complete the task. ‘Do crowd workers perform better compared to in-lab participants?’ is also an important question that we address below.

4. Users in traditional lab approaches are paid remuneration for participation in the study being conducted. This includes compensation for travel and time of the participant. One of the appeals of the crowd-sourcing paradigm is the cost savings achieved by compensating users for just their time. The costs comparison of such an approach is not clear in the literature of dialogue systems research. So the question that arises is ‘how much savings can be achieved by conducting such a crowd-sourcing study compared to in-lab methods?’

In this chapter, we answer these questions showing the cost savings and tradeoffs incurred. It is also an engineering endeavor to build a toolkit supporting the process of spoken dialogue data collection.

Crowd-sourcing in dialogue data collection

In recent years, dialogue system researchers have been attracted to crowdsourcing approaches for a number of data collection tasks that support system training and evaluation. Some of the tasks that have been explored include:

- transcription (Parent & Eskenazi (2010))
- capture of speech and text for training language models (Liu et al. (2010))
- eliciting utterance texts that correspond to specific semantic forms (Wang et al. (2012))
- collecting text templates for generation (Mitchell et al. (2014))
- chat data between remote participants (Lasecki et al. (2013); Miller et al. (2017); Schlangen et al. (2018))
- collecting survey-style judgments about a dialogue system's performance (Yang et al. (2010))
- collecting interactive dialogues in which a single user interacts with a live dialogue system (Meena et al. (2014); Liu et al. (2010); Jiang et al. (2014))

In this chapter, I begin by discussing a web framework that supports crowdsourced collection of spoken dialogue interactions between two remote participants. To the best of our knowledge, this is the first time that crowd-sourcing has been applied to the collection of spoken dialogue interactions between two remote participants in support of dialogue systems research. Crowd-sourcing has been used before to collect text-based chat dialogues (not spoken dialogues) between remote participants (see e.g., Lasecki et al. (2013)). Such human-human dialogue data can be quite valuable, especially in the early stages of designing and building a dialogue system. Human-human dialogues provide examples of domain-specific language and interaction that can inform a range of architecture and design choices in system building, as well as serve as initial training data for system components (Lasecki et al. (2013)). The decision to collect spoken dialogues between human interlocutors online, rather than in a controlled lab setting, is a multi-factorial one. Some of the important considerations include the introduction of browser-mediated interaction, limitations in available modalities for interaction, potential changes in demographics, data quality considerations, and the introduction of communication latency. This chapter discusses the toolkit developed for the crowd-sourcing environment, data collection experimental results, and advantages and limitations of collecting data using the crowd-sourcing environment.

Using crowd-sourcing for collecting data between a human and an SDS is also important to study. In this chapter, I also address a critical bottleneck in the design and evaluation of SDS on the web for running crowd-sourcing studies: the availability and cost of collecting human dialogue data for a new domain. When designing, training, or testing a new dialogue system, the collection of in-domain dialogue data, between a human user and a system prototype (human-agent) is important. In-domain dialogue data is important because it provides examples of domain-specific language and interaction that serve to highlight important semantic and pragmatic phenomena in the domain, inform system design choices, and also serve as training data for system components such as ASR, NLU, and NLG (Lasecki et al. (2013)). This chapter also discusses the development of an SDS for a crowd-sourcing environment. Thus, this is the first work that develops a toolkit for collecting crowd-sourced spoken dialogue data in support of development of an incremental SDS in both human-human and human-agent conditions.

How crowd-sourcing can support the development of SDS components

Figure 3.1 shows the components of a typical SDS. It shows how crowd-sourcing can be used to build corpora for training the components of the SDS pipeline. The speech data collected is transcribed and can be used to train the ASR component. The transcriptions are annotated with labels (e.g., dialogue acts, intents, slot-values, etc.) and used to train the NLU module. In the case of multimodal SDS, annotations for visual content can also be crowd-sourced. The interactions collected are used to train the DM/policy to learn dialogue management behaviors. The NLG module can be trained on the transcriptions of the human playing the role of the agent in the human-human data. The TTS can also be trained on the human speech collected through crowd-sourcing, although for building TTS it is preferable to use clean audio, which depending on equipment may be hard to do via crowd-sourcing (Georgila et al. (2012)).

The outline of this chapter is as follows: i) In Section 3.2, I discuss the domain used for collecting data using crowd-sourcing. ii) In Section 3.3, I discuss the software architecture of the system developed for collecting spoken conversations in support of building incremental SDS. iii) In Section 3.4, I discuss the experiments conducted for collecting data using crowd-sourcing. iv) In Section 3.5, I discuss the results of comparing the data collected in the lab vs. the data collected using crowd-sourcing. v) In Section 3.6, I discuss the extension of the framework for collecting data using a crowd-sourced agent. I also discuss the data collected

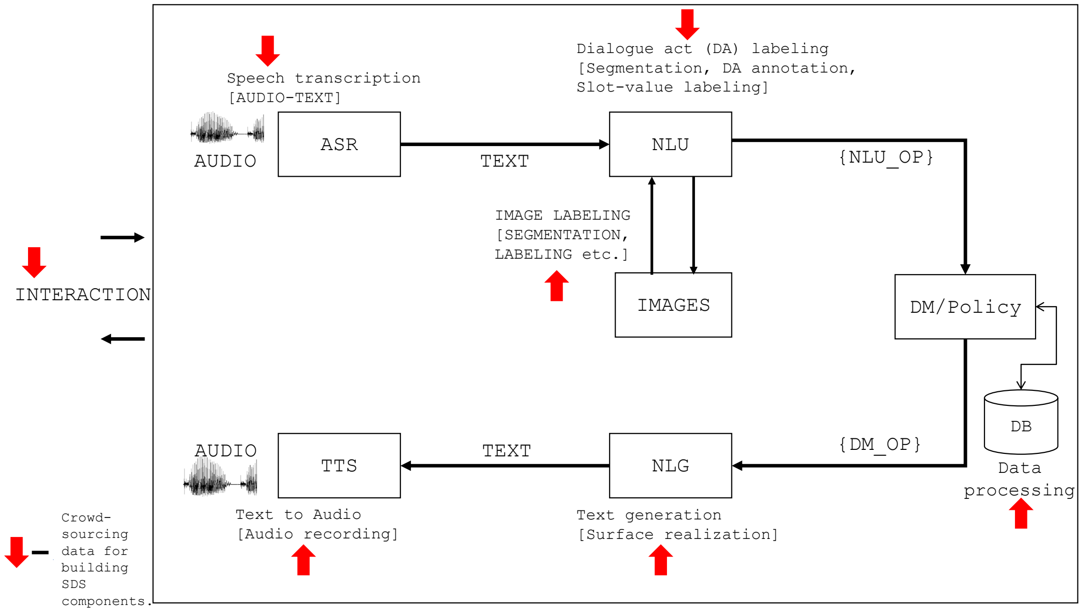


Figure 3.1: The SDS components with the data required for training and evaluating each component.

using the incremental SDS agent and observations from running the study using crowd-sourcing. vi) Finally in Section 3.7, I discuss briefly the transcriptions and annotations performed in this work.

3.2 Rapid Dialogue Game-Image (RDG-Image)

The research that motivates our crowd-sourced data collection involves a fast-paced spoken dialogue game, in which interlocutors describe images to each other. An example interaction, drawn from the lab-based Rapid Dialogue Game (RDG) corpus previously collected (Paetzel et al. (2014)), is shown in Figure 3.2.

In this excerpt, one player (the Director) tries to describe the image depicted with a red border to the other player (the Matcher), who sees the same array of eight images on their screen but with their locations shuffled. The players are under substantial time pressure to complete as many images as they can within a fixed

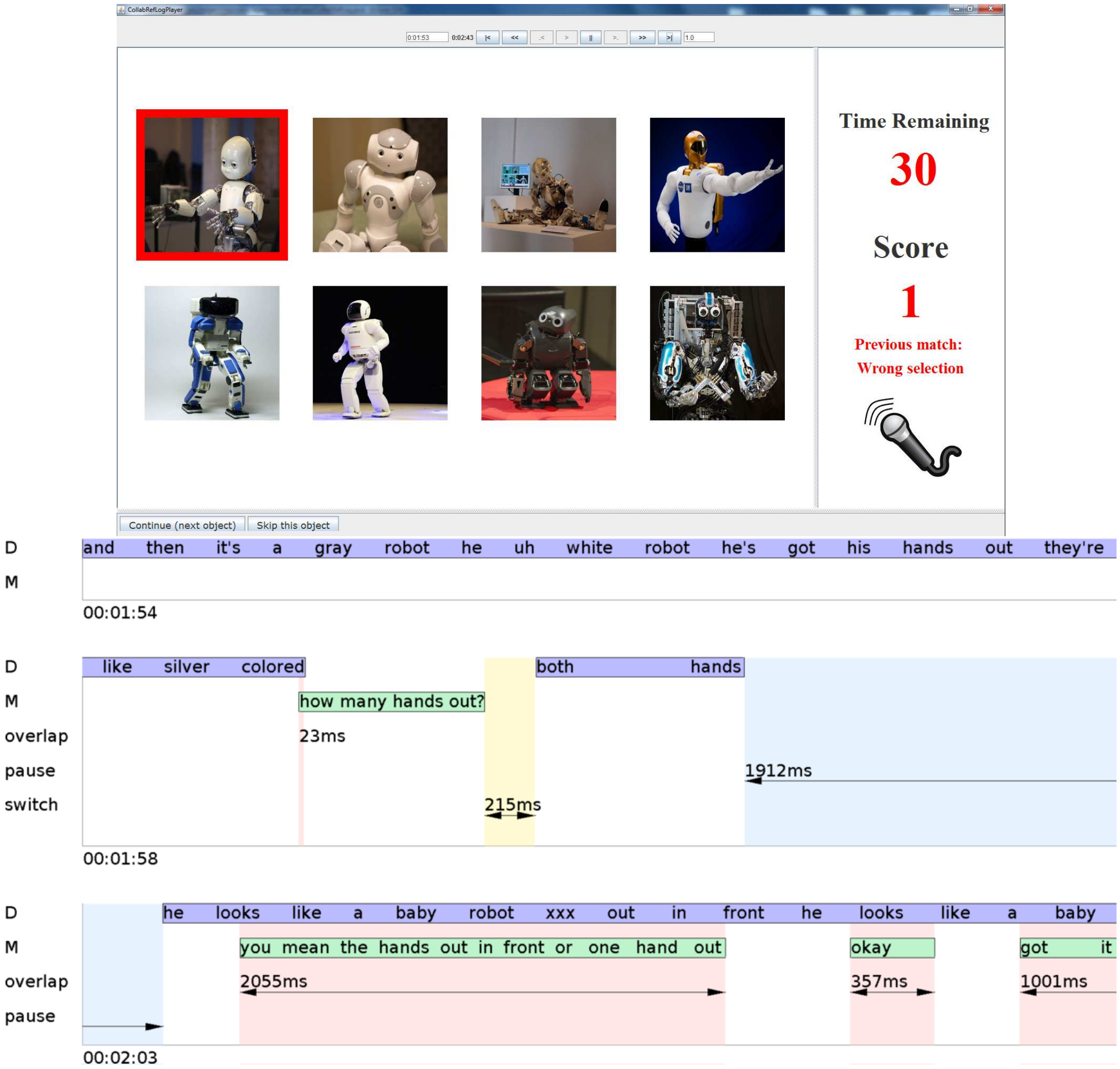


Figure 3.2: An excerpt of human-human gameplay from our lab corpus. Segments of participant speech are arranged to show their temporal extents, with time increasing from left-to-right and top-to-bottom. Speech is segmented at all silent pauses exceeding 300 milliseconds. Periods of overlapping speech are shaded in pink. Periods containing silent pauses by a single continuing speaker are shaded in light blue. Periods of silence at speaker switches are shaded in yellow.

time limit. Natural spoken dialogue in this domain includes frequent overlapping speech (shaded in pink), low latency turn-taking (as when the Matcher asks *how many hands out?* and receives the answer *both hands* 215 milliseconds later), mid-utterance repairs, interruptions, acknowledgments, and other low-latency responses.

Capturing such rapid spoken exchanges over the internet presents a unique challenge. Particularly important factors for our dialogue system research, which aims to replicate these rapid dialogue skills in dialogue systems, include the quality of captured audio, the effect of communication latency on the interaction, the ability to collect examples of excellent gameplay, and the naturalness of the interaction and turn-taking. In addition to describing our web framework, this chapter presents a case study of how these factors differ between the lab-based corpus we previously collected and the crowd-sourced corpus we have collected with the new framework.

3.3 PMU System Architecture

In the dialogue research community, several researchers have recently taken steps toward collecting dialogue data from systems deployed on the web. Jiang et al. (2014) describe an architecture for capturing typed dialogue interactions in a human-agent configuration, with user speech optionally recognized by Google's cloud-based ASR service. Meena et al. (2014) have also been attracted to crowd-sourcing as a potential source of data, and reported a small-scale experiment in this direction. Some research applications such as Let's Go (Raux et al. (2005)) as well as commercial applications (Suendermann et al. (2011); Pieraccini et al. (2009)) have collected telephone-based dialogue data from large user populations. In recent times, systems for collecting data between remote participants, such as HALEF (Suendermann-Oeft et al. (2015)) and ParlAI (Miller et al. (2017)), have been developed. HALEF collects spoken dialogue data between remote participants using telephony and HTML5, while ParlAI (as of June 2019) does not support collection of spoken dialogue data. No such toolkits exist that support collection of

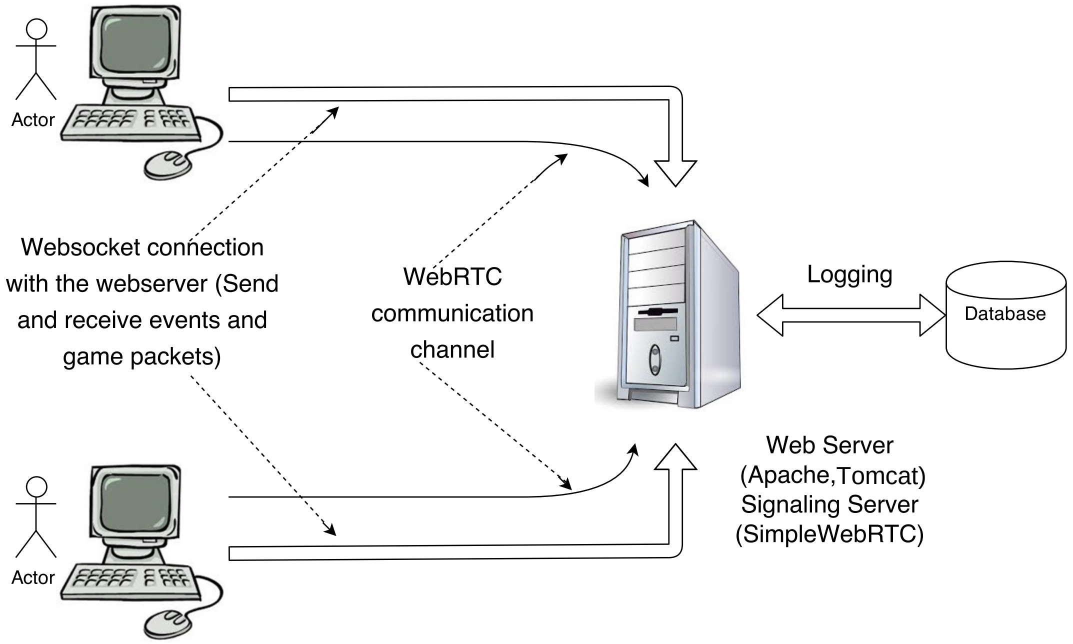

Actor
Websocket connection with the webserver (Send and receive events and game packets)
WebRTC communication channel
Logging
Database
Web Server (Apache,Tomcat) Signaling Server (SimpleWebRTC)
Actor

Figure 3.3: Pair Me Up (PMU) architecture in human-human mode.

conversational data between remote participants in support of building incremental SDS.

HTML5 advancements and wide-scale browser compatibility have enabled the collection of spoken interaction data between remote participants in recent times. This enables the development of real-time conversational interfaces on the browser, which earlier required the usage of proprietary tools such as Skype. Deploying data collection systems on the web, and using crowd-sourcing to recruit remote participants, offers the possibility of increasing the availability of participants while simultaneously driving down the costs of data acquisition. We will now discuss the architecture of the system used to collect the human-human interaction data over the web. This system developed for collecting the human-human interaction data will be referred to as Pair Me Up (PMU).

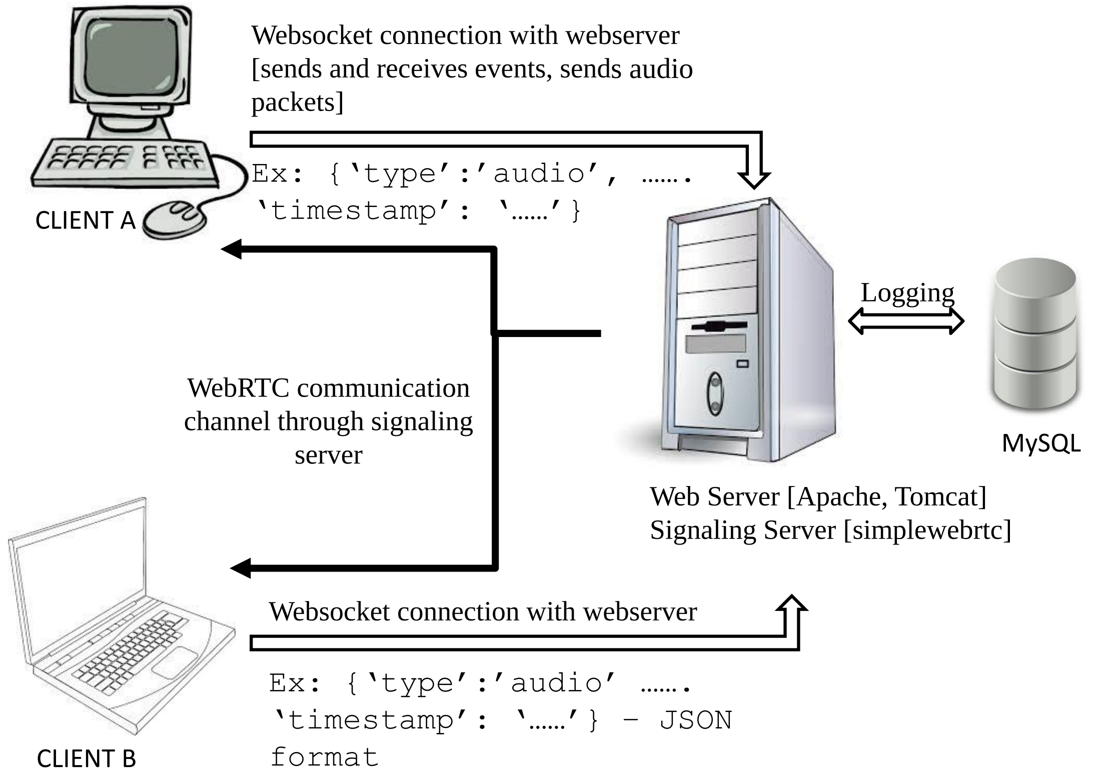

Websocket connection with webserver
[sends and receives events, sends audio packets]
Ex: {'type':'audio', …….
'timestamp': '……'}
CLIENT A
Logging
MySQL
WebRTC communication channel through signaling server
Web Server [Apache, Tomcat]
Signaling Server [simplewebrtc]
Websocket connection with webserver
Ex: {'type':'audio' …….
'timestamp': '……'} - JSON format
CLIENT B

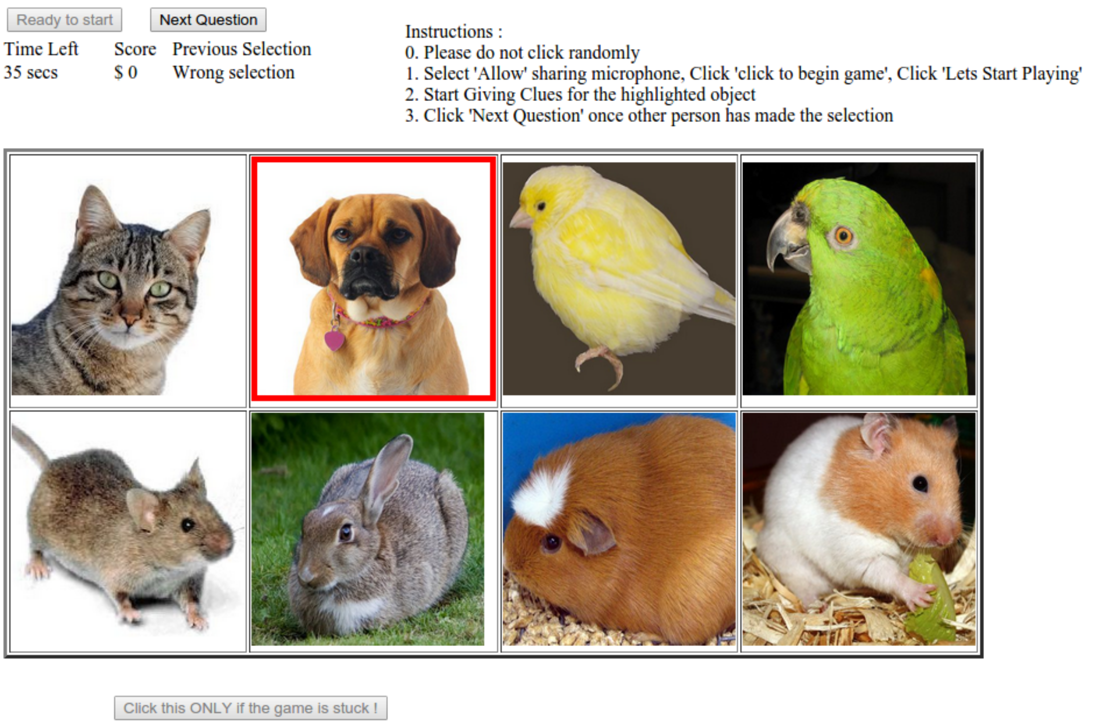
Figure 3.4: Architecture of the Pair Me Up system.

The PMU architecture for human-human data collection is shown in Figure 3.3. The system pairs two web users together and connects them into a shared game session where they can converse freely and interact through their browsers. PMU leverages recent developments in web technologies that support the development of web-based dialogue systems. It shares this approach with recent dialogue systems research, which makes use of emerging web technologies to enable spoken interaction between a remote web user and an automated dialogue system. The framework is designed to support additional games and tasks that can be embedded within a browser. Also, the framework is designed to enable the capture of spoken human-agent dialogues as well as human-human dialogues.

In PMU, several HTML5 web technologies are used to build an interactive game where the servers can initiate events on remote client browsers. Audio is streamed between two remote client browsers, and audio is captured to a server database. Two core technologies the system makes use of are WebSockets and WebRTC. WebSockets enable two-way communication between the client and server, and they specifically enable the server to push events such as image set changes to the clients, and the clients to send audio and game events such as button clicks to the server, without loading a separate URL. The streaming audio communication between the remote clients uses a separate SimpleWebRTC (http://simplewebrtc.com/) channel. SimpleWebRTC utilizes a signaling server process which is hosted on our servers along with our web server (Apache Tomcat). The video channel is disabled for the current study due to bandwidth limitations observed in pilot testing and the fact that RDG-Image players primarily look at the images being described rather than each other. Also, privacy considerations arise when the video is captured. NIST guidelines for Personally Identifiable Information (McCallister et al. (2010)) view video and IP addresses as personally identifiable information, and we avoided collecting this information.

Latency measurement protocol and data synchronization

In a lab-based study, network latency between machines can be minimized through the use of high-speed LAN connections, and computer clocks can be synchronized using a method such as the Network Time Protocol (Mills et al. (2010)). In a crowd-sourced data collection, network latency may be both higher and also harder to control. Additionally, security considerations rule out adjusting a remote user's system clock.



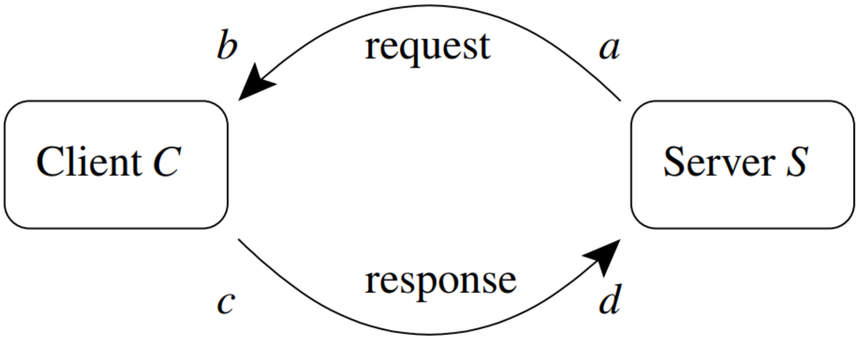
Time Left	Score	Previous Selection
35 secs	$ 0	Wrong selection

Instructions :
0. Please do not click randomly
1. Select 'Allow' sharing microphone, Click 'click to begin game', Click 'Lets Start Playing'
2. Start Giving Clues for the highlighted object
3. Click 'Next Question' once other person has made the selection

Figure 3.5: The web interface of the game.

In our web-based game interface, latency can potentially affect the data we collect in several ways. For example, there can be latency between when a remote user initiates an action in their UI (User Interface) and when the server learns that the action occurred. Conversely, if the server initiates an event in the user's UI (e.g., changing the image set), this event may not actually occur in the user's UI until some time later. Given the sensitivity of our research to having accurate timing of dialogue events, we implemented a simple latency measurement and synchronization protocol that allows us to (1) estimate the network latency between each client and the server, and (2) translate between timestamps collected from client machine system clocks and timestamps on our server.

Like the Network Time Protocol, our approach relies on the transmission of a series of request/response packets between the server and client machines. The



Event	Known time of event on S	Known time of event on C
a : Server S sends request	t_S^a	
b : Client C receives request		t_C^b
c : Client C sends response		t_C^c
d : Server S receives response	t_S^d	

Figure 3.6: Latency measurement protocol.

protocol is illustrated in Figure 3.6. At the beginning of each image set in the game, a request packet is sent from the server S to the remote client C. We denote the server's timestamp when this request is sent by t_S^a, using a subscript for the machine (S or C) whose clock generates the timestamp, and a superscript (a, b, c, or d) for the four sequential events that occur as the request and response are sent and received by each machine. As part of the exchange, Pair Me Up code running in client C's browser computes a client system timestamp t_C^c and immediately sends this value with its response back to the server. The server receives the response at t_S^d. With each request/response cycle, the server therefore has a measure of the round trip latency of server-client communication: $\text{roundtrip} = t_S^d - t_S^a$. Over the course of a game this request/response cycle happens in the background many times between the server and each of the remote clients. Figure 3.7 is an example of these round-trip measurements for a typical crowd user in our data set. In order to relate client event timestamps to server event timestamps, we make the assumption that

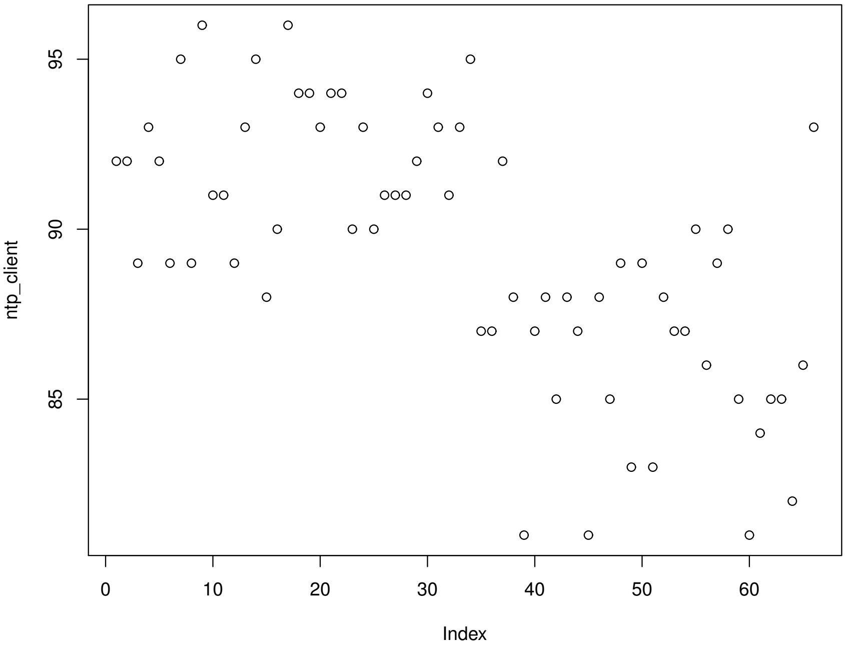


Figure 3.7: The round-trip time (in milliseconds) of query/response exchanges for one user. The round-trip time of a packet between the server and client is not constant. The x-axis shows the index of the packets labeled in the order of time. The y-axis shows the time taken by the packet for round-trip from the server to client and back to server.

the client initiated its response at the midpoint of the server's observed round-trip time:

$$t_S^c = \frac{1}{2}(t_S^a + t_S^d)$$

This provides us with a series of timestamp pairs, t_C^c and t_S^c, for the same event expressed on the client and system clocks. We then use linear regression to estimate a translation between any arbitrary client timestamp t_C^e for event e and the corresponding server timestamp t_S^e:

$$t_S^e = w_1 \cdot t_C^e + w_2 \tag{3.1}$$

Of course, this translation can also be used bidirectionally to translate server timestamps into corresponding client timestamps. To enable approximate synchronization of all the collected data, all events originating on the two remote clients (including user mouse clicks and image selections, button presses, page request times, connection to partner, and audio chunk capture) are logged into the database with the associated client timestamps. Events originating on the server (including image set changes, countdown timer events, and score changes) are logged into the database with the associated server timestamps. All data and events are later approximately synchronized by translating all events onto a common timeline using Equation 3.1. We can reconstruct all the events on the server timeline or user timeline as desired. One limitation of our current approach is that network latency is not completely constant, and thus a dynamic translation might achieve a more accurate synchronization.

3.4 Experiments

In-lab subjects brought to the lab can be tutored and offered guidelines in person by a research staff member which is often important to make the subjects follow the protocols. The background noise can be controlled along with the quality of the equipment involved. However, the main limitation is the bottleneck involved in the number of subjects participating at any given time due to the difficulty involved in parallelizing the task. The difficulty in the task of dialogue data collection in the lab is further exacerbated as the experiments often require multiple participants to be present at the same time and one or more participants do not show up. These limitations can be addressed in the crowd-sourced environments due to a large number of users present in these environments.

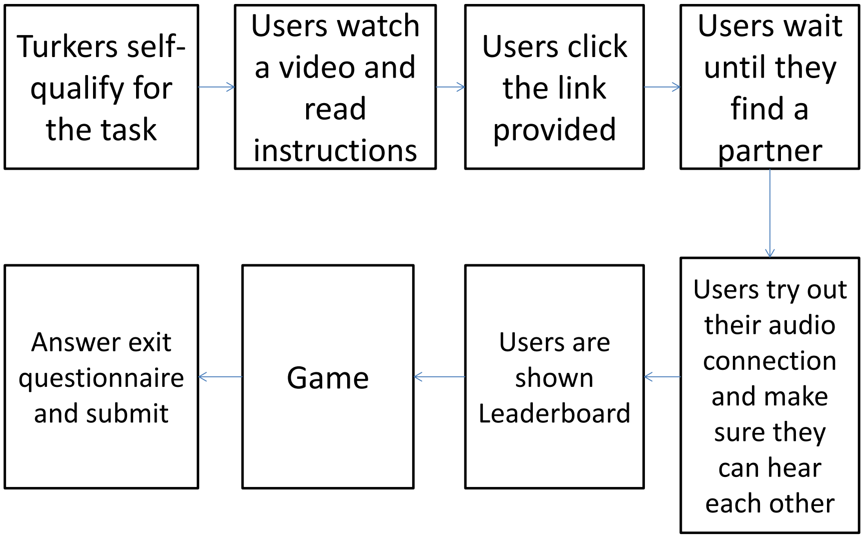

Turkers self-qualify for the task
Users watch a video and read instructions
Users click the link provided
Users wait until they find a partner
Users try out their audio connection and make sure they can hear each other
Users are shown Leaderboard
Game
Answer exit questionnaire and submit

Figure 3.8: The lifecycle of a web-based RDG-Image game HIT.

We recruited 196 individuals from Amazon Mechanical Turk (AMT) to participate in the web-based RDG-Image game. The requirements to participate in the HIT were: (i) a high speed internet connection (5 mbps download, 2 mbps upload); (ii) the latest Google Chrome web browser; (iii) task acceptance of $\geq 92\%$; (iv) previous participation in at least 50 HITs; (v) physical location in the United States; (vi) must be a native English speaker; (vii) must have a microphone; and (viii) must not be on a mobile device.[1]

Figure 3.8 illustrates the lifecycle of the HIT. As part of self-qualifying themselves, Turkers verified their internet connection speed using the speedtest.net web service. Additionally, although this was not a strict requirement, they were strongly encouraged to listen to their computer's sound output using headphones rather than speakers. This instruction was added after pilot testing, to help reduce audio

[1] This requirement is primarily due to the need for simultaneous display of 8 images.

quality issues related to echo.[2] After self-qualifying for the HIT, users proceeded to the instructions, which were provided in both text and video format. The instruction video explained the interface in detail. Users then followed a link and waited until they were paired up with another Turker as a game partner. Access to each Turker's microphone and speakers was then requested from the users. The users then made sure their audio connection worked well. Before playing the game, they were shown a leaderboard where they could see how prior teams performed. After the game, they returned to the AMT site for a post-game questionnaire.

During the data collection, pairing of participants happened on a 'first come, first served' basis. Pair Me Up simply connected each player to the next player who reached the same stage in the HIT, without any scheduling. To attract users, we posted a number of HITs and waited until two consecutive Turkers could be paired up to play the game. Our Pair Me Up server is currently able to support at least 12 simultaneous players (6 simultaneous games). We observed that this approach worked well provided that a sufficient number of HITs was made available on AMT. However, we avoided posting too many HITs at once to prevent exceeding our server's capacity. When too few HITs were available, waiting times for a partner increased. Now we will discuss the results from these experiments.

3.5 Results

3.5.1 Data Collection Throughput

In total our web-based data collection took place over 17 days, and included 177 hours of aggregate HIT time by 196 Turkers. We expected each HIT to take

[2] When the users listen through speakers, it often happens that one of their microphones picks up the speech output of their partner, and echo ensues. We currently do not attempt to cancel this echo.

a minimum of about 15 minutes to complete, including reading instructions, 9 minutes and 20 seconds of actual gameplay, and the post-game questionnaire. The median time was nearly 38 minutes, which is about the same amount of time it took for the participants in the lab to complete the RDG-Image game and fill out all questionnaires. Most of the time spent by our web study participants was spent waiting for a partner. In future work we would like to reduce this wait time by pairing up partners more efficiently. The main bottleneck to parallel game collection on our server is the actual live gameplay, which requires transmission and logging of speech streams. Because our server can support at least 6 simultaneous live games, and the actual dialogue gameplay requires only 9 minutes and 20 seconds per pair, this translates into a potential data collection throughput for the Pair Me Up framework on a single server of hundreds of spoken dialogue games per day. In comparison, our lab-based data collection, which yielded 32 subject pairs, took about a month to orchestrate and complete, due largely to the overhead of individual subject recruitment and scheduling, as well as the impossibility of parallelism given lab resources.

3.5.2 Audio Quality

In our lab-based corpus, audio was captured using high-quality microphones and audio hardware, which were calibrated and adjusted by lab staff for each participant. Additionally, our lab is generally a low noise environment that is free of most distractions. By contrast, in our web audio data, we have very little control over the participants' audio hardware and ambient environments. We observed captured audio to include a wide range of background noises, including televisions, cats meowing, dogs barking, and mobile phones ringing, among other distractions. Our primary use for this audio is through transcription and ASR, in support of dialogue systems research and development. We therefore assess audio quality by way of its suitability for these purposes. We currently have transcribed a subset of the web-based speech amounting to several hundred utterances. For this subset, despite the variable audio conditions, we have encountered no difficulties in transcribing the speech. To assess the audio quality in relation to ASR, we selected a random sample of 105 utterances each from the web corpus and the lab corpus. As part of transcription, these utterances were segmented from surrounding speech (using silence regions exceeding 300ms) and manually transcribed. We then calculated the ASR word error rate for both samples using Google's ASR (https://www.google.com/speech-api/v2/recognize), a broad-coverage cloud-based industry speech recognizer which we have observed to have competitive performance in recent ASR evaluations for dialogue systems at our institute (Morbini et al. (2013)). In our corpora, the observed word error rate (WER) of 24.10 in ASR for web-based audio is significantly higher (W=4647.5, p-value=0.04285, Wilcoxon rank sum test) than the WER of 19.83 for lab-based audio. This increase in WER of 4.27 for web-based audio provides perspective on the trade-offs between controlled lab-based audio capture and crowd-sourced online audio capture for dialogue systems researchers.

3.5.3 Effect of Latency on Game Performance and Synchronization

We summarize the network latency for each user using the round trip time observed in the latency measurement protocol described earlier. Higher values indicate higher network latency that could adversely impact various aspects of gameplay, for example UI responsiveness to user button clicks as well as the speech channel. We observed a mean round-trip latency of 136.9ms (median

108.0ms, standard deviation 84.9ms, min 29.0ms, max 464.0ms). To understand how latency affects overall game performance, we investigated the relationship between round-trip latency and score (number of correct images). We observed a slight weak, but significant, negative correlation between latency and score ($r = -0.16, p < 0.05$). Upon closer examination, the negative effect of latency on score seems to be limited to those players with relatively high latency. We find no significant correlation between score and latency for players whose latency is below 250ms ($r = -0.06, p = 0.44$). Comparing the population of low latency players (latency $<= 250$ms, $N = 177$, mean score 50.7) to high latency players (latency > 250ms, $N = 19$, mean score 40.5), we observe a significant difference in scores ($p < 0.05$, Wilcoxon rank-sum test). We interpret this data as suggesting that if latency is low enough, its effect on game score is negligible. Additionally, we used our latency measurement and synchronization protocol to construct more than 20 synchronized videos that combine the two users' speech streams with a recreation of each user's UI state at each moment (including images observed, button clicks, etc.). If timeline correction using Equation 3.1 is not performed, such videos exhibit numerous clear synchronization problems. After timeline correction, we have found that the combined videos appear remarkably well synchronized. Upon observation of these videos, we are unable to detect any remaining latency or synchronization issues, and we view the data as sufficiently synchronized for our research purposes. In future work we would like to further investigate the exact accuracy of data synchronization achieved.

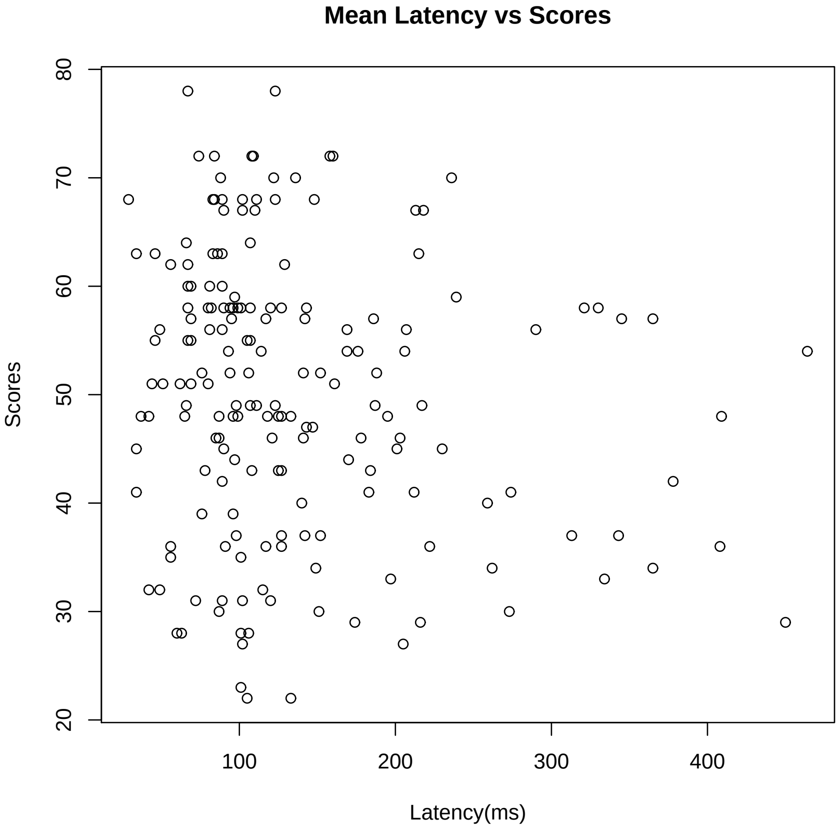


Figure 3.9: Scatter plot of game scores vs. round-trip latency of all the users. The points scored by the users with high speed internet connection (low mean latency) is higher than the users with lower speed of internet connection.

3.5.4 Cost, Gameplay Quality, and Naturalness of Interaction

We summarize the study cost, scores attained, and basic demographic data for our two corpora in Table 3.1. From Table 3.1 we can see that the web-study data is 7.8x less expensive per participant to collect (once the Pair Me Up infrastructure is in place). In terms of acquiring examples of excellent gameplay, which is one of our research requirements, we found that our web-study players scored significantly higher than the players in lab ($W = 5389$, $p = 0.01875$, Wilcoxon rank sum test).

	Web	Lab
N	196	64
Average Pay per player	\$1.915	\$15
Scores(%) [Mean, SD, Min, Max, Median]	49.8, 13.1, 22, 78, 51	45, 13.0, 20, 68, 44
Age(%) [Mean, SD, Min, Max, Median]	31.3, 8.2, 19, 68, 29	36.6, 12.7, 18, 60, 34.5
Gender(%) [Female, Male] (%)	53.3, 46.7	55, 45

Table 3.1: Cost, scores attained, and demographic data for our web and lab studies.

The full explanation for this difference is unclear as there were several differences between the web study and the lab study. One difference is that web-study participants were incentivized with a bonus payment per correct image, while lab study participants were paid a flat rate of \$15 for participation. Demographic differences between Turkers and Los Angeles area Craigslist users may also have played a role; for example, our web-study participants were younger on average. In any case, we conclude that it is possible to collect examples of excellent gameplay for RDG-Image with a crowd-sourced data collection. All participants filled out post-game subjective questionnaires, providing answers on a 5-point Likert scale. We were especially interested in the perceived naturalness of the interaction and the usability of the interface, and we present several observations in Figure 3.10. All significance tests are Wilcoxon rank sum tests.

Web-study participants gave significantly higher ratings of the user interface being intuitive and easy to use (Q1). They also gave higher ratings to the ease of understanding the game rules (Q2) and it being easy to play the game with their partner (Q5). These findings may be partially explained by the more detailed instructions we provided for web users about the browser interface, including the addition of video-based instructions. Demographic differences and possible comfort in using a browser-based interface could potentially play a role as well. In terms of naturalness of the interaction, the results were also favorable for the web-based study. Despite our concern about network latency affecting interaction

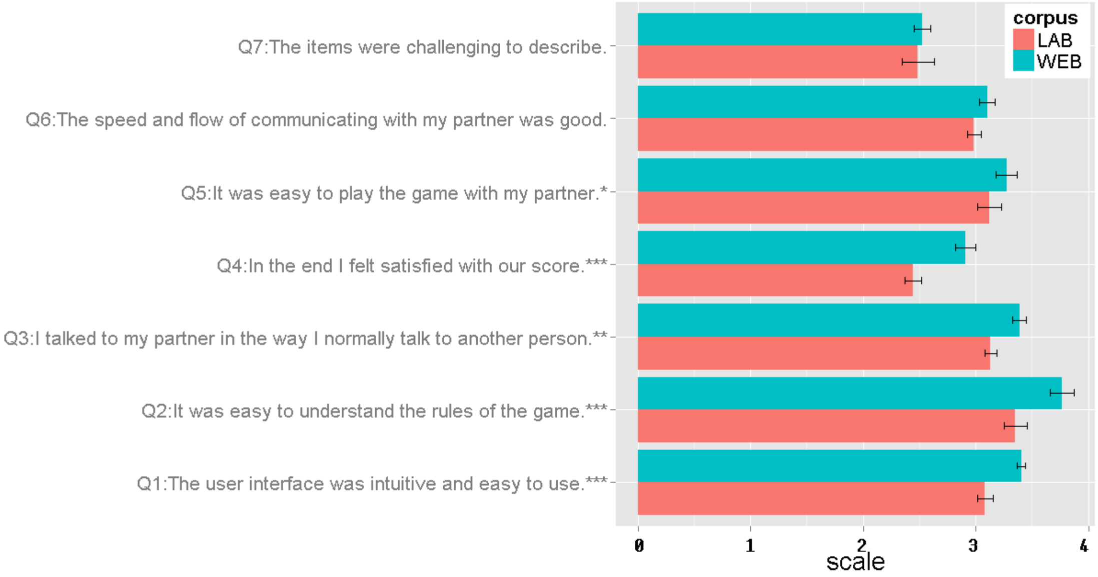


Figure 3.10: Subjective questionnaire results for questions related to interaction naturalness and usability of user interface. Means and standard errors are shown for all questions. (* indicates p <0.05, ** indicates p <0.01, *** indicates p <0.001).

naturalness, we observed no significant difference in ratings of the speed and flow of communication between the web study and the lab study (Q6). In fact, web-study participants gave significantly higher ratings to it being easy to play the game with their partner (Q5), satisfaction with their score (Q4), and a rating of whether they spoke the way they normally do with the partner they were paired with (Q3). The fact that web-study participants scored higher than lab-study participants may play a role in the perceived ease of playing with their partner and score satisfaction.

3.5.5 Discussion

We have presented a web framework called Pair Me Up that enables spoken dialogue interactions between remote users to be collected through crowd-sourcing. We have confirmed, for spoken dialogue interactions in the RDG-Image game, the commonly observed pattern that crowd-sourced data collection over the web can be faster and much less expensive than data collection in the lab. At the same time, we have explored several trade-offs in web-based vs. lab-based data collection for dialogue systems research. In terms of audio quality, we have found an increase of about 4% in ASR word error rate for web-based audio data. Such an increase may be acceptable by many researchers in exchange for easier data collection. In terms of network latency, we have found that while it is an important consideration, it does not rule out natural real-time dialogue between the remote participants, and that data can still be synchronized sufficiently for our purposes using a straightforward latency measurement protocol. We have observed that the quality of gameplay, as determined by scores achieved and several subjective assessments by Turkers, was higher for our crowd-sourced study than in the lab.

In this chapter we have reported on a web-based framework that helps address a critical data-collection bottleneck in the design and evaluation of SDS. We demonstrated the viability of our framework through a data collection study in which 196 remote participants engaged in human-human dialogue interactions in an image matching game. We discussed several of the technical challenges we encountered and some of the limitations in our current process for collecting dialogue data over the web. In future work, it would be fruitful to address the challenge of managing available computing resources better in order to further reduce costs and accelerate data collection.

We are also interested in SDS that operate over the web. The incremental architecture and the design choices involved in the development of such an agent are not trivial. We extended the PMU framework to support the deployment of an agent to collect data in a human-agent configuration. Now we will discuss the modifications to the PMU framework to accommodate deploying an automated agent over the web.

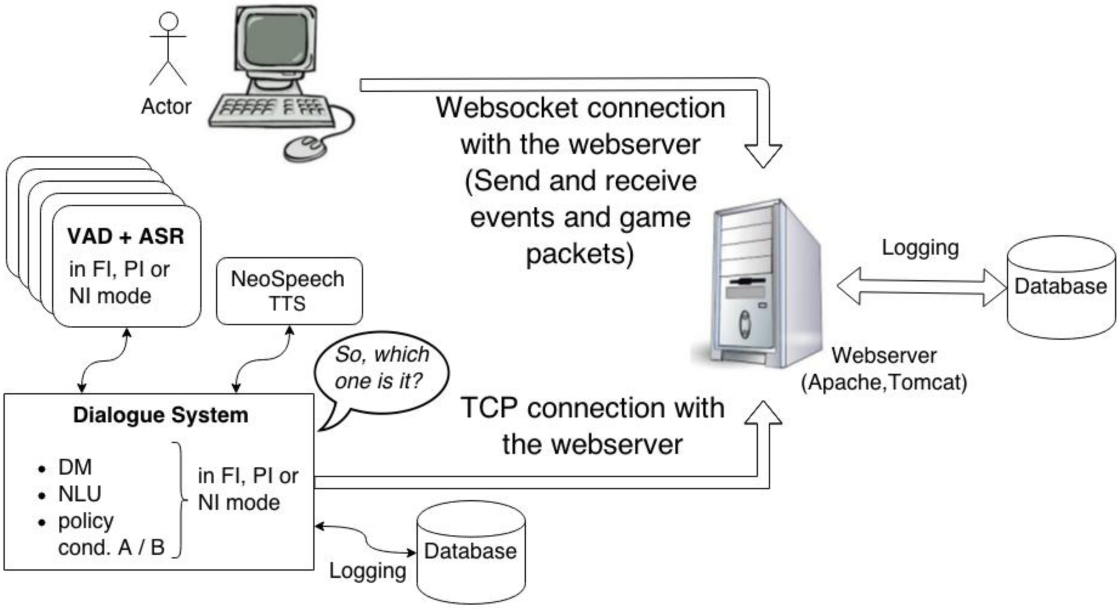


Figure 3.11: Pair Me Up architecture in human-agent mode.

3.6 Extending to Human-Agent Data Collection

The agent mode for PMU is configured in a similar way to the human-human mode, as shown in Figure 3.11. The user connects to the PMU server by following a URL in their browser. A WebSocket connection is used to transmit user audio, system audio, and game events between the remote user and the PMU server. The PMU server runs both a web server process and the automated agent, and these two communicate with each other through TCP sockets. The agent includes internal modules for NLU and DM/Policy. The agent also communicates using TCP socket connections to external processes for Voice Activity Detection (VAD), ASR, TTS, and a database for logging.

Now we will discuss the design of the agents that are implemented to operate on the web.

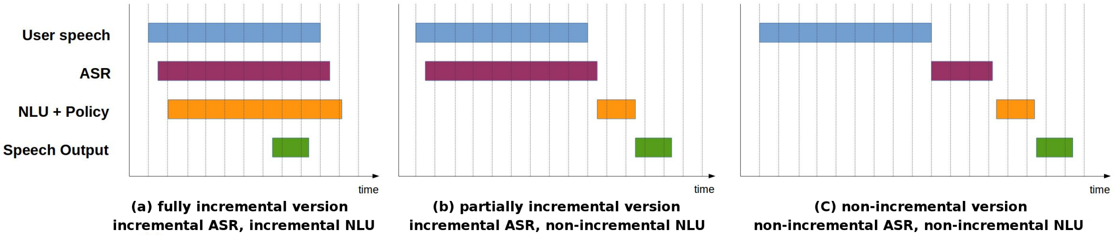

User speech
ASR
NLU + Policy
Speech Output
time
time
time
(a) fully incremental version
incremental ASR, incremental NLU
(b) partially incremental version
incremental ASR, non-incremental NLU
(C) non-incremental version
non-incremental ASR, non-incremental NLU

Figure 3.12: Different versions of the agent.

3.6.1 Agent Design

Six agent versions

To support our research on incremental processing techniques, we ran a data collection and evaluation involving six different versions of the agent. While other researchers might not share our specific interest in these six versions, the desire to compare several alternative system designs in an empirical way, ideally using interactive human-agent data, is common to many research efforts.

In our case, our study was designed to evaluate three versions of incrementality and two different policy optimization metrics against each other. The three incremental versions consist of the fully incremental (FI), partially incremental (PI) and non-incremental (NI) versions. Figure 3.12 illustrates the different versions and their modes of operation. In the FI architecture, the ASR, NLU, and DM/Policy are all operating incrementally after every additional 100 ms of user speech. This setup enables the agent to give fast-paced feedback while the dialogue partner is still talking. For the PI version, only the ASR is operating incrementally; the NLU and DM/Policy wait for a VAD segment (inter-pausal unit) to finish before they start processing. Here, the agent cannot interrupt the user, but is still able to give a quick response once a pause is detected. In the NI architecture, the ASR, NLU, and DM/Policy are all operating on complete VAD segments as input, which increases the delay between the end of the user's speech and the beginning of the agent's response.

Additionally, we optimized policies using two different optimization metrics, which we denote simply A and B in Figure 3.11. The details of the two optimization metrics are omitted; their technical rationale and motivation is beyond the scope of this chapter. Together, the incrementality type and policy type variations creates a 3x2 study design, for a total of six agent versions to evaluate.

An ability to evaluate so many different agent prototypes empirically is valuable for many research questions, but it also confronts researchers with the difficulty of evaluating them with a significant number of participants in a tight timeframe and with limited financial resources.

The agent's internal modules are designed in a way that the agent can easily switch between different policies and incrementality types. It only requires the PMU server to send information about the policy version to use in the beginning of a game round. The agent can even handle interactions in different versions of incrementality simultaneously. The ASR cannot however switch between incremental and non-incremental processing at run-time, which means one instance of the ASR can only serve one version of incrementality.

3.6.2 Crowd-Sourced Data Collection Using Agent

200 native English speakers aged over 18 were recruited on Amazon Mechanical Turk (AMT) to participate in the study. 25 of them were paired with another human (25×2), and 25 played with each of the six versions of the agent (25×6). The study was conducted over a period of 10 days. Table 3.2 summarizes the participant demographics in the study.

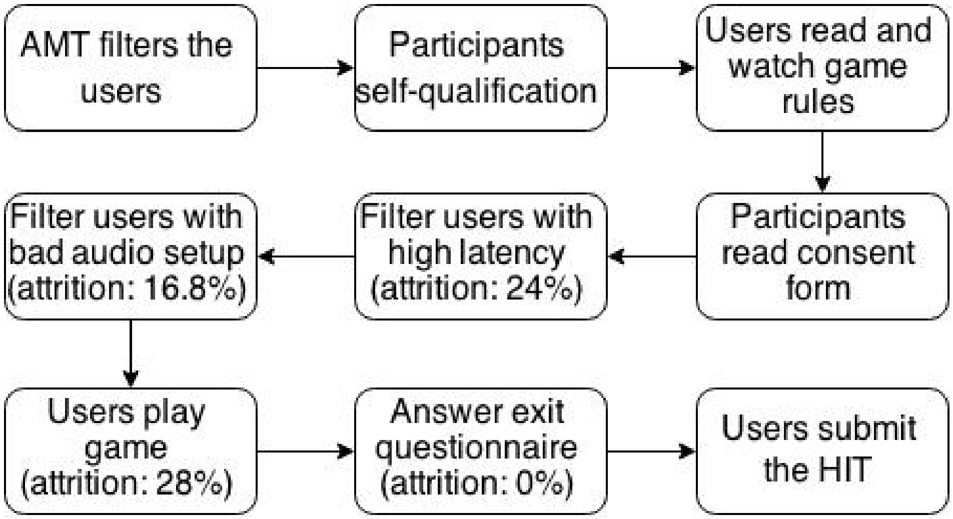

AMT filters the users
Participants self-qualification
Users read and watch game rules
Participants read consent form
Filter users with high latency (attrition: 24%)
Filter users with bad audio setup (attrition: 16.8%)
Users play game (attrition: 28%)
Answer exit questionnaire (attrition: 0%)
Users submit the HIT

Figure 3.13: The steps undergone by the users during the study.

The study was conducted entirely over the internet. The protocol involved in recruiting and filtering the participants to guarantee congenial data for the human-agent condition is shown in Figure 3.13 and discussed in the rest of this section.

AMT filters users. AMT is able to apply certain filtering criteria for the participants. We had AMT apply the following criteria: (i) Participants have an acceptance rate equal to or greater than 92% in their previous Human Intelligence Task (HIT) participations; (ii) previous participation in at least 50 HITs; (iii) physical location in the United States or Canada.

Participant's self qualification. The users who AMT qualified for the HIT were provided instructions to participate only if they met the following criteria: (i) must have the latest Google Chrome web browser; (ii) must be a native English speaker; (iii) must have a microphone; (iv) must not be on a mobile device; (v) must have a high speed Internet connection (5 mbps download, 2 mbps upload). Additionally, users were asked to use earbuds or headphones rather than external speakers, which helps prevent their microphone picking up sounds from their own speakers.

Users read and watch game rules. The rules of the game are explained in text and video format. The users are then led to a consent form web page where **participants read the consent form** and decide if they want to participate or not. The users enter their ID and submit their consent.

Filter users with high latency. To prevent certain users with problematic network latency from participating, we measure the network latency between the user and our server. 24% of the users who consented to the experiment did not qualify at this stage due to high latency or highly variable latency.

Filter users with bad audio setup. The users in the next step were made to listen to an audio file and transcribe it. If the transcriptions were wrong, the users were disqualified. This is to make sure that the users had a functioning speaker/headphone set up. The users then had to speak three pre-selected sentences in their microphone. The ASR transcribes the spoken audio and if the user had at least one word right from the sentences, the users were qualified, else disqualified. 16.8% of the users got disqualified at this step due to a 'bad audio set up'.

The qualified **users play the game** with the agent. 28% of the users who qualified from the previous stage did not finish playing the game with the agent. It happened that sometimes Turkers closed the browser or otherwise stopped participating for reasons we could not discern. After the game, the users were made to **answer an exit questionnaire**. After answering the questionnaire the users were instructed to return to AMT and asked to **submit the HIT**.

Posting the HITs for users to complete was managed manually for this study. The users were given 40 minutes to complete the task.

	Agent	Human
N	150	25
Female (%)	54.7	44
Age (yrs)		
Mean	31.12	31.12
Median	28	28
SD	10.2	10.4

Table 3.2: Demographic data for the 175 human directors, based on whether the matcher was an agent or another human.

Technical challenges encountered

We faced several technical challenges in achieving this data collection. The challenges can be categorized into three main categories.

Filtering out users based on latency. Latency can potentially affect the collected data. We filter out the users with high latency using the protocol discussed in Section 3.3.

Dealing with effects of variable latency. Even with the thresholds mentioned in the previous section, transient fluctuations in network latency can sometimes occur, and we found we needed a special mechanism to ensure the integrity of the audio channel. Audio packets are recorded and sent to the PMU server from the client's browser in chunks of approximately 100 ms. Each chunk is sent separately, and is subject to variable transit time due to varying network latency from moment to moment. The order of these packets is thus not guaranteed and they can arrive out of order. For instance, if the audio packets A, B, C are recorded at times t, $t+100$ms, $t+200$ms respectively, it is possible for the server to receive them in order A, C, B. If not corrected, this order violation would corrupt the captured audio waveform and potentially degrade ASR and system performance. To overcome this issue, we used an auto-incrementing sequence ID that was appended to each audio packet before it left the user's browser. On the server, we monitor these sequence IDs to make sure that the audio packets either arrive in order or are reordered appropriately by the server.

Managing server load. Even though the agent was designed to handle multiple users at a time, we found in pilot testing that processor and memory usage by the system (agent, web server, database, ASR) was sometimes too high to support low-latency gameplay by multiple simultaneous users on the available hardware. We therefore decided to limit the agent to one user per server to avoid this issue affecting gameplay, and deployed the system on a commercial cloud-hosting provider using six different servers. Our study could thus support up to 6 simultaneous users. HITs were kept active for all 6 servers throughout the study. A feature of AMT HITs is that the they are no longer available to others if the HIT is being worked on by a participant. Due to the high attrition rates of participants at various steps in the HIT (Figure 3.13), sometimes a server was left idle for the maximum HIT completion time of 40 minutes. We did not attempt to build a resource management system to enable more efficient use of our computing resources.

3.6.3 Analysis of Crowd-Sourced Study Cost

Table 3.3 shows several types of measured costs that were incurred in this web-based study (Web column). It also includes, for comparison, an estimate of what the corresponding costs would be for a lab-based human-agent study (Lab column). The costs in Table 3.3 for running the study in the lab environment are estimated based on the human-human lab study.

	Web	Lab
Participant Fees	\$1.24	\$15
Staff Time per Participant	2.5 min	∼35 min
Cost of Server Time	\$0.72/hour/machine	–
Participant Time	1193.1 sec	∼1800 sec

Table 3.3: Comparison between studies in the lab and web. Estimated numbers are indicated by ∼.

Participant fees. The web users were compensated an average of \$1.24 (Max=0.56, Min=0.04, SD=0.12) (N=150) per player when interacting with the agent. In the lab, a payment of \$15 was granted for 30 minutes of participation in the human-human study. An incentivized payment based on score, as in the web study, might further increase the cost.

Staff time per participant. To manage the HITs on the web required about 2.5 minutes of staff time per participant. In the lab, a staff member needs 30 minutes plus about 5 more minutes per participant for preparing the lab and the recording equipment.

Cost of server time. For the 150 successful human-agent participants, the servers in this study were actually used for a total of 49.71 hours. The 50 human-human participants required approximately an additional 20 hours of server time. However, due to inefficiencies in our process, during the study, the six servers were kept active for 10 days (1440 server hours). Each server hour costs \$0.72. In the lab, the hardware expenditures for a similar study would be highly dependent on the researcher's environment, but they include the cost of a computer and high-quality audio equipment (about \$800).

Participant time. The mean total gametime on the web was 275 seconds, but mean participant time was 1193.1 seconds. The additional time was spent by the users on validation steps and answering the questionnaire. In the lab, we estimate that participants would need about 30 minutes for completing the study, including reading and signing the consent form, reading the game rules, playing the game, and answering the questionnaire in the end. In practice, the process takes a little more time in the lab as there is additional time needed for the staff member to greet the participant, manually start the software, adjust the microphone placement, answer any questions, etc.

Overall, it can be seen that this crowd-sourced, web-based approach to human-agent dialogue collection offers potential reductions in several types of costs, including substantial reductions in participant fees and staff time per participant.

3.6.4 Limitations

There are several limitations in the way this study was conducted. In the human-human condition, one of the major hurdles is the waiting times involved in creating pairs, which can sometimes be measured in hours. To try to streamline the pairing process, in pilot testing we attempted several methods. We put up a calendar scheduling system where the users could mark their availability, with time slots provided every 30 minutes. Users could avoid waiting to make a pair by selecting a time when another user had stated they were available. However, we found many Turkers would select a time slot but then not show up at the specified time. Another technique we tried was a variant of a calendar where the users were paid $0.05 to mark their availability and to then show up at that time. However, again many Turkers would not show up at the appointed time. We finally adopted a first-come, first-served method that paired consecutive participants. Although this method was relatively slow, as individuals had to wait until a pair could be formed, and had high attrition rates, it was found to work sufficiently well to obtain 25 human-human pairs.

In the human-agent condition, the primary limitation was that there was a large amount of idle system time across our six servers (totaling to about 1370 server hours). This suggests that we had unmet capacity which could have been used to support additional dialogues, or alternatively, we could have used fewer servers to support the same number of users (thus reducing hosting costs). This idle time is related to the high attrition rates (Figure 3.13) and non-uniform participant presence on AMT during the times when our HITs were active. It would be fruitful to tackle these issues by optimizing our HIT and qualification processes in future work.

3.7 Other Pre-processing Steps

Once the audio data is collected, it is necessary to transcribe and annotate this data to aid the process of model building. The audio data is either transcribed automatically or using crowd services such as AMT. In either of the cases, it is important to correct the transcriptions for errors. Figure 3.14 shows the interface used by an expert to correct the transcriptions. The data needs to be annotated with labels (such as dialogue acts) when these labels can not be automatically extracted. In Chapter 6, we will discuss the annotation scheme and the interface used. Once the data is transcribed and annotated it is ready for building models. In the next chapter, I will discuss the process of policy building in incremental spoken dialogue systems.

3.8 Contributions

In this chapter, I discussed the contributions made towards building an incremental SDS.

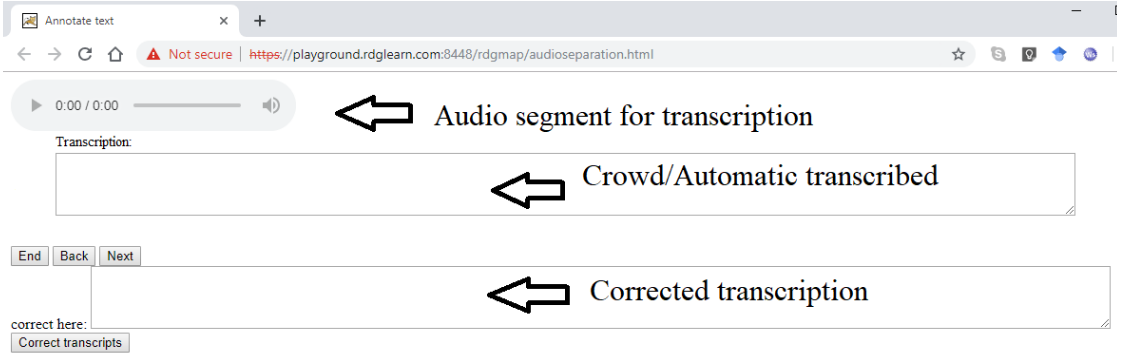

Annotate text
https://playground.rdglearn.com:8448/rdgmap/audioseparation.html
Not secure
0:00 / 0:00
Audio segment for transcription
Transcription:
Crowd/Automatic transcribed
End
Back
Next
Corrected transcription
correct here:
Correct transcripts

Figure 3.14: The transcription interface used to transcribe data using crowd-sourcing.

The PMU framework is one of the first in a limited number of data collection frameworks that utilize the HTML5 paradigm towards building a crowd-sourced spoken dialogue corpus. In fact, to our knowledge, the PMU framework is the only framework in the literature that can support collecting spoken dialogue data on the web for building incremental SDS. The PMU framework can be used for collecting both human-human and human-agent data. HALEF (Suendermann-Oeft et al. (2015)) and ParlAI (Miller et al. (2017)) are two recently developed frameworks for collecting conversational data on the web. However, ParlAI only supports text-based dialogue (as of June 2019), and HALEF is not designed to capture timing information which is essential for building incremental SDS.

We showed that by utilizing the crowd-sourcing paradigm, we can not only collect spoken dialogue data but also save a significant amount of time and money in the process. This is the first work in the literature showing such comparison between data collected in the lab and via crowd-sourcing environments. The users on crowd-platforms were found to be more competitive in the task and scored more points compared to the in-lab participants. The users could also complete the task with minimal supervision. These results could help dialogue researchers leverage the crowd-based data collection paradigm and save time and money in the process with a guarantee of collecting a quality spoken dialogue corpus. The work presented in this chapter has been published in Manuvinakurike et al. (2015), Paetzel et al. (2015), and Manuvinakurike & DeVault (2015).

Chapter 4

Eve: An Incremental Spoken Dialogue System

"Truth is much too complicated to allow anything but approximations."
– John von Neumann, *Mathematician, Physicist, Computer Scientist*

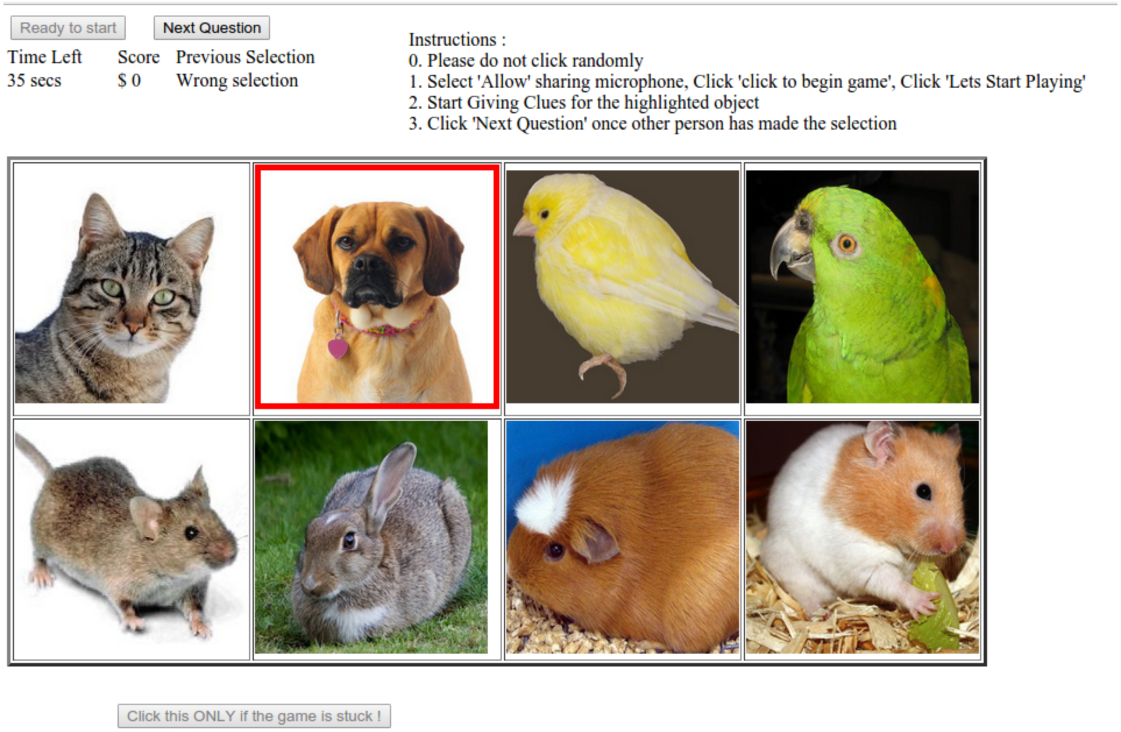
4.1 Introduction

In this chapter, I discuss the development of the RDG-Image agent called Eve and present a study showing the importance of incrementality.

4.1.1 The RDG-Image (Rapid Dialogue Game - Image)

In the RDG-Image, one person acts as the Director and the other as the Matcher. Players see a set of eight images on separate screens. The set of images is exactly the same for both players, but they are arranged in a different order on the screen. Image sets include pets (Figure 4.1), fruits, bicycles, road signs, and robots, among others. One of the eight images is randomly selected as a target image (TI) and it is highlighted on the Director's screen with a thick red border as shown in Figure 4.1.

The goal of the Director is to describe the TI so that the Matcher is able to uniquely identify it from the distractors. For example, the Director might say simply *the dog* for the TI in Figure 4.1. The Director and Matcher are able to talk back-and-forth freely in order to identify the TI. When the Matcher believes he has



Time Left 35 secs
Score $ 0
Previous Selection Wrong selection

Instructions :
0. Please do not click randomly
1. Select 'Allow' sharing microphone, Click 'click to begin game', Click 'Lets Start Playing'
2. Start Giving Clues for the highlighted object
3. Click 'Next Question' once other person has made the selection

Figure 4.1: Browser interface for the Director. The target image is highlighted by a red border. The *Next Question* button moves on to the next target.

correctly identified the TI, he clicks on the image and communicates this to the Director who has to press a button to continue with the next TI. The players also have the option to skip a TI if desired. The team scores a point for each correct guess, with a goal to complete as many images as possible.

Each team participates in 4 main game rounds. In this study, the roles remain the same for the players across all four rounds. Our agent is always in the Matcher role in this study. The maximum number of TIs within each round is 12, and the rounds have a variable duration ranging from 45 to 60 seconds. The time limit for each round was chosen based on analysis of the sub-dialogues for that round's image sets in our earlier game corpora (Paetzel et al. (2014); Manuvinakurike & DeVault (2015)). The time limit was explicitly set to prevent participants in this study from exhausting the 12 images in a round before they run out of time. In this way, the speed and accuracy of communication are always limiting factors to higher scores[1]. The participants never run out of available target images within a round.

One game in this study consists of one training round, during which participants get comfortable with the interface and (if playing with one of our agents) the agent's interaction style, their partner, plus four main game rounds which are scored. The maximum game score is therefore 48 points (4*12). Following our approach in Manuvinakurike & DeVault (2015), participants are incentivized to score quickly with a bonus of $0.02 per point scored. To keep the game fun and challenging, image sets become more difficult over time.

4.2 Observations of Human Matchers

Two corpora of human-human gameplay have previously been collected for the RDG-Image game, including the RDG-Image lab corpus (collected in our lab) (Paetzel et al. (2014)) and the RDG-Image web corpus (collected on the web) (Manuvinakurike & DeVault (2015)). These corpora were used to design our automated agent.

A first step was to identify the most common types of Matcher utterances and behavior in our lab corpus. To support this analysis, 21 dialogue acts (DAs) were defined. The most important DAs for our automated Matcher agents are *Assert-Identified*, used for utterances such as *Got it!* that assert the TI has been identified, and *Request-Skip*, used for utterances such as *Let's move on* that ask the Director to advance to the next TI.

[1] In earlier studies (Paetzel et al. (2014); Manuvinakurike & DeVault (2015)) we used longer time limits, and players could sometimes run out of available images.

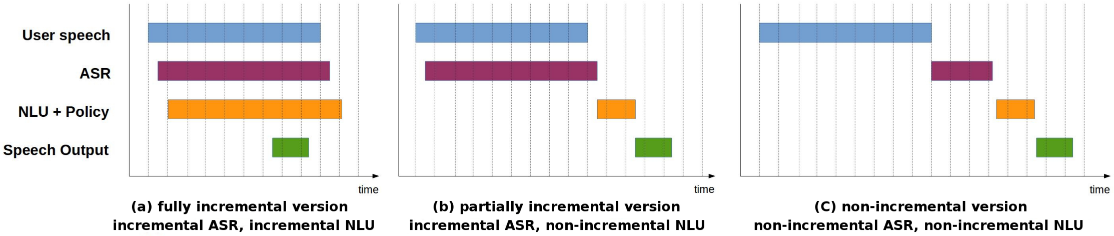


Figure 4.2: Timeline of the processing order of the modules in the three different versions of incrementality.

34 human-human games were manually transcribed and annotated for dialogue acts (DAs) by a human annotator, resulting in 5415 annotated DAs. The inter-annotator agreement, measured by Krippendorf's alpha, is 0.83. 40.70% of all Matcher DAs were *Assert-Identified*, and this is by far the most common DA by the Matcher. For the Matcher, this is followed by 15.83% of DAs which are annotated as *Out-of-domain* DAs such as laughter or meta-level discussion of the game. All other Matcher DAs occur in less than 6.5% of DAs each.

Our analysis of these annotations revealed that, typically, the Matcher simply listens to the Director's continuing descriptions until they can perform an *Assert-Identified*, rather than taking the initiative to ask questions, for example. The top of Figure 4.3 shows a typical image subdialogue.

4.3 Design of the Agent Matcher

Based on our observations of human Matchers, we focused our design of Eve on the *Assert-Identified* and *Request-Skip* acts. *Request-Skip* is a move not often used by Matchers in human-human gameplay, where teams tend to take additional time as needed to agree on each image, and where teams eventually, score a point for 92-98% of the TIs they encounter (depending on the image set). We anticipated that Eve might struggle with certain images or image sets, because its NLU would be data-driven and its understanding limited to previously seen description types. Eve is therefore designed to use *Request-Skip* strategically if trying to score on the current TI appears not to be a good use of time.

4.3.1 Training Data

To train our agent, the 16 image sets containing the most training examples per set were chosen from the RDG-Image lab and web corpora. Additionally, two sets of simple geometric shapes from the lab corpus were selected to serve as a training round in this study. The lab corpus includes 34 games with 68 unique participants and 1877 image subdialogues. The web corpus includes 179 participants (some of them in multiple games) and 2788 image subdialogues. In our total training data, on average, there are 259.17 image subdialogues per image set and 32.40 per image.

4.3.2 Voice Activity Detection (VAD), Automatic Speech Recognition (ASR)

Audio is streamed from the user's browser to our voice activity detector, which uses the Adaptive Multi-Rate (AMR) codec 3rd Generation Partnership Project (2008) to classify each incoming 20ms audio frame as containing voice activity or not. The VAD works incrementally in all versions of our agent. It emits voice activity events and delivers segments of detected speech (in units of 100ms) to the ASR.

Our ASR is based on Kaldi (Povey et al. (2011)) and is specifically adapted from the work of Plátek & Jurčíček (2014), which provides support for online incremental recognition. Discriminative acoustic models are trained using a combination of our in-domain audio data and out-of-domain audio using Boosted Maximum Mutual

Information (BMMI) with LDA and MLLT feature transformations (Plátek & Jurčíček (2014)). Statistical language models are created using our transcribed data.

Incremental ASR. In versions of our agent with incremental ASR, detected user speech is streamed into the ASR every 100ms for online decoding, and incremental (partial) ASR results are immediately computed and sent to the NLU and policy modules. Incremental ASR is illustrated at the left of Figure 4.3. It is used in the fully incremental and partially incremental versions of our agent, which are illustrated in Figure 4.2(a) and (b).

Non-incremental ASR. In the non-incremental version of our agent (see Figure 4.2(c)), detected user speech is buffered until the VAD segment is concluded by the VAD. At that point, all speech is provided to the ASR and the final ASR result is computed and provided to the NLU and policy.

4.3.3 Natural Language Understanding (NLU)

Our NLU operates on 1-best text outputs from the ASR. At each time t, all the 1-best texts for the current TI (i.e., spanning multiple VAD segments) are concatenated to form a combined text d_t which we call the image subdialogue text. For example, at time $t = 2.72$ in Figure 4.3, the NLU input is $d_t =$ *uh okay a rock*.

Prior to classification, stop-words are filtered out.[2] This process yields for example the filtered text filtered(*uh okay a rock*) = *rock*. From the filtered text, unigrams and bigrams are calculated. To reduce overfitting, only those unigrams and bigrams which occur more than three times in our training corpus are kept. The remaining unigrams and bigrams are used as input for the classifiers. We also

[2] The stop-word list is based on http://jmlr.org/papers/volume5/lewis04a/a11-smart-stop-list/english.stop and extended by domain-specific stop words.

considered other features like text length and number of pauses, but they did not significantly increase the classification accuracy.

A separate classifier is trained for each image set. The approach is broadly similar to DeVault et al. (2011b), and each partial ASR result is probabilistically classified as one of the eight TIs. The training data maps all the image subdialogue texts in our corpora for that image set to the correct TI. To select the classifier type, Weka (Hall et al. (2009)) was used on manually transcribed data from the RDG-Image lab corpus. Multiple classifiers were tested with 10-fold cross validation. The best performance was achieved using a Naive Bayes classifier, which classified 69.15% of test instances correctly. Maximum Entropy classification performed second best with 61.37% accuracy.

4.3.4 General Form of Eve's Dialogue Policies

Eve's dialogue policies take the following form. Let the image set at time t be $\mathcal{I}_t = \{i_1, ..., i_8\}$, with the correct target image $T \in \mathcal{I}_t$ unknown to the agent. The maximum probability assigned to any image at time t is $P_t^* = \max_j P(T = i_j | d_t)$. Let elapsed($t$) be the elapsed time spent on the current TI up to time t.

Eve's parameterized policy is to continue waiting for additional user speech until either her confidence P_t^* exceeds a threshold IT, or else the elapsed time on this TI exceeds a threshold GT. The *identification threshold (IT)* represents the minimal classifier confidence at which Eve performs an *Assert-Identified* (by saying *Got it!*). The *give-up threshold (GT)* is the time in seconds after which Eve performs a *Request-Skip* (by saying *I don't think I can get this one. Let's move on, I just clicked randomly.*).

Eve's policy is invoked by different trigger events depending on the incremental architecture. In the fully incremental (FI) version (Figure 4.2(a)), the policy is

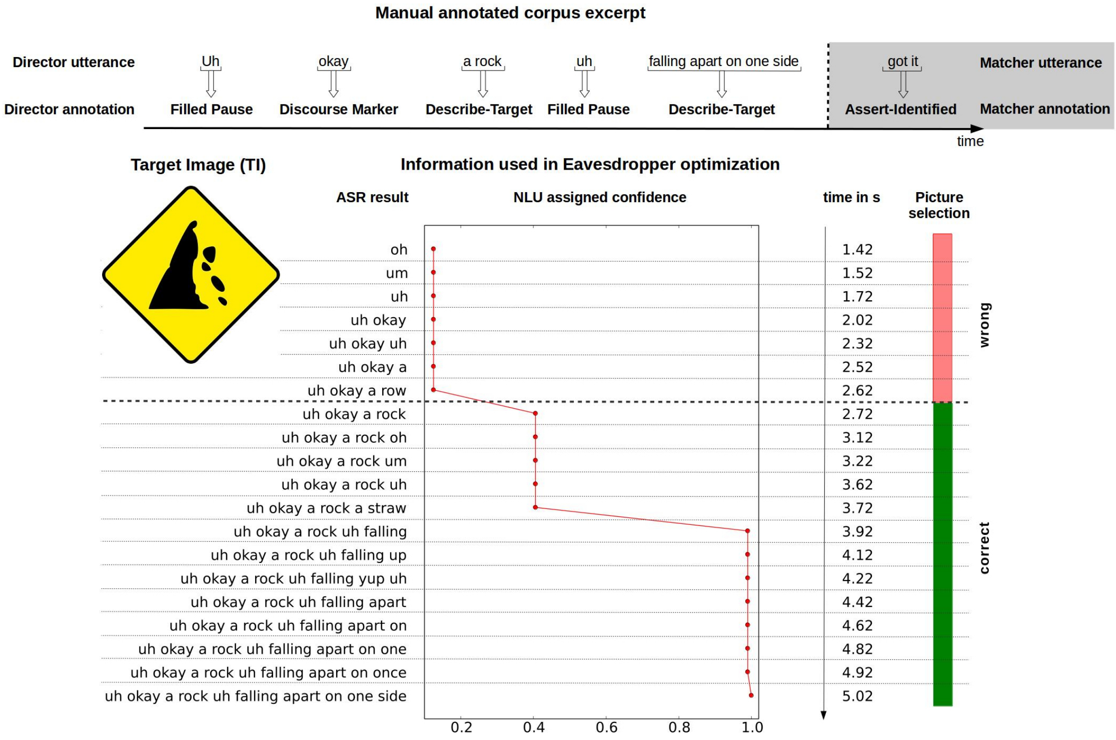


Figure 4.3: An image subdialogue from the RDG-Image lab corpus. The upper part shows the manual DA annotation. The lower part shows information used in the eavesdropper policy optimization. For brevity, we include only partial ASR results that differ from the previous one. In the middle and on the right are the NLU's evolving classification confidence, elapsed time, and correctness of the NLU's best guess image.

invoked with each new partial and final ASR result (i.e., every 100ms during user speech). In the partially incremental (PI) and non-incremental (NI) versions (Figure 4.2(b) and (c)), the policy is invoked only after a new final ASR result becomes available. Each time Eve's policy is invoked, Eve selects an action using Algorithm 1.[3] Eve's policy allows the agent to make trade-offs that incorporate both its confidence in its best guess and the opportunity cost of spending too much time on an image. In Section 4.4, we describe how we optimize the numeric parameters IT and GT in these policies.

[3] Requiring $|\text{filtered}(d_t)| \geq 1$ prevents Eve from ever saying *Got it!* before any content words (non-stop words) have been received from the ASR.

Algorithm 1 Eve's dialogue policy

```
if $P_t^* >$ IT & $|\text{filtered}(d_t)| \geq 1$ then
    Assert-Identified
else if elapsed($t$) $<$ GT then
    continue listening
else
    Request-Skip
end if
```

4.3.5 Text-to-Speech Synthesis (TTS)

Eve uses NeoSpeech[4] TTS. All Eve utterances are pre-synthesized to minimize output latency.

4.3.6 Discussion of Incremental Architectures

The non-incremental (NI) version serves as a baseline for the performance if none of ASR, NLU, or policy are carried out incrementally. The partially incremental (PI) version helps quantify the benefits that come from reducing system latency through online decoding in the ASR. The fully incremental (FI) version explores the benefits of reacting more continuously during user speech.

4.4 Policy Optimization

Optimization of the parameters IT and GT in Algorithm 1 is done using a metaphor of the agent as an *eavesdropper* on human-human gameplay. To train our agent, we start by imagining the agent as listening to the speech in human-human image subdialogues from our corpora. We imagine that as the human Director describes an image to his partner, our eavesdropping agent simulates making its

[4] http://www.neospeech.com/

own independent decisions about when, if it were the Matcher, it would commit to a specific TI (by saying “Got it!”) or request an image skip (by saying “Let’s move on..”).

For example, in Figure 4.3, we visualize the ASR results that would be arriving in the FI architecture, and the time at which they would be arriving, as this human Director describes the TI as *uh okay a rock uh falling apart on one side*. In the middle and right, we visualize what the agent’s NLU confidence would be in its best guess (P_t^*) as these ASR results arrive. At the right, we show that this best guess is incorrect until time 2.72.

In our optimizations in this study, we assume that the objective metric to be maximized is *points per second* (points/s). The key idea in this optimization is that each value of parameters IT and GT in Algorithm 1 translates into a specific simulatable agent response and outcome for each Director description of a TI in our corpus. For example, if IT=0.3 and GT=5, then in the figure’s example the agent would commit to its best interpretation at time 2.72 by performing Assert-Identified (“Got it!”). The agent would turn out to be correct and score a point. The time taken to score this point would be 2.72 seconds, plus some additional time for the Matcher to say “Got it!” and for the Director to click the Next Question button in the UI (see Figure 4.1). Our agent needs 0.5 seconds to say “Got it!”, and we add an additional 0.25 seconds equal to the mean additional Director click latency in our corpora. The total simulated time for this image is therefore 2.72+0.5+0.25 = 3.47 seconds.[5]

If one simulates this decision-making across an entire corpus, then for each value of IT and GT, one can calculate the total number of points hypothetically

[5] Note that when our agent performs Request-Skip, it is still able to select its best guess image, and so it may still score a point for that image (as human players can).

	Fully Incremental (FI)		Partially Incremental (PI)		Non-incremental (NI)	
Image set	IT	GT	IT	GT	IT	GT
Pets	0.7	8	0.52	8	0.89	2
Zoo	0.61	8	0.58	3	0.23	4
Cocktails	0.88	8	0.48	1	0.44	10
Bikes	0.80	18	0.49	7	0.0	0

Figure 4.4: *Identification threshold (IT)* and *give-up threshold (GT)* in optimized policies for four image sets. 14 additional image sets are omitted.

scored, total time hypothetically elapsed, and thus an estimated performance in points/s for the policy. As the parameter space is tractable here, we perform grid search across possible values of IT (step .01) and GT (step 1) and select values that maximize total points/s. We carried out this optimization *for each combination of image set and incrementality type.* Our optimization accounts for when ASR results would become available in a given incremental architecture.

Perhaps the biggest concern with this approach is that it assumes that human Directors, when interacting with the agent, would produce similar utterances to what they produced when interacting with a human Matcher. We have two reasons for believing this is true enough. First, as discussed in Section 4.2, the Matcher's utterances in human-human gameplay typically play a limited role in changing the Director's descriptions. Second, our results in live human-agent interactions, reported in Section 4.6, confirm that high performance can be achieved under this assumption.

In Figure 4.4, the learned values for IT and GT are compared over four sample image sets (from among the 18 that are trained) in various incrementality conditions. An interesting observation is that *the optimized dialogue policy changes as the incrementality type changes.* For example, the FI policy for pet images (depicted in

	Fully Incremental		Partially Incremental		Non-Incremental		Human	
	Points/s	points	Points/s	points	Points/s	points	Points/s	points
Pets	0.185	182	0.151	188	0.151	154	0.069	227
Zoo	0.220	203	0.184	196	0.177	193	0.154	243
Cocktails	0.118	153	0.103	137	0.102	172	0.124	237
Bikes	0.077	126	0.073	147	0.071	100	0.072	223

Figure 4.5: Offline policy evaluation results for all three incrementality types and four image sets. 14 additional image sets are omitted for space reasons.

Figure 4.1) will wait up to 8 seconds (GT) for the confidence to reach 0.7 or higher (IT). The NI policy, on the other hand, will give up if confidence does not reach 0.89 within 2 seconds. Intuitively, one reason these policies can vary is that an ability to understand and respond incrementally can reduce the risk associated with waiting for additional user speech and ASR results. In the PI and NI versions, once the user begins to speak, the agent must wait for the user to complete their (possibly long) utterance before it can assess the (possibly unhelpful) new information and respond. The decision to let the user speak is therefore relatively heavyweight. In the FI version, the agent always has the option to listen to a little more speech and reconsider.

4.4.1 Offline Policy Evaluation Results

Our eavesdropper framework allows policies to not only be trained, but also evaluated in offline simulation, both in terms of total points scored and total points/s (which is the direct optimization metric). An excerpt from our offline evaluation results, using hold-one-user-out cross-validation, is shown in Figure 4.5. In these offline results, the agent is sometimes able to achieve higher points/s than our human Matchers did in human-human gameplay. This is true for some image sets in all three incrementality types. In general, we also observe that simulated points/s decrease as the level of incrementality in the system decreases. Note that the total number of simulated points achieved by these policies is generally less than what human players scored; the agents optimized for points/s are less likely to score a point for each image, but make up for this in speed. These offline results led us to hypothesize that, in live interaction with users, the FI agent would score higher than the less incremental versions in a time-constrained game.

4.5 Online Human-Agent Study

The data was collected using crowd-sourcing from 125 participants (discussed in 3.6). Of the 125 participants, 50 were paired with each other (forming 25 human-human pairs) and 25 were paired with each of the FI, PI, and NI agents. None participated in our study more than once. From self-disclosure of the Directors, 50% were female, all were over 18 (mean age 31.01, std. 10.13), and all were native English speakers.

After each game, participants answered a questionnaire that included basic demographic questions and also asked for their judgments on various aspects of interaction with their partner.

4.6 Human-Agent Evaluation Results

In this section, we summarize our user study results, many of which are visualized in Figure 4.6. We evaluate our FI, PI, and NI agents by game score and by user's perceptions as captured in post-game questionnaires. Users responded to a range of statements with answers on a five point Likert-scale ranging from *Totally disagree* (0) to *Totally agree* (4). We compare the responses of the Director in human-human

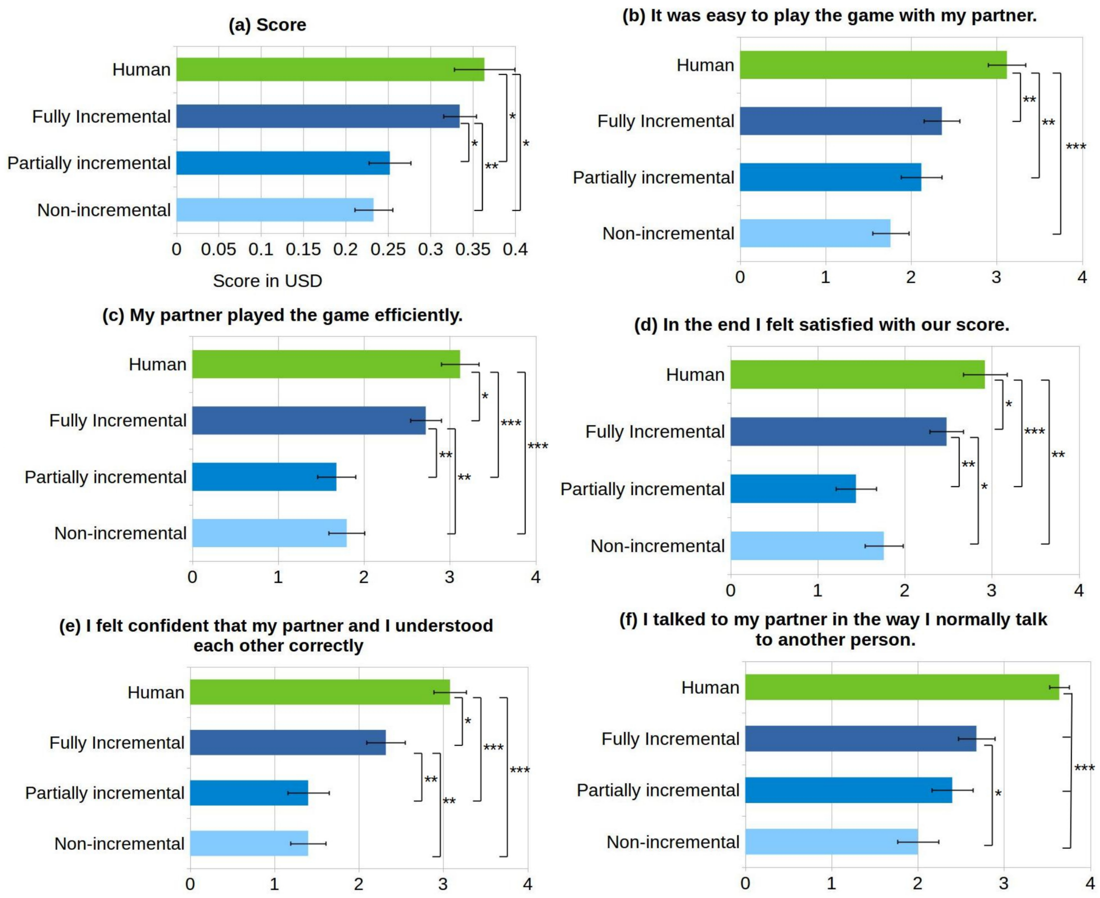


Figure 4.6: Scores and survey responses by condition (means and standard errors). Significant differences in Wilcoxon rank sum tests are indicated by * ($p < 0.05$), ** ($p < 0.005$), and *** ($p < 0.0005$).

(HH) pairs to the responses of human Directors playing with our agent as Matcher. All significance tests in this section are Wilcoxon rank sum tests.

Score (Figure 4.6a)). We report scores in U.S. dollars paid to participants for correct TIs ($0.02/correct TI). The FI system achieved a mean score of $0.33 that is significantly better than the mean $0.25 for PI (p=0.01324) and the mean $0.23 for NI (p = 0.001999). No significant difference in score was observed between the PI and NI versions. These results suggest that, beyond incorporating online decoding in the ASR to reduce ASR latency, also incorporating an incremental NLU+policy is important to score maximization.

Our FI agent's performance in terms of score is quite strong, and comparable to HH scores. Although the mean HH score of \$0.36 was a little higher than that of our FI agent (\$0.33), the difference is not significant. The best FI score of \$0.50 is higher than 76% of HH teams, and its worst score of \$0.14 is higher than 20% of HH teams. HH teams scored significantly higher than the PI ($p = 0.03766$) and NI ($p = 0.00755$) versions of the system, which suggests the importance of pervasive incremental processing to achieving human-like performance in some dialogue systems.

Satisfaction with score (Figure 4.6d). Human participants were significantly more satisfied with their score when working with a human Matcher than with any version of our agent (for the FI version, $p = 0.0372$). Participants who played with the FI agent were significantly more satisfied with their score than those in the PI ($p = 0.001863$) and NI ($p = 0.0172$) conditions. These results generally mirror our findings for game score, and score and score satisfaction are clearly connected.

Perceived ease of gameplay (Figure 4.6b). Human partners were perceived as significantly easier to play with than all agent versions. We observed a trend (not quite significant) for people to consider it easier to play with the FI version than with NI version ($p = 0.05177$).

Perceived efficiency (Figure 4.6c). Human partners were rated as significantly more efficient than the FI ($p = 0.03833$), PI ($p = 0.000064$) and NI ($p = 0.000073$) agents. Among the agent versions, the FI agent was rated significantly more efficient than PI ($p = 0.0009283$) and NI ($p = 0.002189$). This result echoes previous findings of increases in perceived efficiency for incremental systems, though here with a differing system architecture and task (Skantze & Hjalmarsson (2010)).

Perceived understanding of speech (Figure 4.6e). Human partners elicited the most confidence that the two players were understanding each other. This perceived understanding of each other's speech was significantly higher in FI than in PI (p = 0.009997) and NI (p = 0.006177). It is interesting to consider that the NLU in these three versions is identical, and thus the level of actual understanding of user speech should be similar across conditions. We speculate that the greater responsiveness of the FI system increased confidence that users were being understood.

Perceived naturalness of user speech (Figure 4.6f). One of our survey items investigated whether people felt they could speak naturally to their partner, in "the way I normally talk to another person". Human partners scored significantly higher than all agent versions here. The FI agent scored significantly higher than the NI agent (p = 0.03748).

4.7 Contributions

In this chapter, I presented the design, training, and evaluation of a high-performance agent that plays the RDG-Image game in the Matcher role. Our policy training approach allows the system to be optimized based on its specific incremental processing architecture. In a live user evaluation, three agent versions utilizing different degrees of incremental processing were evaluated in terms of game performance and user perceptions. This is the first such study in the literature to show that the most fully incremental agent achieves game scores that are comparable to those achieved in human-human game-play, are higher than those achieved by partially and non-incremental versions, and are accompanied by improved user perceptions of efficiency, understanding of speech, and naturalness of interaction.

The work presented in this chapter has been published in Paetzel et al. (2015). In the next chapter, I will discuss how I applied reinforcement learning to improve the policy developed in this chapter.

Chapter 5

Incrementality with Reinforcement Learning

> *"It doesn't matter how beautiful your theory is, it doesn't matter how smart you are. If it doesn't agree with experiment, it's wrong."*
>
> – Richard P. Feynman, *Theoretical Physicist*

In this chapter, I explore utilizing Reinforcement Learning (RL) algorithms for modeling incrementality in the RDG-Image domain. RL approaches are scalable as they have the potential to automatically learn a policy similar or better than the policy presented in Chapter 4. In this chapter, I explore the application of RL towards learning a dialogue policy and compare it to the very strong baseline developed in Chapter 4.

5.1 Introduction

Reinforcement Learning (RL) is a popular approach for dialogue management (see e.g., Henderson et al. (2008); Georgila & Traum (2011)). RL provides a framework for learning new policies from data compared to rule-guided approaches that require comparatively more significant human design and authoring. RL has been applied to learning policies in incremental scenarios and has been shown to achieve promising results (Dethlefs et al. (2012); Selfridge et al. (2013); Khouzaimi et al. (2016)). However, past works have not compared RL approaches to strong baselines.

In this chapter, I present the following contributions: provide an RL method for incremental dialogue processing that performs better in offline simulations (based on real user data) than the high-performance CDR baseline (based on carefully designed rules) presented in Chapter 4. As we saw, this is a very strong baseline which has been shown to perform very efficiently (nearly as well as humans) in this dialogue game (Chapter 4). I also show that in an interaction study with humans a version of Eve that uses the RL policy is qualitatively perceived as better by users compared to the CDR version of Eve. In many studies that use RL for dialogue policy learning, the focus is on the RL algorithms, the state-action space representation, and the reward function. As a result, the rule-based baselines used for comparing the RL policies against are not as carefully engineered as they could be, i.e., they are not the result of iterative improvement and optimization using insights learned from data or user testing. This is understandable since building a very strong baseline would be a big project by itself and would detract attention from the RL problem. In our case, there was a pre-existing strong CDR baseline policy which inspired us to investigate whether an RL policy could outperform it. One of our main contributions is that we provide a detailed comparison of the RL policy and the CDR baseline policy, including information about how much effort and time it took to develop each one of them. We also highlight the cases where the RL policy performs better, and show that understanding the RL policy can provide valuable insights which can inform the creation of an even better rule-based policy.

5.2 Eve's Dialogue Policy

In Chapter 4 we saw that Eve's policy decides between waiting and interrupting the user with As-I or As-S to maximize the score in the game. Eve does this by taking three actions: i) WAIT: Listen more in the hope that the user provides more information; ii) As-I: Make the selection and request the next TI; iii) As-S: Skip the current image set and request the next TI as it might not be fruitful to wait more. Eve's policy depicted in Algorithm 1 (see Chapter 4), uses two threshold values namely identification threshold (IT) and give-up threshold (GT) to select these actions. The IT and GT values are learned separately for each image set. The IT learned is the least confidence value (P_t^*) above which the agent uses the As-I action. GT is the maximum time the agent should WAIT before giving up on the current image set and requesting the human Director to move on to the next TI. As we discussed in Chapter 4, the IT and GT values are learned using an offline policy optimization method called the eavesdropper simulation, which performs an exhaustive grid search to find the optimal values of IT and GT for each image set. In this simulation, the agent is trained offline on the HH (human-human) conversations and learns the best values of IT and GT, i.e., the values that result in scoring the maximum points in the game. For example, the optimal values learned for the image set shown in Figure 5.6 were IT=0.8 and GT=18sec.

The Eve agent is very efficient and carefully engineered to perform well in this task, and serves as a very strong baseline. In the real user study reported in Paetzel et al. (2015), Eve in the HA (human-agent) gameplay scored nearly as well as human users in the HH gameplay. Thus this study provides an opportunity to compare an RL policy with a strong baseline policy that uses a hand-crafted carefully designed rule structure (CDR baseline). Figure 5.6 shows an example from the HA corpus. The data used in the current work comes from both the HH and HA datasets (see Table 5.1).

Branch	# users	# sub-dialogues
Human-Human lab	64	1485
Human-Human web	196	5642
Human-Agent web	175	7393

Table 5.1: Number of users and number of TI subdialogues used for our study.

5.2.1 Improving NLU with Agent Conversation Data

Obviously, the success of the agent heavily depends on the accuracy of the NLU module. We investigated whether using HA data would improve the NLU accuracy or not. Using data from all of the human Director's speech for all the TIs in the HH branch only, the NLU accuracy was found to be 59.72%. Using data from the HA branch only resulted in a lower NLU accuracy of 48.70%. Combining the HH and HA training data resulted in a higher accuracy of 61.89%. The improvement associated with training on HH and HA data is significant across all sets of images[1]. Thus, in this work we use the best performing NLU with the data trained from both the HH and HA subsets of the corpus. The overall reported NLU accuracy was averaged across all the image sets. The NLU module was trained with the same method as in Chapter 4. Note that for all our experiments, 10% of the HH data and 10% of the HA data was used for testing, and the rest was used for training. Figure 5.1 shows an example of the NLU confidence values for the human Director's description as a function of time.

[1] All the significance tests are performed using student's t test.

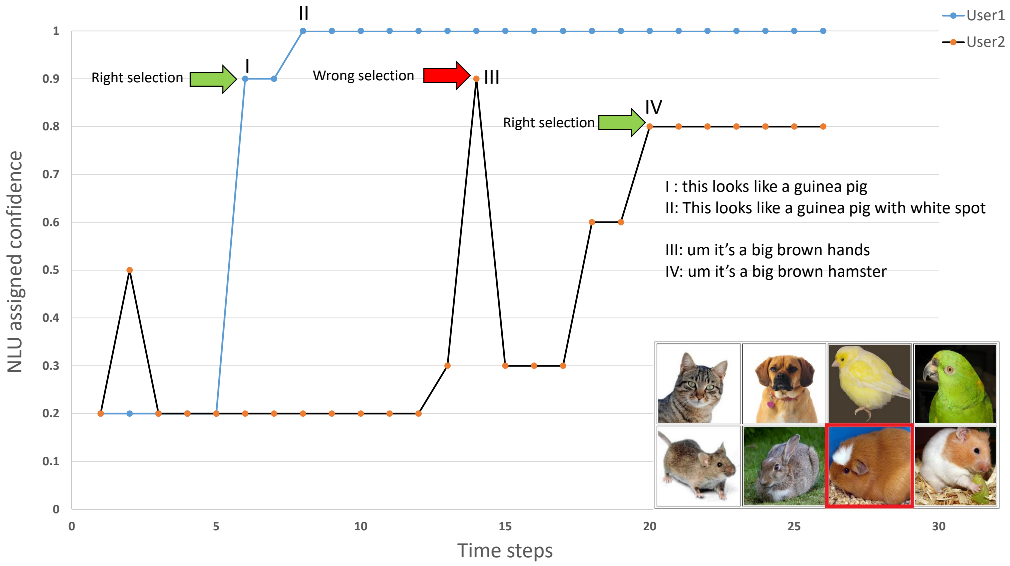


Figure 5.1: The NLU confidence curve for the human Director's descriptions for the TI highlighted with a red border.

5.2.2 Room for Improvement

Though the baseline agent is impressive in its performance there are a few shortcomings. We investigated the errors being made by the baseline policy and identified four primary limitations in its decision-making. Examples of these limitations are shown in Figure 5.2, depicting the NLU assigned confidence (y-axis) for the human TI descriptions plotted against the time steps (x-axis).

First, the baseline commits to As-I as soon as the confidence reaches a high enough value (IT threshold), or As-S when the time consumed exceeds the GT threshold. In Case 1 the agent decides to skip (As-S) because the time consumed has exceeded the GT threshold, instead of waiting more which would allow for a more distinguishing description to come from the human Director.

Second, its performance can be negatively affected by instability in the partial ASR results. Examples of partial ASR results are shown in Figure 5.8. In Case 2,

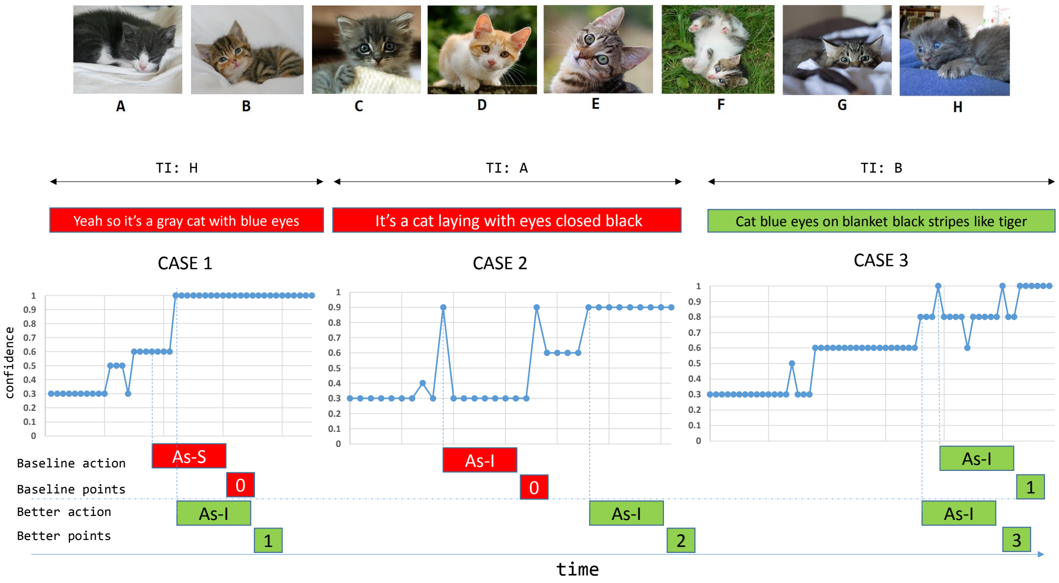


Figure 5.2: Examples where the agent can do better. The red boxes show the wrong selection by the agent.

the agent could learn to wait for higher time intervals as the ASR partial outputs become more stable.

Third, the baseline only commits at high confidence values. Case 3 shows an instance where the agent can save time by committing to a selection at a much lower confidence value.

Fourth, as we can see from Algorithm 1 presented in Chapter 4, the baseline policy does not use "combinations" (or joint values) of time and confidence to make detailed decisions.

Perhaps using RL can not only help the agent learn a more complex strategy but could also provide insights into developing a better engineered policy which would not have been intuitive for a dialogue designer to come up with. That is, RL could potentially help in building better rules that would be much easier to incorporate into the agent and thus improve its performance. For example, is there a combination of time and confidence which is an effective As-I strategy, i.e., not committing at some initial time slices for high confidence values and committing at lower confidence values as the user consumes more time?

5.3 Design of the RL Policy

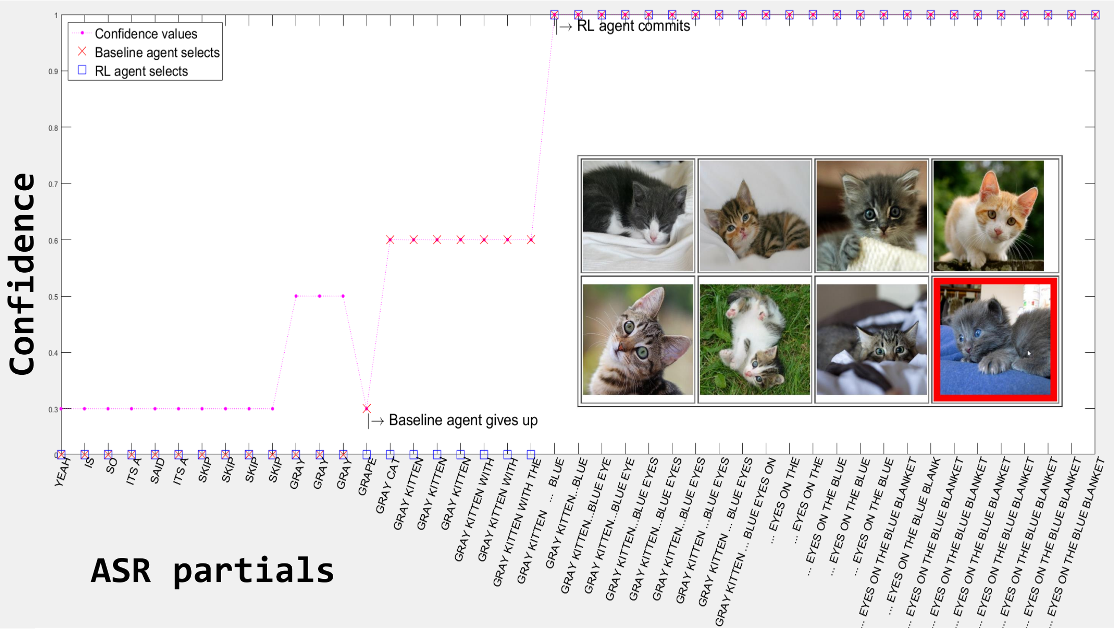


Figure 5.3: Actions taken by the CDR baseline and the RL agent.

The incremental policy decision making is modeled as an MDP (Markov Decision Process), i.e., a tuple (S, A, TP, R, γ). S is a set of states that the agent can be in. In this task S is represented by (P_t^*, t_c) features where P_t^* is the highest confidence score assigned by the NLU for any image in the image set $(P_t^* \longmapsto \mathbb{R}; 0 \leq P_t^* \leq 1)$ and t_c is the time consumed for the current TI $(t_c \longmapsto \mathbb{R}; 0 \leq t_c \leq 45.0)^2$. The RL learns a policy π mapping the state (S) to the action (A), $\pi : S \rightarrow A$, where $A = \{$As-I, As-S, WAIT$\}$ are the actions to be performed by the agent to maximize the overall reward in the game. The As-I and As-S actions map to their corresponding

[2] Each round lasts a maximum of 45 seconds.

utterances. The WAIT action corresponds to the agent listening to the user without interrupting. R is the reward function and γ a discount factor weighting long-term rewards. TP is the set of transition probabilities after taking an action.

When the agent is in state $S_t = (P_t^*, t_c)$, executing the WAIT action results in moving to state S_{t+1} which corresponds to a new d_{t+1} which corresponds to the new utterance and thus yielding new P_t^* and t_c for the given episode. The As-I and As-S actions result in goal states for the agent[3]. Separate policies are trained per image set similar to the baseline. The difference between the As-I and As-S action is in the rewards assigned. The reward function R is as follows. After the agent performs the As-I action, it receives a high positive reward for the correct image selection and a high negative penalty for the wrong selection. This is to encourage the agent to learn to guess at the right point of time. There is a small positive reward of δ for "WAIT" actions, to encourage the agent to wait before committing to As-I selections. No reward is provided for the As-S actions. This is to discourage the agent from choosing to skip and scoring the points by chance, and at the same time not penalize the agent for wanting to skip when it is really necessary. The reward function for As-S prevents the agent from getting heavy negative penalties in case the wrong images are selected by the NLU.

$$R = \begin{cases} +\delta & \text{if action is WAIT} \\ +100 & \text{if As-I is right} \\ -100 & \text{if As-I is wrong} \\ 0 & \text{if action is As-S} \end{cases}$$

[3] The time shown in the reported results as consumed by the RL policy for the current TI includes a time offset of 750ms which is the sum of 250ms for the agent to say "got it" and 500ms for the Director to request a new TI. This is included to maintain consistency with the baseline.

In this work we use the Least Squares Policy Iteration (LSPI) (Lagoudakis & Parr (2003)) RL algorithm implemented in the BURLAP[4] java code library to learn the optimal policy. LSPI is a sample efficient model-free off-policy method that combines policy iteration with linear value function approximation. LSPI in our work uses State-Action-Reward-State (SARS) transitions sampled from the human interactions data (HH and HA). We use Gaussian radial basis value function (RBF) representation for the confidence (P_t^*) and time consumed (t_c) features. We treat the state features as continuous values. The confidence values and time consumed values are continuous in nature within the bounds defined i.e., $0.0 \leq P_t^* \leq 1.0$ and $0.0 \leq t_c \leq 45.0$. The basis function returns a value between 0 and 1 with a value of 1 when the query state has a distance of zero from the function's "center" state. As the state gets further away, the basis function's returned value degrades to a value of zero.

We run LSPI with a discount factor of 0.99 until convergence occurs or a maximum of 50 iterations is reached, whichever happens first. We use 250k available SARS transitions from the HH and HA interactions to train the policy. The LSPI returns a Greedy-Q policy which we use on the test data.

Figure 5.3 shows the modus operandi of the policy in this domain. For every time step the ASR provides a 1-best partial hypothesis for the speech uttered by the test user. This partial speech hypothesis is input to the NLU module which returns the confidence value (P_t^*). The time consumed (t_c) for the current TI is tracked by the game logic.

	pets		zoo		kitten		cocktail		bikes		yoga		necklace	
	PPS	P	PPS	P	PPS	P	PPS	P	PPS	P	PPS	P	PPS	P
Baseline	0.22	37	0.28	27	0.14	14	0.18	23	0.09	13	0.20	3	0.20	4
RL agent	0.23	39	0.31	32	0.13	16	0.19	25	0.14	22	0.11	18	0.12	20

Table 5.2: Comparison of points per second (PPS) and points (P) earned by the baseline and the RL agent on the test set.

5.4 Experimental Setup

For testing, we use the real user held out conversation data from the HH and HA datasets. The IT and GT thresholds for the baseline Eve were also retrained (Paetzel et al. (2015)) using the same data and NLU as used to train the RL policy. Figure 5.3 shows the setup for testing and comparing the actions of the RL policy and the baseline. Every ASR partial corresponds to a state. For every ASR partial we obtain the highest assigned confidence score from the NLU, use the time consumed feature from the game, and obtain the action from the policy. If the action chosen by the policy is "WAIT" then we sample the next state. For each pair of confidence and time consumed values we obtain the actions from the baseline and the RL policy separately and compare them with the ground truth to evaluate which policy performs better. Once the policy decides to take either the As-I or As-S action then we advance the simulated game time by an additional interval of 750ms or 1500ms respectively. This is to simulate the conditions in the real user game where we found that users on average take 500ms to click the button to load the next set of TIs, and the agent takes 250ms to say the As-I utterance and 1000ms to say the As-S utterance. The next TI is loaded at this point and then the process is repeated until the game time runs out for each user round.

[4] http://burlap.cs.brown.edu/

5.5 Results

The policy learned using RL (LSPI with RBF functions) performs significantly better ($p<0.01$) in scoring points compared to the baseline agent in offline simulations. Also, the RL policy takes relatively more time to commit (As-I or As-S) compared to the baseline.[5] The idea of setting the IT and GT threshold values in the baseline originally aimed at scoring points rapidly in the game, i.e., the baseline agent was optimized at scoring the highest number of points per second (PPS). The PPS parameter is a measure of how effective the agent is at scoring points overall, and is calculated as the ratio of the total points scored by the agent divided by the total time consumed. Table 5.2 shows the points per second and the total points scored in some of the image sets by the baseline and the RL. We can observe that the RL consistently scores more points than the baseline, however this comes at the cost of additional time. By scoring more points overall than the baseline, the RL also scores higher in the PPS metric ($p<0.05$). Table 5.3 shows the total points scored and the total time spent across all users by the baseline and the agent. Each set here refers to one round in a game.

	Baseline		RL	
Set	P	t (s)	P	t (s)
1	96	510.8	107	528.1
2	75	525.0	85	537.9
3	42	298.9	74	595.2
4	49	531.9	76	592.3

Table 5.3: The points scored (P) and the time consumed (t) in seconds for different image sets (Set).

Figure 5.5 depicts this result for an image set of bikes (images shown in Figure 5.6). We plot the total time spent by the agent and the total points scored.

[5] $p=0.06$; we cannot claim that the time taken is significantly higher but there is a trend.

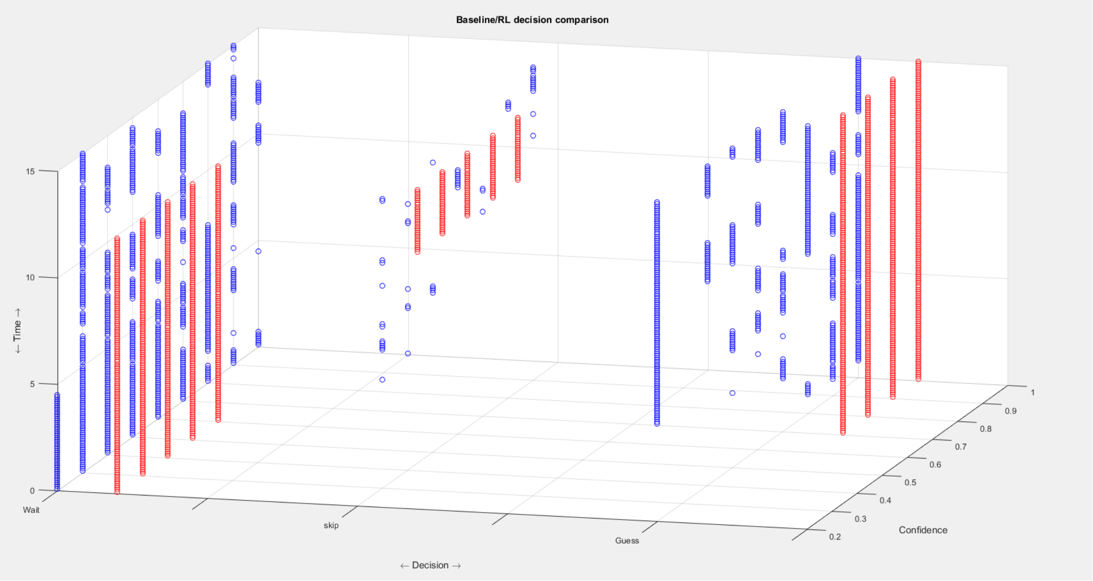


Figure 5.4: Decisions of the RL policy (in blue) vs. the baseline policy (in red).

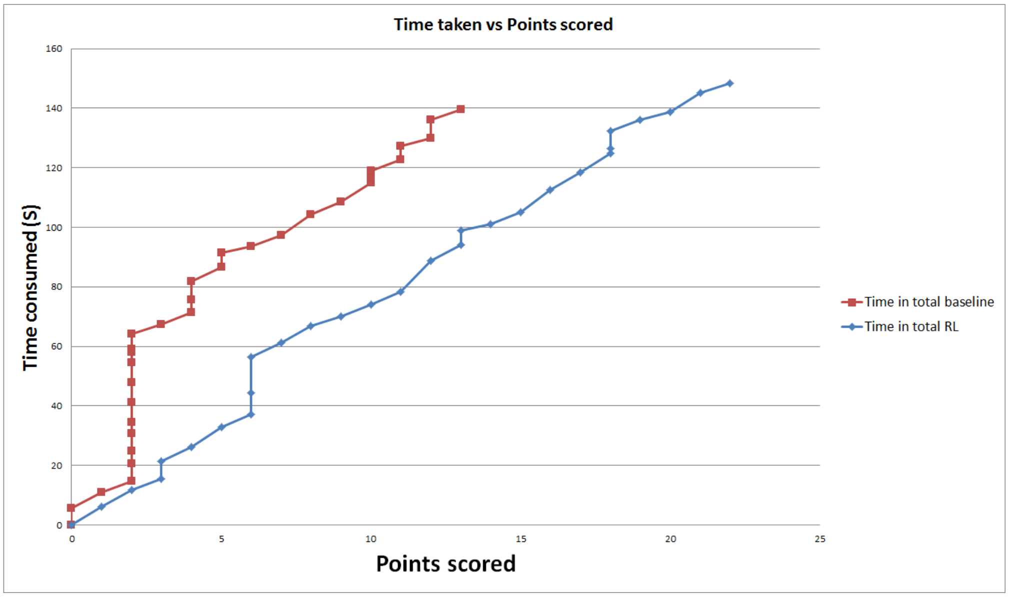


Figure 5.5: The RL policy scores significantly more points than the baseline by investing slightly more time (graph generated for one of the image sets).

Clearly, the RL policy manages to score more points than the baseline in a given amount of time. In order to understand the differences in the actions taken by the RL policy and the baseline policy, we plot on a 3 dimensional scatter plot, the

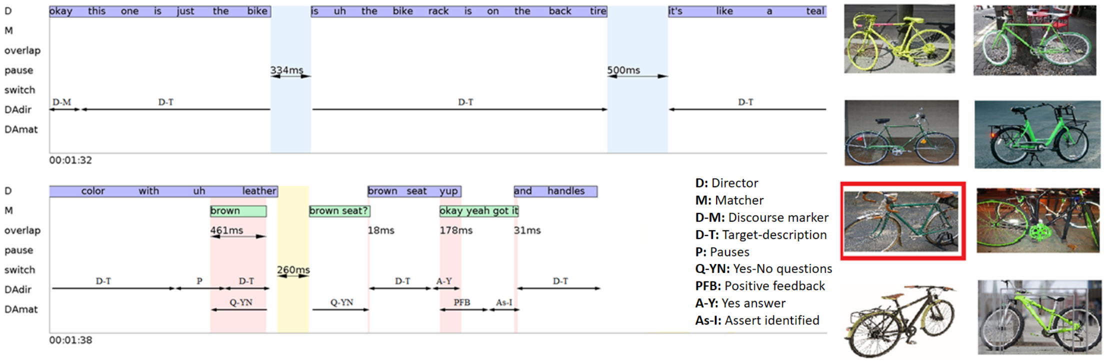


Figure 5.6: Shows an example dialogue for an episode in the human-human corpus.

action taken by the policy for confidence values between 0 and 1 (spaced at 0.1 intervals) and the time consumed between 0s to 15s (spaced at 100ms intervals) for one of the image sets (bikes). Figure 5.4 shows the decisions made by the RL (in blue) compared to the decisions made by the baseline (in red). As we can see there is not much variety in the decisions of the baseline policy; it basically uses thresholds (see Algorithm 1 presented in Chapter 4) optimized using real user data. Below we summarize our observations regarding the actions taken by the RL policy.

i) Regardless of whether the confidence value is high or low, the RL policy learns to wait for low values of the time consumed. This may be helping the RL policy to avoid the problem illustrated in Case 2 in Figure 5.2, where instability in the early ASR results for a description can lead an incorrect guess to be momentarily associated with high confidence. The RL policy is more keen on waiting and decides to commit early only when the confidence value is really high (almost 1.0). ii) Requiring a lower degree of confidence when the time consumed is high was also found to be an effective strategy to score more points in the game. Thus the RL policy learns to guess (As-I) even at lower confidence values when the time consumed reaches high values. This combination of time and confidence values helps the RL agent perform better w.r.t. points and consequently PPS in the task.

It is also important to note that the agent does not wait eternally to make its selection. The human TI descriptions are collected from real user gameplay that lasts for a limited number of time steps. That is, the maximum number of points that the RL policy can score in simulation is limited by the number of images described in the real user gameplay. In the case of the "WAIT" action beyond this point the agent fails to gather high rewards as the As-I action was never selected. By the virtue of this design feature, the RL agent has implicitly learned the notion of playing the game at a high pace.

Note also that the RL agent has not learned to always commit at a later time than the baseline. Table 5.4 shows the percentage of times (in the test games) where the RL policy chooses a different strategy than the baseline. We can see that the RL policy commits at the same time instances as the baseline about 48% of time. 44.77% of the time the baseline commits to the TI faster and about 7% of the time the RL decides to commit earlier to the TI compared to the baseline.

The cases shown in Figure 5.2 provide examples of how the RL policy can outperform the baseline in this offline study.

i) As the RL agent has learned to not commit to a decision early it can wait enough time to observe more user words and thus reach higher confidence (Case 1).

ii) The RL agent is not keen on committing when it sees an early high confidence value (like IT for the baseline) but rather waits which may enable the ASR partials to become more stable (Case 2).

iii) The RL agent also learns to commit at low confidence values as the time consumed increases and sometimes even committing earlier than the baseline (Case 3).

% times Same commit times	48.06
% times Baseline has faster commit	44.77
% times RL has faster commit	7.17

Table 5.4: Comparison of commit strategies between baseline and RL (%).

5.5.1 Contrasting Baseline and RL Policy Building Efforts

Building an SDS with carefully crafted rules has often been criticized as a laborious and time consuming exercise. This is in contrast to the alternative data oriented approaches, which are often argued to require less time to engineer a solution and be more scalable. Development of the baseline system's policy component took an NLP researcher approximately two months, including experimentation with alternative rule structures and development of the parameter optimization framework. The same amount of effort was put into developing the RL policy by a researcher with similar skills. Building the RL policy involved experimenting with various reward functions to suit the task. Though the reward function is simplistic in our case, a high negative reward for wrong As-I actions was required for RL to learn useful policies. It also takes effort and experimentation to select the right algorithm (LSPI with value function approximation vs. Vanilla Q-learning). It is thus hard to claim which approach is more time-efficient (in terms of development effort). Figure 5.7 shows a comparison of the baseline policy and the RL policy learned with the Vanilla Q-learning algorithm which did not perform well. It performed worse than the baseline. We also need to keep in mind that: i) We cannot claim that the rules learned by the RL policy could not be implemented in the hand-crafted system. Bounds on the time and confidence (for example: do not commit as soon as the confidence exceeds a threshold but rather wait for a few additional time steps, it is okay to commit at lower confidence values for higher time values to perform better, etc.) can be included in the Algorithm 1 and the system can be deployed with ease. ii) It usually takes time and effort to build a common infrastructure to experiment between the two strategies. In this case, experimenting with the incremental RL policy was simpler as the infrastructure and the methodology existed from the previous work discussed in Chapter 3 and Chapter 4. Despite the fact that both approaches required similar development effort, in the end, RL did learn a better strategy automatically, at least in our offline simulations (based on real user data). RL provides advantages compared to the baseline method. Adding new constraints into the baseline can be hard. This is because the baseline method uses exhaustive grid search to set its parameter values, and it might be exponentially costly to do this with more constraints. On the other hand, RL is more scalable as adding features is relatively easy with RL.

Until now, we have showed that RL has potential for learning policies to make incremental decisions that yield better results than a high performance CDR baseline. Our experiments were performed in simulation (albeit using real user data) and the next step is to investigate whether these improvements transfer to real time experiments (real time interaction of the agent with human users). Another interesting avenue for future work is to implement a hybrid approach of engineering a hand-crafted policy using the intuitions learned from using RL. There are still regions of the state space that were not fully explored by RL. On the other hand, as we saw, RL can potentially learn interesting policies which would not have been intuitive for a dialogue designer to come up with. Therefore, we plan to explore incorporating intuitions from the RL into the high performance CDR baseline and see which avenue would be more fruitful and if we can get the best of both worlds.

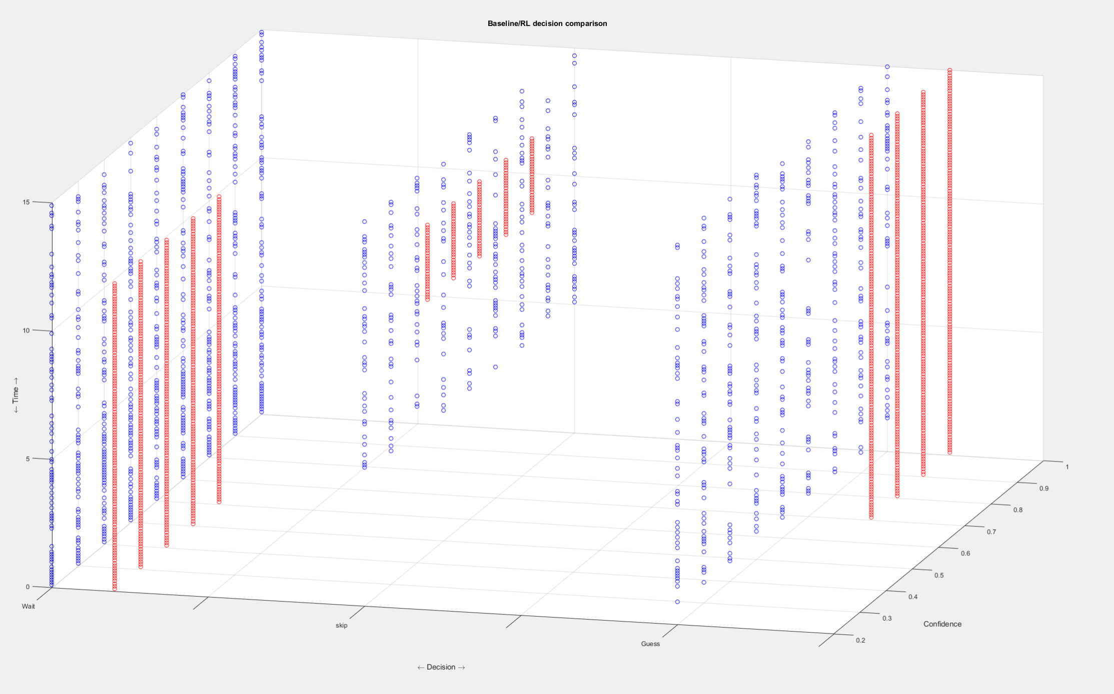


Figure 5.7: Policy learned by the Vanilla Q-learning algorithm (blue) compared to the baseline (red).

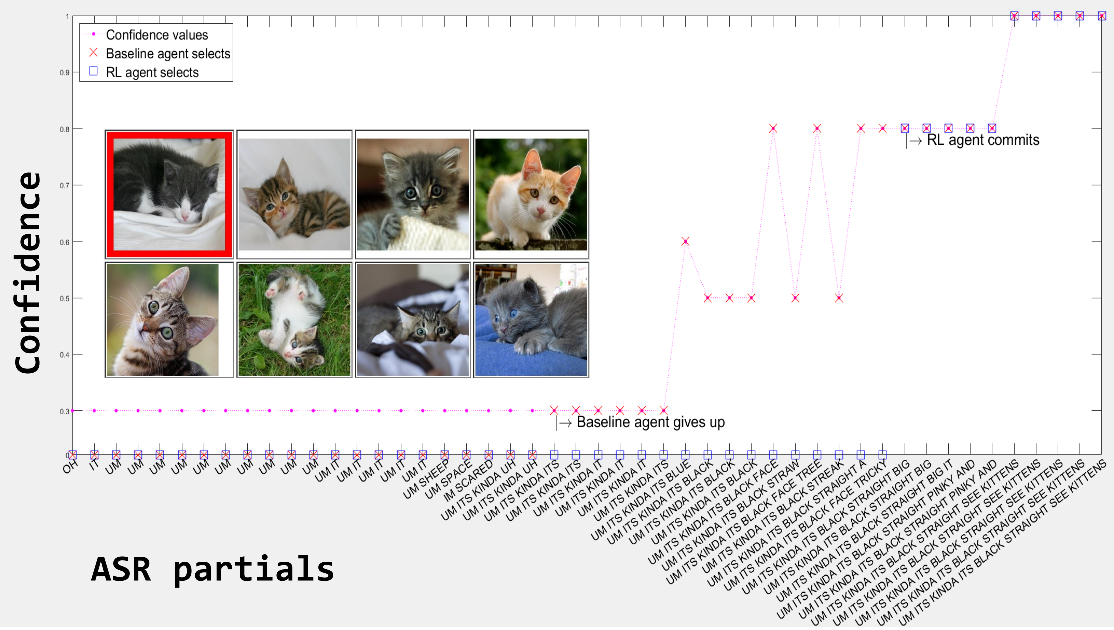


Figure 5.8: Actions taken by the baseline and the RL agent for the 1-best ASR increments. The image set is also shown.

5.6 Live Human Interaction Study

The policy automatically learned by the RL approach outperforms the baseline measured as points/second in the offline evaluation. It is important to verify the performance of such an RL policy in live interaction with humans (not just offline simulations based on real user data). In this section, I will discuss building such an agent and measuring the performance of the agent in a real user live interaction study, and compare the results with the strong baseline policy.

5.6.1 Agent Design

The agent architecture introduced in Section 3.6 is adopted to run the modified RL-based Eve. The agent ASR was replaced with Google ASR[6] service[7]. The development of the NLU and the RL policy used for the study was described earlier in the chapter (Section 5.3). No major differences were observed in the NLU confidence scores between the ASR transcripts from Kaldi and the Google ASR[8]. The baseline's dialogue policy, using the eavesdropper framework, was retrained using the same NLU results as the RL approach. The dialogue policy learned using RL was logged as a stream (file) and loaded by the agent during deployment. The agent utilizes the same utterances as the agent used for the baseline. Neospeech TTS was used for converting text to speech.

[6] https://cloud.google.com/speech-to-text/

[7] It is important to note that swapping the ASR from Kaldi to Google did not yield any significant differences in points per second results. However, the word error rate was found to be lower.

[8] Although Google ASR yielded lower word error rates, the important keywords were identified equally well by Kaldi and Google ASR. Hence, we did not observe any major differences in the NLU scores.

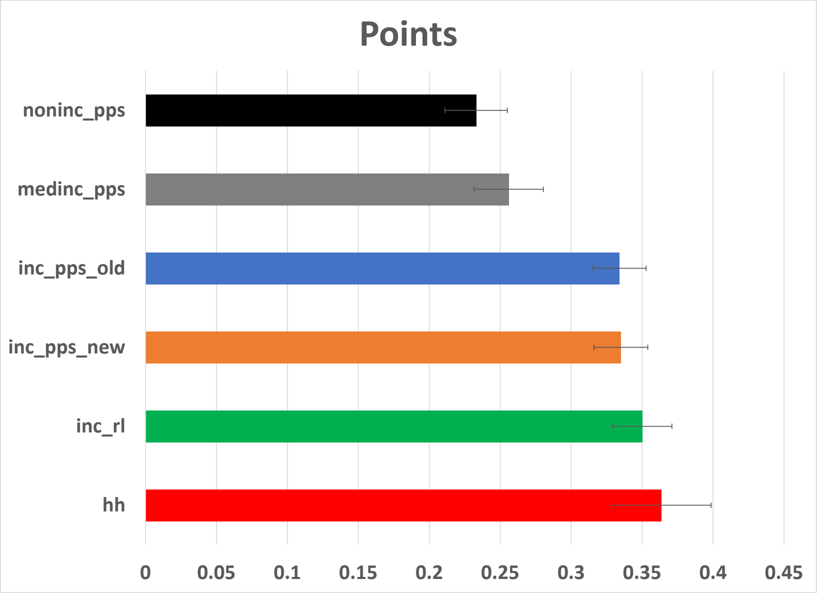
The agent was deployed as a web service and operated over the browser using a unique HTTP URL. This deployment of the agent over the web allows conducting the study using crowd-sourcing. I will now describe the study design.

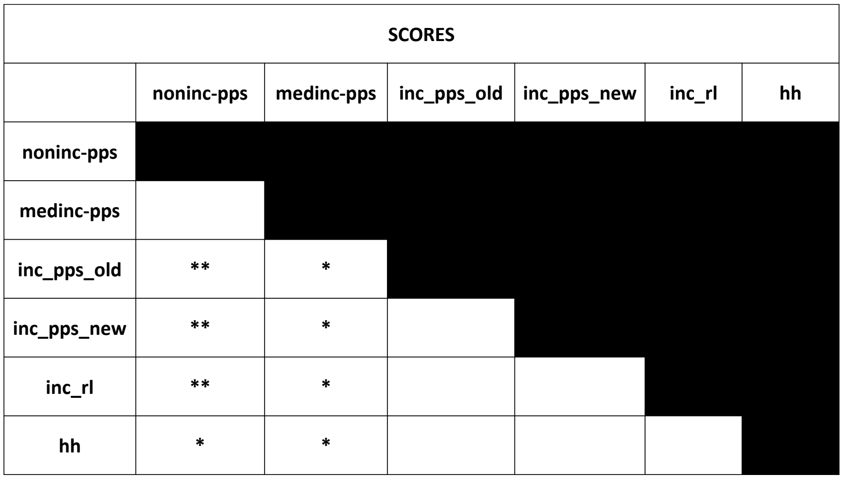
5.6.2 Study Design

The participants for the study were recruited using Amazon Mechanical Turk (AMT) following the study protocol described in Figure 3.13. The participants from AMT were required to i) have a qualification acceptance rate of 92% or higher, ii) be native English speakers, iii) be geographically located in the US or Canada, iv) have high-speed internet, and v) use either a laptop or a PC (no mobile hand-held devices were allowed). The participants self-qualified for the speed of the connection and the device. The participants were then screened to ensure proper audio equipment and setup. They were required to speak using their microphone and transcribe a small portion of speech to ensure that their audio setup was proper. They were then shown a video explaining the rules of the game. Before the actual game, the participants played a practice round, and upon completion of the practice round, they played the game by conversing with the agent. Upon completion of the game, the participants were required to answer the questionnaire asking them to rate their partner. The results are described below. We initially ran the study with the CDR baseline (25 participants per different CDR version, see below), and then deployed the RL agent (another 25 participants) and analyzed the differences.

5.6.3 Results

In this study, we compare the results obtained using the current experiment and the previously recorded observations across the following conditions. i) noninc_pps: condition with the agent running non-incrementally with the policy optimized




SCORES

	noninc-pps	medinc-pps	inc_pps_old	inc_pps_new	inc_rl	hh
noninc-pps	■	■	■	■	■	■
medinc-pps		■	■	■	■	■
inc_pps_old	**	*	■	■	■	■
inc_pps_new	**	*		■	■	■
inc_rl	**	*			■	■
hh	*	*				■

Figure 5.9: Difference in points scored in different settings of the agent and the human-human condition. (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$)

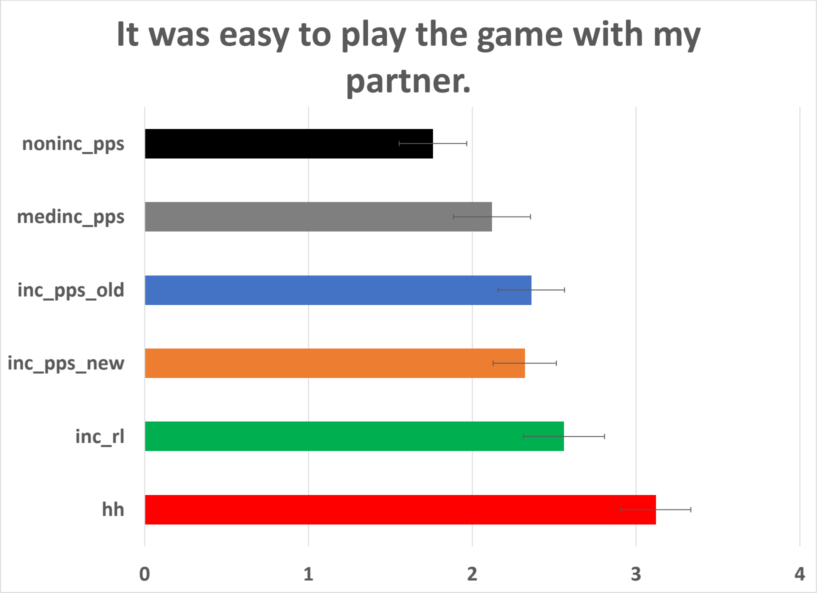
on points per second as described in Chapter 4. The scores reported are for the previously conducted experiments in Chapter 4. ii) medinc_pps: condition with the agent running in the partially incremental mode with the policy optimized on points per second as described in Chapter 4. The scores reported are for the previously conducted experiments in Chapter 4. iii) inc_pps_old: condition with the agent running with a fully incremental model with the policy optimized on points per second as described in Chapter 4. The scores reported are for the previously conducted experiments in Chapter 4. iv) inc_pps_new: condition with the agent running with a fully incremental model, with the new ASR and retrained NLU model, and with the policy optimized on points per second. The scores for this condition are obtained using the experiments run currently. v) inc_rl: condition with the agent running with a fully incremental model and with the policy optimized using RL (same ASR and NLU models as for the inc_pps_new condition). The scores for this condition are obtained using the experiments run currently. vi) hh: human-human conversation gameplay recorded earlier in a similar setting. All the significance tests were conducted using the Wilcoxon rank-sum test.

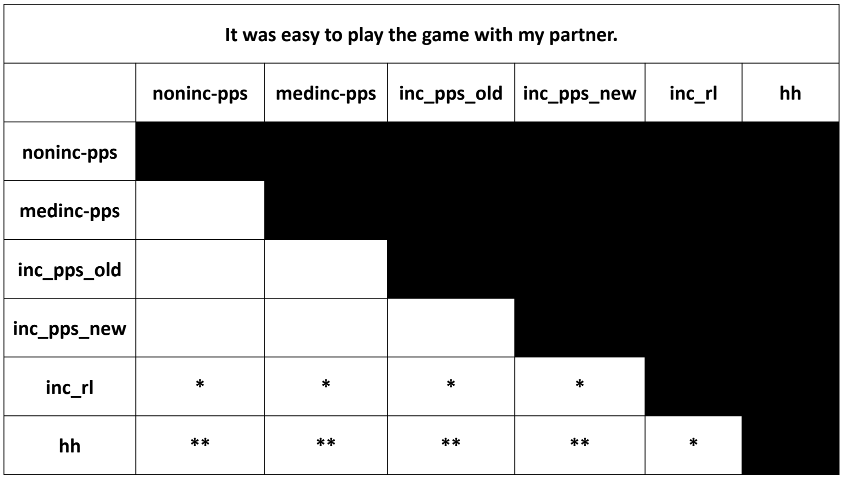
Quantitative comparison

Scores comparison. We measure the differences between the points scored by the agents across different conditions and humans. We find that there are no significant differences between the scores of the human-human condition and the inc_pps_new and inc_rl conditions. However, the mean scores by the agent in the RL condition are higher than the mean scores by the agent in the CDR condition. Figure 5.9 shows the comparison of scores in different conditions and the significance differences across different conditions.

Qualitative comparison

For measuring qualitative differences between the agents across different conditions, we ask the users to respond to a 5-point Likert scale questionnaire. We ask

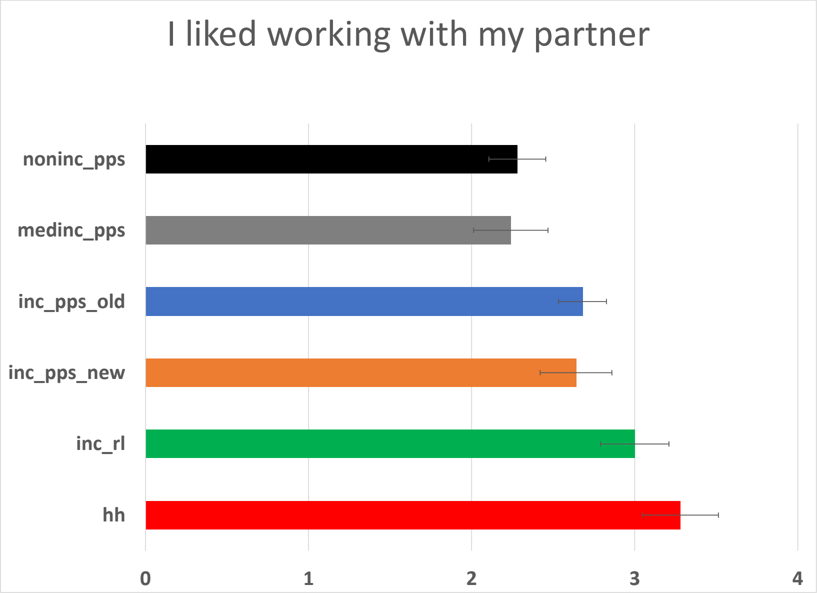




It was easy to play the game with my partner.						
	noninc-pps	medinc-pps	inc_pps_old	inc_pps_new	inc_rl	hh
noninc-pps						
medinc-pps						
inc_pps_old						
inc_pps_new						
inc_rl	*	*	*	*		
hh	**	**	**	**	*	

Figure 5.10: Mean values of user responses and standard error bars for the question 'It was easy to play the game with my partner' as rated by the participants. Here we see that users found it easy to play the game in the RL condition (significantly higher ratings than the other agent conditions). Users found it easiest to play the game with other humans. (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$)

the users to rate if they 'strongly disagree', 'disagree', are 'neutral', 'agree', and 'strongly agree' to the statements.

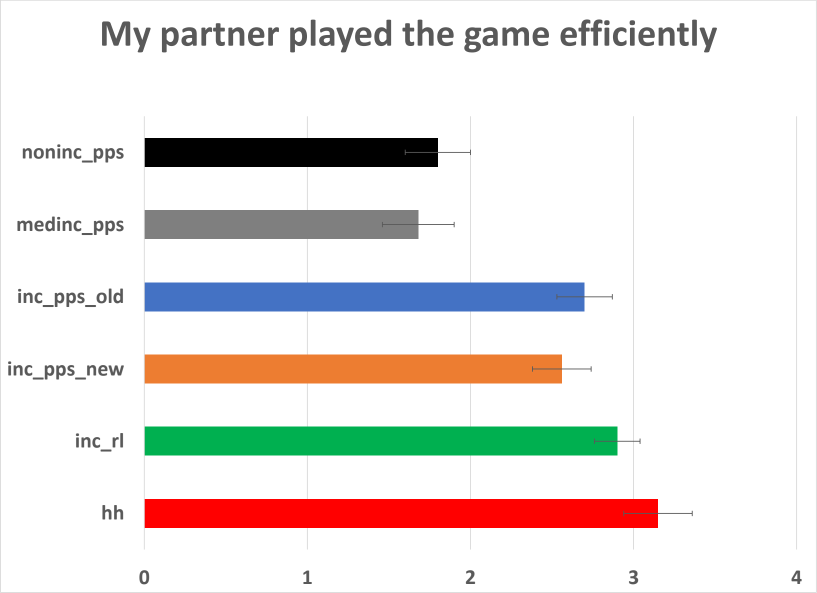


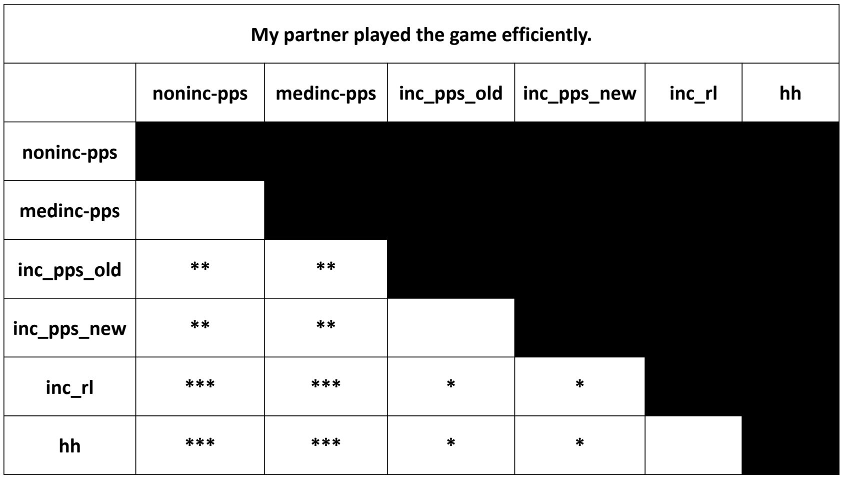
I liked working with my partner.

	noninc-pps	medinc-pps	inc_pps_old	inc_pps_new	inc_rl	hh
noninc-pps						
medinc-pps						
inc_pps_old						
inc_pps_new						
inc_rl	**	**	**	**		
hh	***	***	***	***	*	

Figure 5.11: Mean values of user responses and standard error bars for the question 'I liked working with my partner' as rated by the participants. Here we find that users rated the RL condition higher compared to the other agent conditions. Humans still outperform all the agent conditions. (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$)

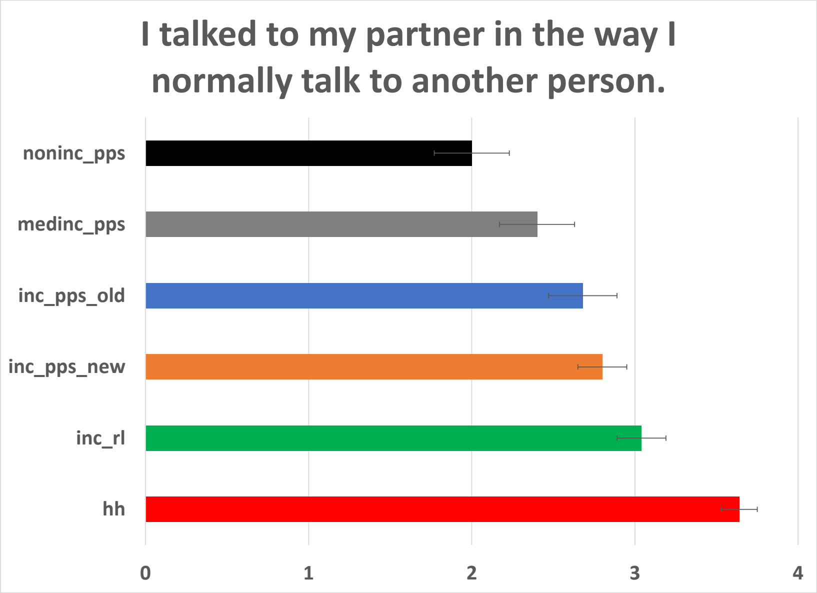
Easy to play. For this measure, we ask how easy they felt playing the game with their partner. The question is purposefully kept general to measure the relative

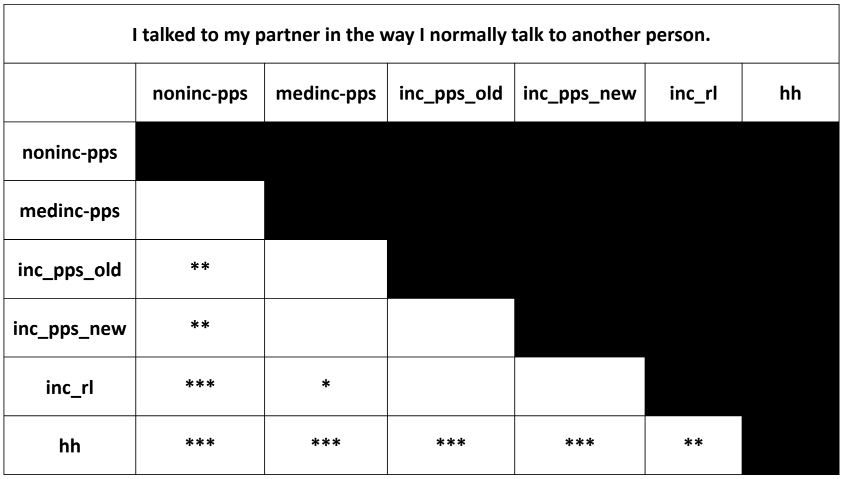




My partner played the game efficiently.						
	noninc-pps	medinc-pps	inc_pps_old	inc_pps_new	inc_rl	hh
noninc-pps						
medinc-pps						
inc_pps_old	**	**				
inc_pps_new	**	**				
inc_rl	***	***	*	*		
hh	***	***	*	*		

Figure 5.12: Mean values of user responses and standard error bars for the question 'My partner played the game efficiently' as rated by the participants. Here we find that users felt that the humans played the game significantly more efficiently than the agents. We also find that users felt that the RL agent played the game more efficiently than the other versions of the agent. Also, there is no significant difference between the RL agent and the human condition. (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$)



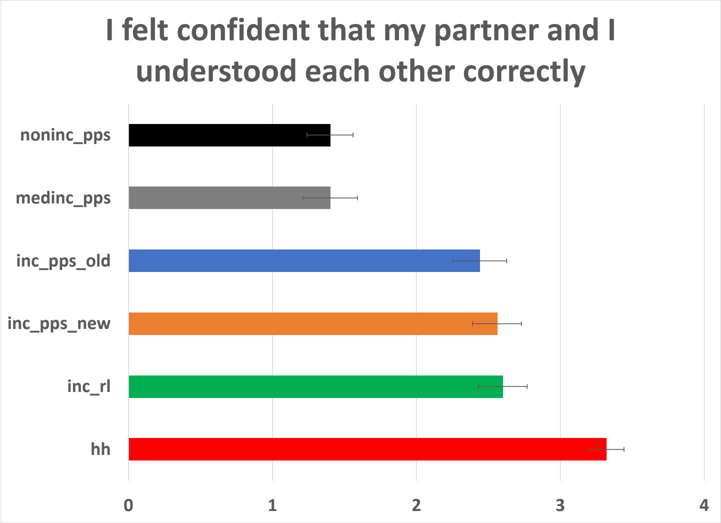


I talked to my partner in the way I normally talk to another person.

	noninc-pps	medinc-pps	inc_pps_old	inc_pps_new	inc_rl	hh
noninc-pps						
medinc-pps						
inc_pps_old	**					
inc_pps_new	**					
inc_rl	***	*				
hh	***	***	***	***	**	

Figure 5.13: Mean values of user responses and standard error bars for the question ‘I talked to my partner in the way I normally talk to another person’ as rated by the participants. We find that users rated that they spoke normally to other humans the most. There are no significant differences between the RL agent and the other fully incremental agents. (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$)

ease of playing the game with their partner. The users found it the easiest to play the game with humans. Also, the users found it relatively easier to play the game



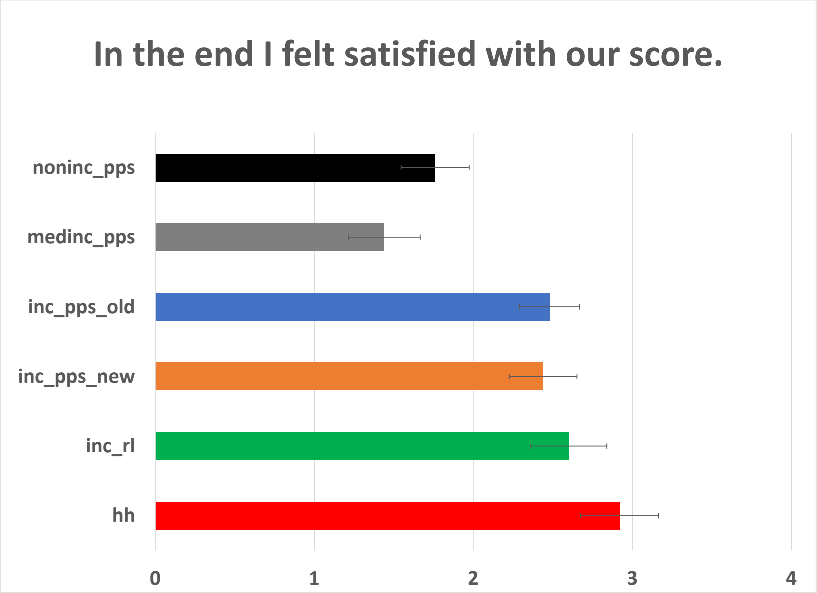
I felt confident that my partner and I understood each other correctly.

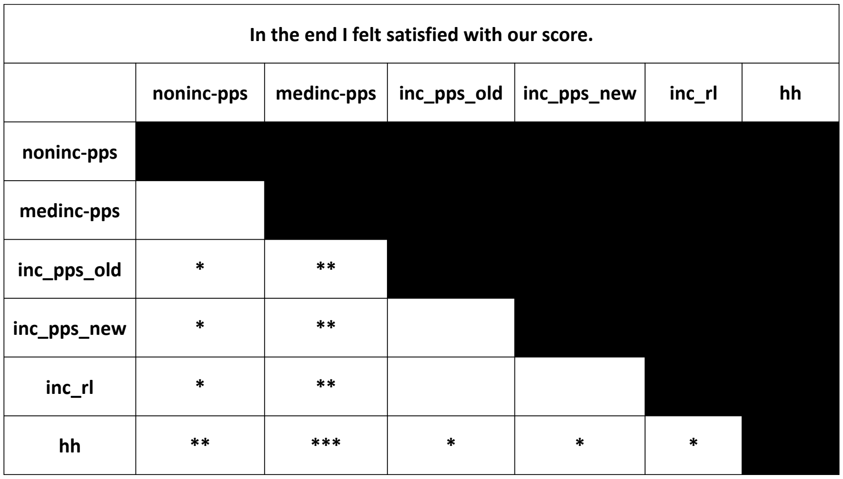
	noninc-pps	medinc-pps	inc_pps_old	inc_pps_new	inc_rl	hh
noninc-pps						
medinc-pps						
inc_pps_old	*	*				
inc_pps_new	*	*				
inc_rl	**	**				
hh	***	***	*	*	*	

Figure 5.14: Mean values of user responses and standard error bars for the question 'I felt confident that my partner and I understood each other correctly' as rated by the participants. We find that users rated humans most favorably on this metric. There are no significant differences between the RL agent and the other fully incremental agents. (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$)

with the RL version compared to the other versions of the agent (and this result is statistically significant). The results are shown in Figure 5.10.

Liked working with partner. For this measure, we ask users to rate how much they liked working with their partner during the game. This measure is created with the intention of measuring their overall like-ness of working with their





In the end I felt satisfied with our score.						
	noninc-pps	medinc-pps	inc_pps_old	inc_pps_new	inc_rl	hh
noninc-pps						
medinc-pps						
inc_pps_old	*	**				
inc_pps_new	*	**				
inc_rl	*	**				
hh	**	***	*	*	*	

Figure 5.15: Mean values of user responses and standard error bars for the question ‘In the end I felt satisfied with our score’ as rated by the participants. There are no significant differences between the RL agent and the other fully incremental agents. Users were most satisfied with their scores when playing the game with human partners. (* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$)

partner (Eve). We find that for this metric the users liked working with the humans the most. The RL condition is significantly better than the other agent conditions. The results are shown in Figure 5.11.

Played game efficiently. For this measure, we ask users to rate their perception of efficient gameplay. For this metric the users rated the human condition as more efficient. The RL agent was rated as a more efficient game-player than the other versions of the agent (and this result is statistically significant). Also, there was no significant difference between the RL condition and the human condition. The results are shown in Figure 5.12.

In these three metrics, we find that the RL agent performs better than all the previous versions of the agent. In the metrics mentioned below, we find that the RL has done as well as the other fully incremental versions of the agent, and better than the partially incremental and non-incremental versions of the agent.

Spoke normally. For this measure, we find that the users felt that they spoke naturally when conversing with other humans. We find no other significant differences between the fully incremental agents (Figure 5.13). We also find similar patterns in the **score satisfaction** question (Figure 5.15). Furthermore, we find that the users felt that the agent **understood** them equally well across the incremental conditions (Figure 5.14).

From these observations, we can conclude that the RL agent is perceived as better by the users than the baseline agents in various metrics. We, however, did not find any significant differences in the scores. However, the mean values of the scores by the RL agent are higher. The lack of significance could be due to a small number of users. For some of the ratings the RL agent was significantly better than the fully incremental baselines, and in all cases the RL agent was significantly better than the partially incremental and non-incremental baselines. It is also interesting that in terms of perceived game efficiency there was no significant difference between the RL condition and the human condition. These experiments show that RL can be used to streamline the development of an agent that has been shown to perform nearly as well as humans. In the subsequent sections, we show another advantage of utilizing RL for the development of incremental SDS, namely transfer learning. This approach helps extend building a dialogue policy for unseen images.

5.7 Transfer

The dialogue policy discussed in this chapter so far is specific to a given image set. The question that arises is: When conversational data is not available for a given image context set is it possible to leverage conversations from different image sets to build an incremental dialogue policy? In this section, we explore the development of a dialogue policy a) by leveraging an already learned policy from a different image set, and b) by utilizing visual object annotations.

Visual object annotations. Visual object annotations refer to the natural language descriptions of the objects present in the image. Figure 5.16 shows an example image with bounding boxes and labels. Bounding boxes are assigned to objects and labels correspond to each of the bounding boxes. Each such bounding box contains a description of the object labeled. Such visual object annotations for images can be obtained using visual object annotation algorithms (e.g., Kulkarni et al. (2013)) or through crowd-sourcing paradigms (e.g., Lin et al. (2014); Krishna et al. (2017)). In this work, we obtain visual object annotations through crowd-sourcing.

The images used in the RDG-Image dataset as described in the earlier sections do not contain visual object annotations. These annotations for the images in the 16 context sets are obtained using crowd-sourcing. The images were posted on

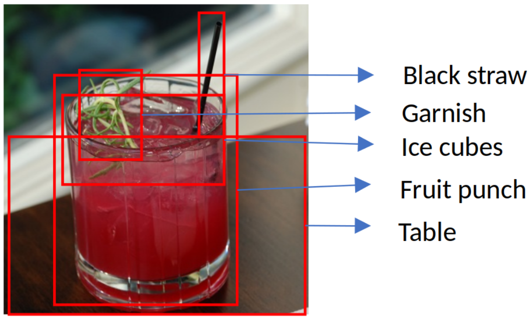


Figure 5.16: An example image with visual object annotations.

AMT and workers with a task qualification rate of >95% in a minimum of 100 tasks and within US/Canada were recruited to perform the task. Each worker provided 5 visual object annotations and each image was presented to 5 workers. These labels were then checked for accuracy by an expert. The images were annotated by the workers without the reference of the complete context set, i.e., without the other images that each image would appear with in each round of the game. Figure 5.18 shows examples of the object annotations obtained for the images in a context set.

In this section, I explore the following directions: i) Utilize the dialogue policy learned for a different image set to bootstrap the development of a dialogue policy for a new unseen image set with the goal of achieving better performance with fewer data points. ii) When no dialogue data is available, is it possible to develop an initial policy with a simulated Director which produces descriptions using visual object annotations? We explore augmenting policy learning using this simulated user (Director) based dialogue policy. iii) Compare with a policy learned without any augmentation using a randomly initialized policy as a baseline.

5.7.1 Setup

There are 16 context sets in total, with each context set consisting of 8 images. These 16 context sets are divided into two halves, the first half is used as a 'seen' image set and the other half as an ‘unseen’ image set. The augmentation part of the training process is performed using only the ‘seen’ part of the image sets. We split the ‘unseen’ human-human dialogue data into a training set (approximately 80% of the data) and a test set (20% of the data). The training set of the ‘unseen’ dataset is used to retrain the transferred policy in batches and we test the performance using the test set of the ‘unseen’ dataset.

The three different conditions under which the dialogue policy is evaluated are: i) Case 1 (NT): In this case, there is no transfer or any other kind of bootstrapping of the learning process, which basically means that we start training on the train portion of the ‘unseen’ dialogue data from scratch. This represents our baseline for comparison. ii) Case 2 (T): In this case, we develop a simulated user which utilizes visual object annotations for generating Director descriptions. These simulated dialogues are first used to learn a policy and then we use this policy to bootstrap the learning process for our ‘unseen’ image set. Thus in this case we do not use any real dialogue data in the bootstrapping process, we just use visual object annotations. iii) Case 3 (TEVE): In this case, we utilize an already learned policy (trained on the ‘seen’ human-human dialogue data). We use this policy to bootstrap the learning process for our ‘unseen’ image set. So basically the ‘seen’ human-human dialogue data is only used in the TEVE case. We will now discuss these cases in detail. The policies are learned using LSPI (Lagoudakis & Parr (2003)) and Vanilla Q-learning (Sutton & Barto (1998)). The state and action representations are as described in Section 5.3. For the ‘unseen’ image set all policies are trained on the train portion of the ‘unseen’ human-human dialogues and tested on the test portion of the ‘unseen’ human-human dialogues. But as we discussed above in Case 2 and Case 3 the policy is bootstrapped whereas in Case 1 the policy is not bootstrapped (i.e., the starting policy is random).

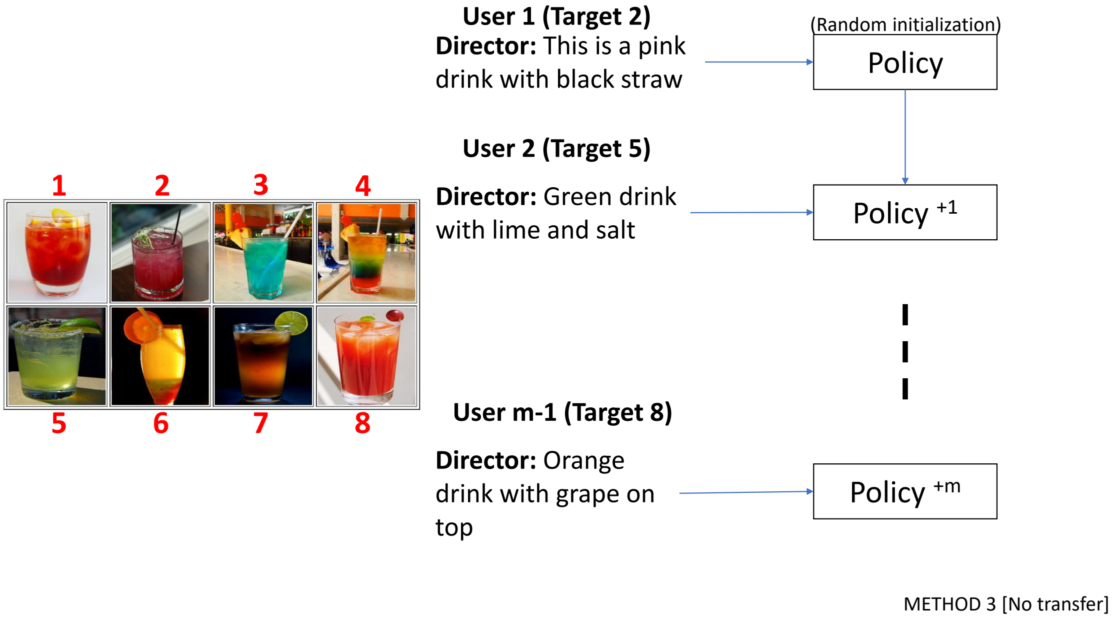


Figure 5.17: The training methodology for Case 1.

Case 1 (NT)

In Case 1, we train the policy for a given image set utilizing the conversations between the Director and the Matcher in the train portion of this image set (‘unseen’ human-human dialogue data). There is no transfer or any kind of bootstrapping of the learning process. Figure 5.17 shows the method followed to train the policy in this case. This is the base case scenario where we train the dialogue policy utilizing the conversations between the human Director and the human Matcher incrementally in batches. In this case, we use the descriptions from the human Director describing the target image to train the policy.

Case 2 (T)

In recent times visual image object annotation datasets such as ImageNet (Deng et al. (2009)), MS COCO (Lin et al. (2014)), and VisualGenome (Krishna et al. (2017)) have proven to be extremely useful for computer vision tasks such as image

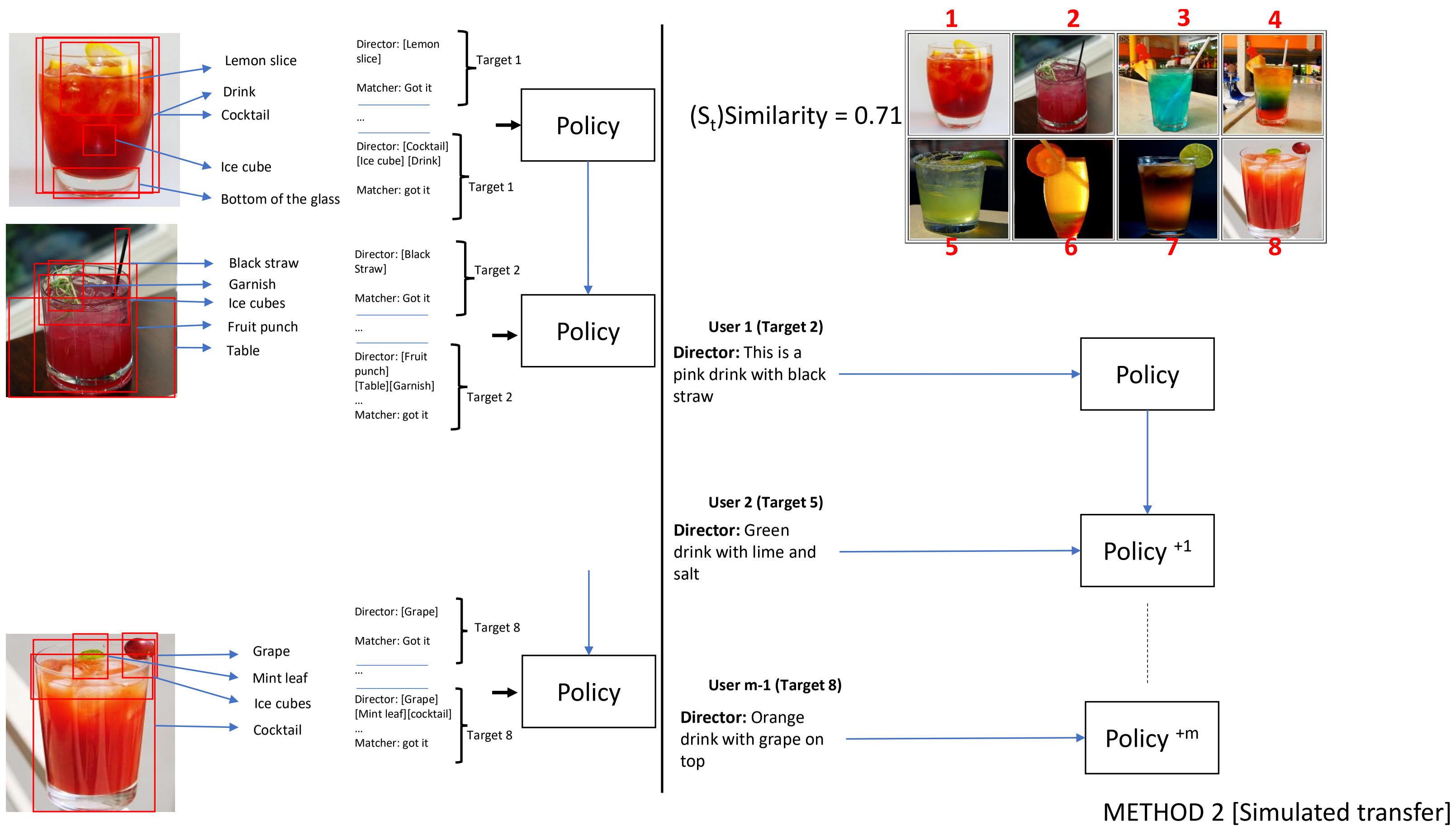


Figure 5.18: The training methodology for Case 2.

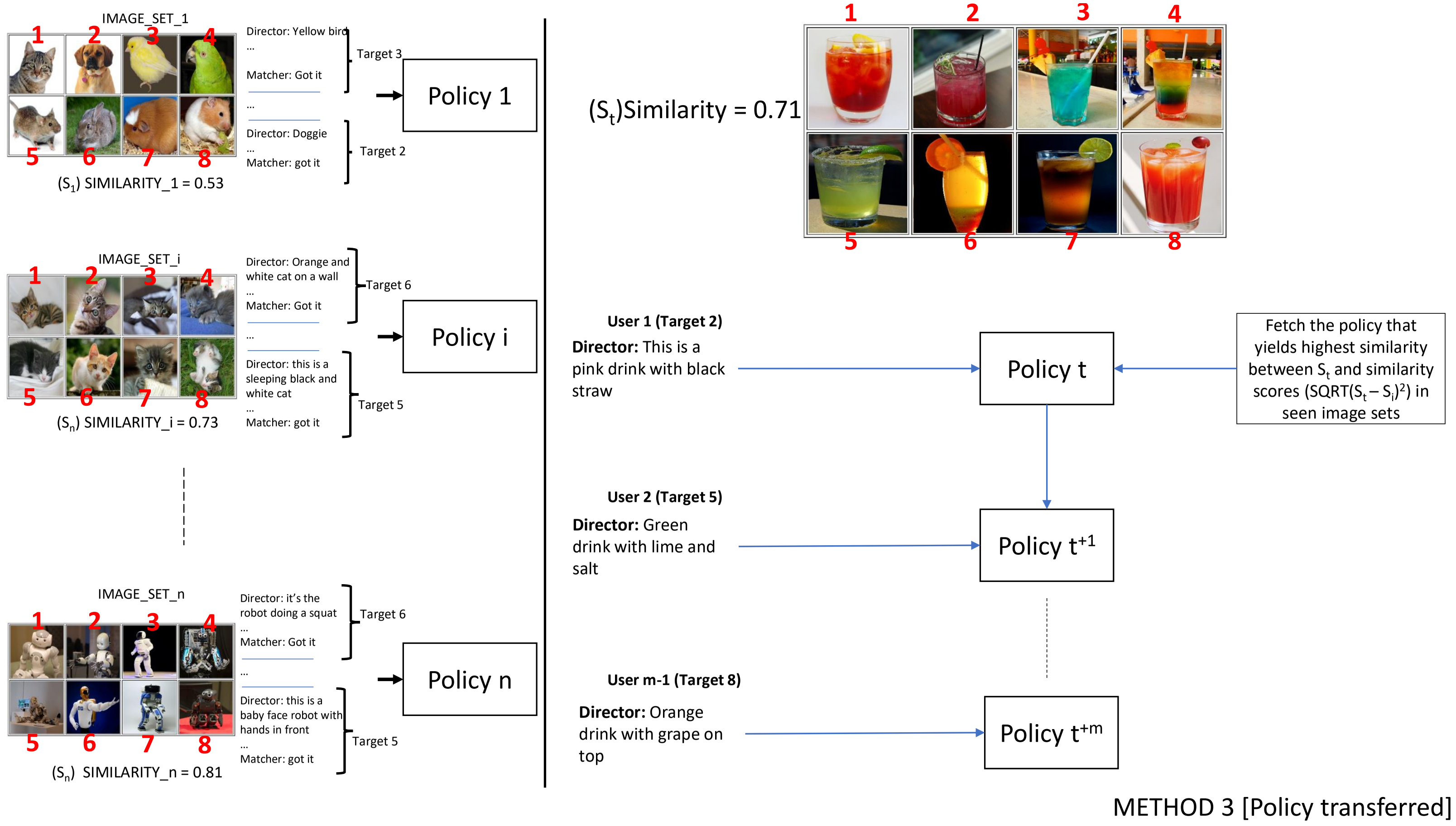

IMAGE_SET_1
1
2
3
4
5
6
7
8
(S$_1$) SIMILARITY_1 = 0.53
Director: Yellow bird
...
Matcher: Got it
...
Director: Doggie
...
Matcher: got it
Target 3
Target 2
Policy 1
IMAGE_SET_i
1
2
3
4
5
6
7
8
(S$_n$) SIMILARITY_i = 0.73
Director: Orange and white cat on a wall
...
Matcher: Got it
...
Director: this is a sleeping black and white cat
...
Matcher: got it
Target 6
Target 5
Policy i
IMAGE_SET_n
1
2
3
4
5
6
7
8
(S$_n$) SIMILARITY_n = 0.81
Director: it's the robot doing a squat
...
Matcher: Got it
...
Director: this is a baby face robot with hands in front
...
Matcher: got it
Target 6
Target 5
Policy n
(S$_t$)Similarity = 0.71
1
2
3
4
5
6
7
8
User 1 (Target 2)
Director: This is a pink drink with black straw
Policy t
Fetch the policy that yields highest similarity between S$_t$ and similarity scores (SQRT(S$_t$ – S$_i$)^2) in seen image sets
User 2 (Target 5)
Director: Green drink with lime and salt
Policy t^{+1}
User m-1 (Target 8)
Director: Orange drink with grape on top
Policy t^{+m}
METHOD 3 [Policy transferred]

Figure 5.19: The flow of information in Case 3.

classification, segmentation, vision understanding, and object identification. The images in these datasets are annotated with the ground-truth labels of objects and their relations. These object labels in the images can help with the task of generating visual reference resolution descriptions as the Director typically acts. These labels are utilized to build a simple simulated Director, which is then used to train a Matcher policy. The simulated Director in this case uses the ground-truth object labels and describes them in a sequence. Algorithm 2 shows the operation of the simulated Director.

Algorithm 2 Simulated Director algorithm

```
Input: Object annotations $A = \{A_0^0, A_1^0 ... A_n^8\}$, Target Image: $t$
Output: Target description $T_t$
$T_t = \{\}$
$A^t = \{A_0^t, A_1^t ... A_k^t\}$
$shuffle(A^t)$
$l = rand(1, |A^t|)$
for $A_i^t \in A^t$ do
    if $A_i^t$ not in $A - A^t$ AND $length(T_t) <= l$ then
        $T_t \leftarrow \{A_i^t\} \cup T_t$
    else
        CONTINUE
    end if
end for
if $length(T_t) < l$ then
    $T_t \leftarrow T_t \cup \{X \subset A - A^t : |X| = l\}$
end if
return $T_t$
```

Simulated Director. The simulated Director utilizes the visual image bounding box object annotations to generate descriptions. The bounding box annotations for the images do not exist for the images present in the RDG-Image dataset. These bounding box annotations were collected using crowd-sourcing for the ‘seen’ and ‘unseen’ images in the dataset. These annotation labels for the images present in the RDG-Image were obtained by posting the images on AMT and asking the users to perform the annotation and provide the descriptions. Figure 5.18 shows sample annotations for the images. The simulated Director in this work generates the description for the image using these annotations ('seen' object annotations). Algorithm 2 shows how this simulated Director operates. The simulated Director is provided with the object annotations of 8 images in the context set ($A = \{A_0^0, A_1^0 ... A_n^8\}$) and the target image t. The output of this Director is T_t which is a set of object annotations treated as the target image descriptions. The maximum length of any target image description is determined by the length (l), which is a random number generated within the range of 1 and the length of target image descriptions. l is the number of unique annotations that can be a part of T_t. l unique annotation labels for the target image (T_t) are generated from the target image annotations (A^t) by random selection. If l unique annotation labels are not present for the given image, the agent fetches annotation labels randomly from the remaining labels for the given target image and appends them. Each call to the simulated Director generates a new target description for the target image.

These simulated dialogues are used to train a policy using LSPI. LSPI uses the state representation, actions, and reward function as described in Section 5.3. The generated Q-values are now used to initialize the RL policy when training using the human Director description dialogues in the train portion of the 'unseen' dialogues, as in (Torrey & Shavlik (2010); Taylor & Stone (2009)).

Case 3 (TEVE)

In Case 3, we assume that prior conversations from a different set of images have occurred and a policy has been learned. In this case, we utilize the policy from a different image set ('seen' dialogues) to initialize the training process for the ‘unseen’ image set. The generated Q-values are now used to initialize the RL policy when training using the human Director description dialogues in the train portion of the ‘unseen’ dialogues, as shown in Figure 5.19. The challenge, in this case, is to choose the right Q-values to initialize the policy. In this case, the assumption is that the policy for the image sets that have similar difficulty will be similar and hence transferring the Q-values from an image set of similar difficulty will help train a policy faster. The challenge is, however, selecting the right target image set from which the policy can be transferred. We use image similarity as a proxy for difficulty. This is intuitive as the images that have high similarity tend to be the image sets with higher difficulty. To calculate image similarity, we feed the images through a convolution neural network (Szegedy et al. (2015)) trained using ImageNet (Deng et al. (2009)) and get the penultimate layer weights. These weights form the vector representation for each image in the context set. We find the average Euclidean distance between these image vectors, which will yield a single similarity score. We use this score to find the closest policy from the trained policies and use these Q-values for initialization. We then train the initialized policy using the dialogues from the human Directors.

5.7.2 Experiments

In this section, we will discuss our experiments using the three methods described above. The focus of this experiment is to measure the policy performance differences across conditions. It is, however, important to note that the policy requires input (P_t^*) from the NLU. In order to compare the different dialogue policies fairly, the NLU has to remain the same across the three conditions. Hence, the NLU is trained using the data from the training set (comprised of bounding box annotations) before starting the policy training. For training the policy, we use LSPI with value

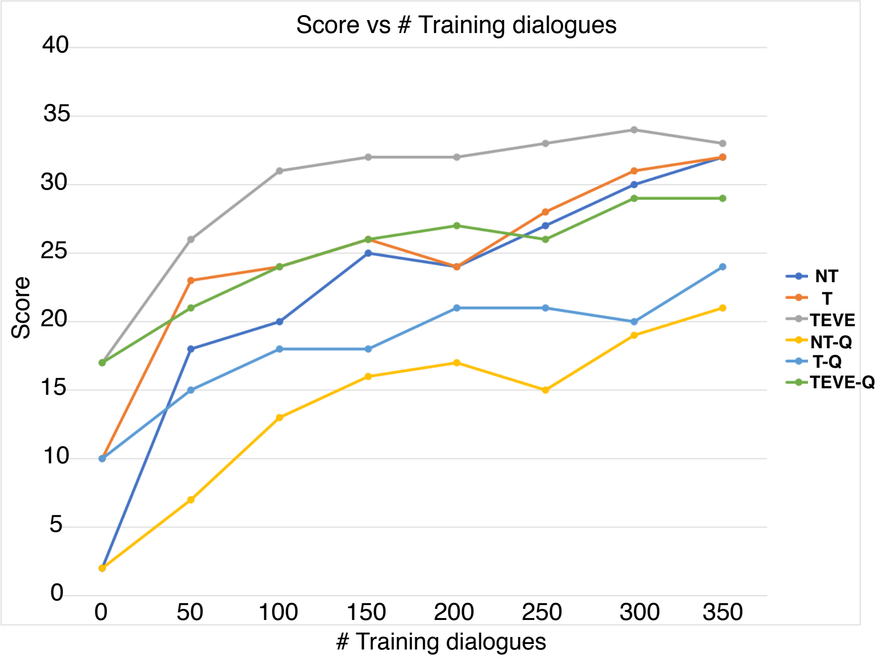


Figure 5.20: Scores for different transfer conditions.

function approximation using the same features as we used to train the RL policy described in Section 5.3. We test the policy decisions using the held out test set after training the policy in Cases 1, 2, and 3 for every 50 target descriptions. The policies for Cases 1, 2, and 3 are tested beginning at point 0. This initial point for Case 1 is the randomly initialized Q-values. For Case 2, it is the Q-values learned from simulated Director descriptions, and for Case 3, it is the policy from a different context set (different image set with Director descriptions from real dialogue data). We now begin training the dialogue policy by introducing the training conversations. We train all the policies with different initialization, train with the same amount of data, and analyze their differences in performance using the same test set. For training the policy we explore Vanilla Q-learning (Sutton & Barto (1998)) and LSPI (Lagoudakis & Parr (2003)). The state representation, actions, and rewards are the same as those in Section 5.3.

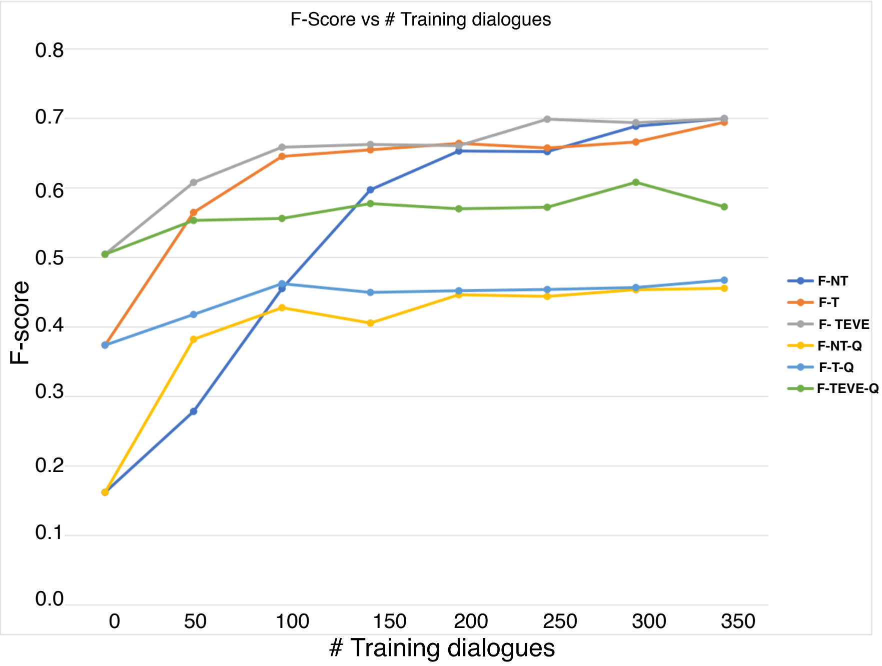


Figure 5.21: F-scores for different transfer conditions.

5.7.3 Results

In this section I will discuss the results obtained from the experiments. The results are reported across the 6 conditions mentioned below.

- NT-Q: This is Case 1 (no transfer) run with Vanilla Q-learning as the learning algorithm.
- NT: This is Case 1 (no transfer) run with LSPI as the learning algorithm.
- T-Q: This is Case 2 (transfer using the simulated Director) run with Vanilla Q-learning as the learning algorithm.
- T: This is Case 2 (transfer using the simulated Director) run with LSPI as the learning algorithm.

- TEVE-Q: This is Case 3 (transfer using the policy learned with dialogue data from another image set) run with Vanilla Q-learning as the learning algorithm.
- TEVE: This is Case 3 (transfer using the policy learned with dialogue data from another image set) run with LSPI as the learning algorithm.

Metrics. The **F-score** in this case is a measure of how good the agent is while taking the 'select' and 'wait' actions which are important for the agent's performance in gameplay. The agent is said to make the correct 'wait' decision if the selection from the NLU is incorrect. The agent's 'select' decision is considered right if the NLU output is the correct target image.

Humans do not make a 'skip' decision in the game. However, the decision to 'skip' is made by the agent to increase efficiency in gameplay. The agent in the case of 'skip' is neither penalized nor awarded points towards the calculation of the F-score. The **score** is calculated using the game logic when the agent makes the selection.

Figure 5.20 shows the difference in performance across the 6 conditions. We see that the agent learning the policy in the TEVE condition performs the best. The agent achieves better initial performance with 0 training data specific to the target image set. The agent also achieves superior performance upon training on the complete image set. The TEVE-Q condition achieves a better initial performance but soon loses its performance gains. This is because Vanilla Q-learning does not use any kind of function approximation which is required when the data is sparse as is our case. At the end of the 350 training dialogues, we observe that the conditions trained using LSPI are better than those trained using Vanilla Q-learning. We also observe that the T and NT conditions reach a performance almost similar to that of the TEVE condition upon training with all of the 350 dialogues.

Figure 5.21 shows the differences in F-scores measured across different conditions. We find a similar pattern, that the TEVE condition achieves better initial performance, and the performance gains taper upon seeing additional data. The gains in performance in the Vanilla Q-learning conditions are much lower than the gains that we see in the conditions trained using LSPI.

5.7.4 Discussions

From the results we can observe that: i) Using a policy pre-trained with real dialogue data (TEVE) achieves a much better performance. ii) When no real dialogue data is available for pre-training, using a simulated Director (T) achieves a better performance than a policy that is randomly initialized (NT). iii) LSPI with value function approximation yields higher scores than Vanilla Q-learning.

Using real user dialogues from a different image set (TEVE) performs better than the T and NT conditions. The policy decisions from a different image set are common across multiple image sets for a few state space features. For instance, with very low confidence values (e.g., ~ 0.0) and when the time consumed is low (e.g., < 2s) the decision to 'wait' is common across multiple image sets. Similarly, when the confidence value is very high (~ 1.0) the decision to say 'got it' is common across multiple image sets. Utilizing such common policy decisions across multiple image sets by initializing the Q-values yields better performance compared to the random initialization approaches that choose one of the actions randomly. Similarly, the decisions to 'skip' are also learned from other image sets (e.g., low confidence values at high time values). When a new image set is encountered, the RL can be optimized on a few of the state space values (confidence and time) without the need to relearn actions for all the state space values. This gain in performance can be achieved by utilizing the transfer approach.

The simulated conversations (T conditions) were designed to also help learn such patterns. However, the gain in performance is lower than in the TEVE condition. The visual object annotations are not sufficient to generate descriptions similar to humans. The visual object annotations are generated per image without the reference context set. The humans do not construct the target image (TI) descriptions in the same way as the visual object annotations (e.g., ‘it has a black straw’ [human TI description] vs. ‘black straw is in the drink’ [visual object annotation]). But the visual object annotations can yield descriptions that can still be used to initialize the Q-values. This results in a performance lower than the TEVE condition but better than the baseline NT condition.

5.8 Contributions

In this chapter, we utilized the RL framework to build an incremental dialogue policy for the RDG-Image game. We showed that in comparison with a strong human-level performing baseline, the RL approach performs better. The RL approach outperforms the strong baseline policy in an offline setting, and in a live human interaction study it achieves a better qualitative performance. This is one of the few studies in the literature that uses RL for incremental dialogue policy learning. Our work is also one of the rare studies in the literature where the RL policy is compared with a strong baseline, and not just a baseline that was specifically built for comparing with the RL policy. Another benefit of the RL approach is that it can be used in a transfer learning setting. We demonstrate that using a transfer learning approach, where the transferred Q-values are learned from a similar domain, helps learn a better dialogue policy with fewer data samples. This approach is beneficial when real conversational data is not available or limited. This is the first time in the literature that transfer learning is applied to incremental dialogue policy learning. Part of the work presented in this chapter has been published in Manuvinakurike et al. (2017).

In the next chapter, I will focus on the NLU module and discuss methods to extend the work with sophisticated semantic capabilities.

Chapter 6

Incremental Dialogue Act Recognition

"Action expresses priorities"

– Mahatma Gandhi, *Lawyer, Politician, Activist, Writer*

6.1 Introduction

One of the drawbacks of Eve is limited understanding and generation capabilities. During conversation Eve assumes that every utterance from the user is a target description. This limits Eve's capability to understand rich and diverse human utterances. The key to generating a diverse set of responses is to understand the diverse speech acts used by the users while conversing. For instance, to respond to a question with an answer, it is important to identify that a question has been asked. In this chapter, we will develop methods that enable SDS with such capabilities. We will motivate an annotation scheme for the tasks tackled in this work consisting of dialogue acts. We will then design a task for incremental recognition of these dialogue acts. The focus of this chapter will be to enhance the semantic processing capabilities in incremental SDS. The focus will be mainly on understanding as in the previous chapters.

6.2 Related Work

In this chapter, one of the factors we are concerned with is the alignment between dialogue acts (DAs) and individual words as they are spoken within Inter-Pausal Units (IPUs) (Koiso et al. (1998)) or *speech segments*[1]. A range of prior research has addressed several issues that arise in connection with the alignment of DAs and IPUs. Some researchers have focused on the problem of DA-internal pauses, developing classification approaches to decide whether consecutive segments of speech separated by a pause should be concatenated for interpretation (Komatani et al. (2015); Bell et al. (2001)). Other researchers have focused on determining, as quickly as possible, whether a pause in user speech represents an end-of-utterance (Ferrer et al. (2003); Raux & Eskenazi (2008); Atterer et al. (2008)). The detection of an end-of-utterance is closely related to system turn-taking decisions, and the optimization of a system's turn-taking policy during a user pause has also been explored, often with classification approaches (Sato et al. (2002); Takeuchi et al. (2004); Raux & Eskenazi (2008)). Beyond the work on this alignment problem, a related line of work has looked specifically at DA segmentation and classification given an input string of words together with an audio recording to enable prosodic and timing analysis (Petukhova & Bunt (2014); Zimmermann (2009); Zimmermann et al. (2006); Lendvai & Geertzen (2007); Ang et al. (2005); Nakano et al. (1999); Warnke et al. (1997)).

This work generally encompasses the problems of identifying DA-internal pauses as well as locating DA boundaries within speech segments. Prosody information has been shown to be helpful for accurate DA segmentation (Laskowski & Shriberg (2010); Shriberg et al. (2000); Warnke et al. (1997)) as well as for DA classification

[1] We use the two terms interchangeably in this chapter to refer to a period of continuous speech separated by pauses of a minimum duration before and after.

(Stolcke et al. (2000); Fernandez & Picard (2002)). In general, DA segmentation has been found to benefit from a range of additional features such as pause duration at word boundaries, the user's dialogue tempo (Komatani et al. (2015)), as well as lexical, syntactic, and semantic features. Work on system turn-taking decisions has used similar features to optimize a system's turn-taking policy during a user pause, often with classification approaches; e.g., (Sato et al. (2002); Takeuchi et al. (2004); Raux & Eskenazi (2008)). For DA segmentation and classification, researchers have explored both sequential models as well as joint models that optimize both decisions together. Recent joint models include Petukhova & Bunt (2014), Morbini & Sagae (2011), and Zimmermann (2009). In this work, we extract lexical and prosodic features from a stream of user speech which has either been human-transcribed or is being automatically transcribed by ASR. We then use a sequential model that automatically segments DAs with a CRF before classifying them with a SVM. The model runs incrementally after each new word. The two closest points of contact are Zimmermann (2009), which uses a CRF to jointly segment and classify DAs, and Petukhova & Bunt (2014), which uses a BayesNet or RIPPER to incrementally segment and classify DAs in 10 dimensions simultaneously with the DIT++ tagging scheme. In comparison with some related work, we are less focused on rapid identification of end-of-utterance or "end-of-DA". We rerun our segmentation and classification algorithm after every recognized word, and thus can recognize boundaries at the end of speech segments, but in this work we do not put a heightened emphasis on low-latency detection of such boundaries. To our knowledge, very little research has looked in detail at the impact of adding incremental DA segmentation to an implemented incremental system (though see Nakano et al. (1999)).

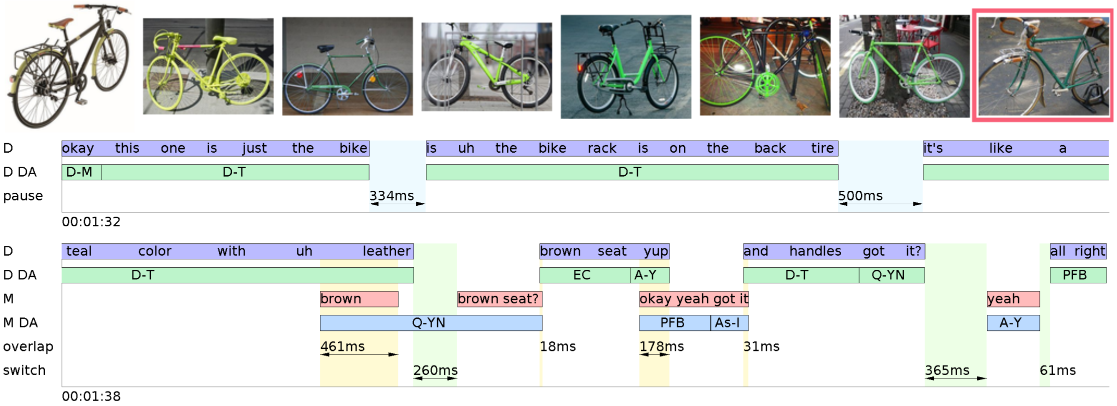


Figure 6.1: An example RDG-Image dialogue, where the Director (D) tries to identify the target image, highlighted in red, to the Matcher (M). The DAs of the Director (D DA) and Matcher (M DA) are indicated.

6.3 RDG-Image

6.3.1 Corpus & Annotations

For the previously collected data sets of human-human gameplay in RDG-Image both in a lab setting (Paetzel et al. (2014)) and in an online, web-based version of the game (Manuvinakurike & DeVault (2015); Paetzel et al. (2015)) we annotated the data. To support the experiments, a single annotator segmented and annotated the main game rounds from our lab-based RDG-Image corpus with a set of DA tags.[2] The corpus includes gameplay between 64 participants (32 pairs, age: $M = 35$, $SD = 12$, gender: 55% female). 11% of all participants reported they frequently played similar games before; the other 89% had no or very rare experience with similar games. All speech was previously recorded, manually segmented into speech segments (IPUs) at pauses of 300ms or greater, and manually transcribed. The new DA segmentation and annotation steps were carried out at the same time by adding

[2] We excluded from annotation the training rounds in the corpus, where players practiced playing the game.

Example	# IPUs	# DAs	Annotation
1	1	5	PFB that's okay ‖ D-T um this castle has a ‖ ST oh gosh this is hard ‖ D-T this castle is tan ‖ D-T it's at a diagonal with a blue sky
2	1	2	D-T and it's got lemon in it ‖ Q-YN you got it
3	1	2	PFB okay ‖ D-T this is the christmas tree in front of a fireplace
4	1	2	EC fireplace ‖ As-I got it
5	2	2	D-M all right ‖ D-T this is ... this is this is the brown circle and it's not hollow
6	3	1	D-T this is a um ... tan or light brown ... box that is clear in the middle
7	3	2	D-M all right ‖ D-T he's got he's got that ... that ... first uh the first finger and the thumb pointing up
8	3	2	ST um golly ‖ DT this looks like a a a ... ginseng ... uh of some sort
9	2	4	ST oh wow ‖ D-M okay ‖ D-T this one ... looks it has gray ‖ D-T a lotta gray on this robot

Table 6.1: Examples of annotated DA types, DA boundaries (‖), and IPU boundaries (...). The number of IPUs and DAs in each example are indicated.

boundaries and DA labels to the transcribed speech segments from the game. The annotator used both audio and video recordings to assist with the annotation task. The annotations were performed on transcripts which were seen as segmented into IPUs.

Table 6.1 provides several examples of this annotation. We designed the set of DA labels to include a range of communicative functions we observed in human-human gameplay, and to encode distinctions we expected to prove useful in an automated agent for RDG-Image. Our DA label set includes Positive Feedback (PFB), Describe Target (D-T), Self-Talk (ST), Yes-No Question (Q-YN), Echo Confirmation (EC), Assert Identified (As-I), and Assert Skip (As-S). We also include a filled-pause DA (P) used for 'uh' or 'um' separated from other speech by a pause. The complete list of 18 DA labels and their distribution are included in Tables 6.4 and 6.5. To assess the reliability of annotation, two annotators annotated one game (2 players, 372 speech segments); we measured kappa for the presence of boundary markers (‖) at 0.92 and word-level kappa for DA labels at 0.83.

Describe Target (D-T) is concerned with the description of the target image. (D-T) covers 61.7% of the total utterances by the Director. **Echo Confirmation (EC)** corresponds to the confirmation of the partner's query by repeating the same phrase. **Positive Feedback (PFB)** is the partner providing positive feedback to the user's queries typically. **Action Directive (A-D)** corresponds to the players asking their partner to perform certain game actions such as clicking the image. **Q-YN** is a simple yes-no question which demands a yes-no answer from the partner. **AY and AN** are the yes and no answers to **Q-YN**. **Clarification Question (Q-C)** is a question seeking clarification about the description of the image. **Discourse Markers (D-M)** are DAs which indicate to the partner that the player is about to say something. **(Q-Wh)** are wh-questions posed by either of the players. **Q-D** are disjunctive questions. **Filled pauses (P)** and **hedges (H)** are annotated as well. The **Assert Identified (As-I)** dialogue act corresponds to the Matcher action typically signalling that the player has identified the image that was described by their partner. **Matcher Assertions (As-M)** are Matcher assertions made for courtesy purposes. **Assert Skip (As-S)** are the assertions made by the Matcher asking the Director to move on to the next target image. **ST** corresponds to self talk statements which are usually the murmurs or statements that did not receive response from the partner. Transcribed utterances that correspond to conversations outside the gameplay are tagged **G**.

Summary statistics for the annotated corpus are as follows. The corpus contains 64 participants (32 pairs), 1,906 target images, 8,792 speech segments, 67,125 word tokens, 12,241 DA segments, and 4.27 hours of audio. The mean number of DAs per speech segment is 1.39. In Table 6.3, we summarize the distribution in number of DAs initiated per speech segment. 23% of speech segments contain the beginning of at least two DAs; this highlights the importance of being able to find the boundaries

Total participants	64 (32 pairs)
Total target images	1,906
Total speech segments	8,792
Total word tokens	67,125
Total DA segments	12,241
Total duration of audio	4.27 hours

Table 6.2: Summary statistics for the DA-annotated RDG-Image corpus.

Number of DAs	0	1	2	≥ 3
% of speech segments	3	74	18	5

Table 6.3: The distribution in the number of DAs whose first word is within a speech segment.

between multiple DAs inside a speech segment. Most DAs begin at the start of a speech segment (i.e.. immediately after a pause), but 29% of DAs begin at the second word or later in a speech segment. 4% of DAs contain an internal pause and thus span multiple speech segments.

6.4 Segmentation and Dialogue Act Recognition

In this section, we present a case study of implementing an incremental DA segmentation capability for the RDG-Image game illustrated in Figure 6.1. As we discussed in previous chapters, in this game two players converse freely in order to identify a specific target image on the screen (outlined in red). When played by human players, as in Figure 6.1, the game creates a variety of fast-paced interaction patterns, such as question-answer exchanges. Our motivation is to eventually enable a future version of our automated RDG-Image agent (Paetzel et al. (2015)) to participate in the most common interaction patterns in human-human gameplay.

For example, in Figure 6.1, two fast-paced question-answer exchanges arise as the director D is describing the target image. In the first, the Matcher M asks *brown...brown seat?* and receives an almost immediate answer *brown seat yup.* A moment later, the Director continues the description with *and handles got it?*, both adding *and handles* and also asking *got it?* without an intervening pause. We believe that an important step toward automating such fast-paced exchanges is to create an ability for an automated agent to incrementally recognize the various DAs, such as yes-no questions (Q-YN), target descriptions (D-T), and yes answers (A-Y) in real-time as they are happening.

In this chapter, first, we define a sequential approach to incremental DA segmentation and classification that is straightforward to implement and which achieves a useful level of performance when trained on a small annotated corpus of domain-specific DAs. Second, we explore the performance of our approach using both existing and new performance metrics for DA segmentation. Our new metrics emphasize the importance of precision and recall of specific DA types, independently of DA boundaries. These metrics are useful for evaluating DA segmenters that operate on noisy ASR output and which are intended for use in systems whose dialogue policies are defined in terms of the presence or absence of specific DA types, independently of their position in user speech. This is a broad class of systems. Third, while much of the prior work on DA segmentation has been corpus-based, we report here on an initial integration of our incremental DA segmenter into an implemented, high-performance agent for the RDG-Image game. Our case study suggests that incremental DA segmentation can be performed with sufficient accuracy for us to begin to extend our baseline agent's conversational abilities without significantly degrading its current performance in the game.

DA	**Description**	**Example**
D-T	Describe target	this is the christmas tree in front of a fireplace
As-I	Assert Identified	got it
NG	Non-game utterances	okay there i saw the light go on
PFB	Positive feedback	okay
ST	Self-talk statements	ooh this is gonna be tricky
P	Filled pause	uh
D-M	Discourse marker	alright
Q-YN	Yes-No question	is it on something white
A-Y	Yes answer	yeah
EC	Echo confirmation	the blue
As-M	Matcher assertions	it didn't let me do it
Q-C	Clarification question	bright orange eyes?
A-D	Action directive	oh oh wait hold on
A-N	No answer	no, nah
H	Hedge	i don't know what it is
Q-D	Disjunctive question	are we talking dark brown or like caramel brown
Q-Wh	Wh-question	what color's the kitty
As-S	Assert skip	i'm gonna pass on that

Table 6.4: The complete list of DAs in the annotated RDG-Image corpus.

DA	All	Dir	Mat	DA	All	Dir	Mat
D-T	41	60	0	EC	2	.5	6
As-I	15	0	46	As-M	2	0	4
NG	11	9	11	Q-C	2	.5	4
PFB	8	10	7	A-D	1	.3	2
ST	4	4	4	A-N	.5	.7	.2
P	4	6	2	H	.5	.7	0
D-M	3	5	.2	Q-Wh	.3	0	.5
Q-YN	3	.6	7	As-S	.1	0	.1
A-Y	2	3	1	Q-D	.4	0	1.2

Table 6.5: DA distribution. We report the relative percentages for each DA out of all DAs, Director DAs, and Matcher DAs, respectively.

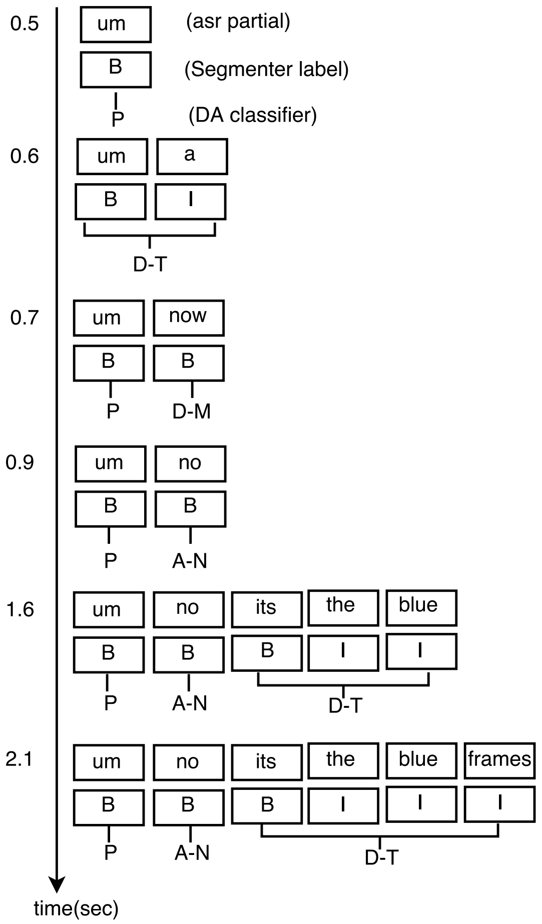


Figure 6.2: The operation of the pipeline on selected ASR partials (with time index in seconds).

6.5 Technical Approach

The goal for our incremental DA segmentation component is to segment the recognized speech for a speaker into individual DA segments and to assign these segments to the 18 DA classes in Table 6.4. We aim to do this in an incremental (word-by-word) manner, so that information about the DAs within a speech segment becomes available before the user stops or pauses their speech.

Figure 6.2 shows the incremental operation of our sequential pipeline for DA segmentation and classification. We use Kaldi (Povey et al. (2011)) for ASR, and we adapt the work of (Plátek & Jurčíček (2014)) for incremental ASR. The pipeline is invoked after each new partial ASR result becomes available (i.e., every 100ms), at which point all the recognized speech is resegmented and reclassified in a *restart incremental* (Schlangen & Skantze (2011)) design. The input to the pipeline includes all the recognized speech from one speaker (including multiple IPUs) for one target image sub-dialogue.

In our sequential pipeline, the first step is to use sequential tagging with a CRF (Conditional Random Field) (Lafferty et al. (2001)) implemented in Mallet (McCallum (2002)) to perform the segmentation. The segmenter tags each word as either the beginning (B) of a new DA segment or as a continuation of the current DA segment (I).[3] Then, each resulting DA segment is classified into one of 18 DA labels using an SVM (Support Vector Machine) classifier implemented in Weka (Hall et al. (2009)).[4]

6.5.1 Features

Prosodic features. We use word-level prosodic features similar in nature to Litman et al. (2009). The alignment between words and computed prosodic features is achieved using a forced aligner (Baumann & Schlangen (2012)) to generate word-level timing information. For each word, we first obtain pitch and RMS values every 10ms using InproTK (Baumann & Schlangen (2012)). Because pitch and energy features can be highly variable across users, our pitch and energy features are represented as z-scores that are normalized for the current user up to the current

[3] Note that our annotation scheme completely partitions our data, with every word belonging to a segment and receiving a DA label. We have therefore elected not to adopt BIO (Begin-Inside-Outside) tagging.

[4] This DA classification approach is similar to the incremental classification model of DeVault et al. (2011b), but here using simple DA labels as output instead of complete semantic frames.

word. For the pitch and RMS values, we obtain the max, min, mean, variance and the co-efficients of a second degree polynomial. Pause durations at word boundaries provide an additional useful feature (Kolář et al. (2006); Zimmermann (2009)). All numeric features are discretized into bins. We currently use prosody for segmentation but not classification.[5]

To avoid overfitting on the prosodic features we calculate the z value for each of the prosodic (pitch and power) value. In an incremental scenario the estimated values of the mean (μ) and standard deviation (σ) for the current user are re-calculated after every word. The z-values are used as prosodic features instead of raw features.

$$Z - value = \frac{x - \mu}{\sigma}$$

Lexico-syntactic & contextual features. We use word unigrams along with the corresponding part-of-speech (POS) tags, obtained using Stanford CORENLP (Manning et al. (2014)), as a feature for both the segmentation and the DA classifier. Words with a low frequency (<10) are substituted with a low frequency word symbol. The top level constituent category from a syntactic parse of the DA segment is also used.

Several contextual features are included. The role of the speaker (Director or Matcher) is included as a feature. Previously recognized DA labels from each speaker are included. Another feature is added to assist with the Echo Confirmation (EC) DA, which applies when a speaker repeats verbatim a phrase recently spoken by the other interlocutor. For this we use features to mark word-level unigrams that appeared in recent speech from the other interlocutor. Finally, a categorical

[5] For the experiments reported in this chapter, prosodic features were calculated offline, but they could in principle be calculated in real time.

feature indicates which of 18 possible image sets (e.g., bikes as in Figure 6.1) is under discussion; simpler images tend to have shorter segments.[6]

6.5.2 Discussion of Machine Learning Setup

A salient alternative to our sequential pipeline approach – also adopted for example by Ang et al. (2005) – is to use a joint classification model to solve the segmentation and classification problems simultaneously, potentially thereby improving performance on both problems (Petukhova & Bunt (2014); Morbini & Sagae (2011); Zimmermann (2009); Warnke et al. (1997)). We performed an initial test using a joint model and found, unlike the finding reported by Zimmermann (2009), that for our corpus a joint approach performed markedly worse than our sequential pipeline.[7] We speculate that this is due to the relative sparsity of data on rarer DA types in our relatively small corpus. For similar reasons, we have not yet tried to use RNN-based approaches such as LSTMs, which tend to require large amounts of training data. We will adopt deep learning methods in later sections on another dataset in a similar setting.

6.6 Experiment and Results

We report on two experiments. In the first experiment, we train our DA segmentation pipeline using the annotated corpus of Section 6.3.1 and report results on the observed DA segment boundaries (Section 6.6.1) and DA class labels (Section 6.6.2). In the second experiment, presented in Section 6.6.3, we report on

[6] The image set feature affects the performace of the segmenter only slightly.

[7] We used a joint CRF model similar to the BI coding of Zimmermann (2009).

Condition	Transcripts (T)	Segment Boundaries (S)	DA labels (D)
HT-HS-HD	Human	Human	Human
HT-HS-AD	Human	Human	Automated
HT-AS-AD	Human	Automated	Automated
AT-AS-AD	ASR	Automated	Automated

Table 6.6: Conditions for evaluating DA segmentation and classification.

Condition	# IPUs	Example	
HT-HS-HD	1	(a)	A-N um no ‖ D-T it’s the blue frame ‖ D-T but it’s an orange seat and an orange handle
HT-HS-AD	1	(b)	A-N um no ‖ D-T it’s the blue frame ‖ D-T but it’s an orange seat and an orange handle
HT-AS-AD	1	(c)	P um ‖ A-N no ‖ D-T it’s the blue frame ‖ D-T but it’s an orange seat ‖ D-T and an orange handle
AT-AS-AD	1	(d)	A-N on no ‖ D-T it’s the blue frame ‖ D-T but it’s an orange seat ‖ D-T and orange ‖ A-N no

Table 6.7: Examples of DA boundaries (‖) and DA labels in each condition.

a policy simulation that investigates the effect of our incremental DA segmentation pipeline on a baseline automated agent’s performance.

For the first experiment, we use a hold-one-pair-out cross-validation setup where, for each fold, the dialogue between one pair of players is held out for testing, while automated models are trained on the other pairs.

To evaluate our pipeline, we use four data conditions, summarized in Table 6.6, that represent increasing amounts of automation in the pipeline. These conditions allow us to better understand the sources for observed errors in segment boundaries and/or DA labels. Our notation for these conditions is a compact encoding of the data sources used to create the transcripts of user speech, the segment boundaries, and the DA labels. Our reference annotation, described in Section 6.3.1, is notated HT-HS-HD (human transcript, human segment boundaries, human DA labels). Example segmentations for each condition are in Table 6.7.

Condition	Features	**Accuracy**	**F-Score**		**DSER**
			B tag	I tag	
1-DA-per-IPU		0.78	0.23	0.87	0.26
HT-AS-AD	Prosody (I)	0.72	0.62	0.69	0.42
HT-AS-AD	Lexico-Syntactic & Contextual (II)	0.90	0.82	0.82	0.31
HT-AS-AD	I+II	0.91	0.83	0.84	0.30
Human annotator		0.95	0.91	0.94	0.15

Table 6.8: Observed DA segmentation performance. These results consider only DA boundaries.

6.6.1 Evaluation of DA Segment Boundaries

In this evaluation, we ignore DA labels and look only at the identification of DA boundaries (notated by || in Table 6.7, and encoded using B and I tags in our segmenter). For this evaluation, we use human transcripts and compare the boundaries in our reference annotations (HT-HS-HD) to the boundaries inferred by our automated pipeline (HT-AS-AD).[8]

In Table 6.8, we present results for versions of our pipeline that use three different feature sets: only prosody features (I), only lexico-syntactic and contextual features (II), and both (I+II). We include also a simple 1-DA-per-IPU baseline that assumes each IPU is a single complete DA; it assigns the first word in each IPU a B tag and subsequent words an I tag. Finally, we also include numbers based on an independent human annotator using the subset of our annotated corpus that was annotated by two human annotators. For this subset, we use our main annotator as the reference standard and evaluate the other annotator as if their annotation were a system's hypothesis.[9]

[8] We evaluate our DA segmentation performance using human transcripts, rather than ASR, as this allows a simple direct comparison of inferred DA boundaries.

[9] For comparison, the chance-corrected kappa value for word-level boundaries is 0.92; see Section 6.3.1.

Condition	Metrics used for human transcripts			Alignment-based metrics	
	DER	Strict	Lenient	Levenshtein-Lenient	CER
HT-HS-AD	0.39	0.09	0.09	0.07	0.27
HT-AS-AD	0.72	0.38	0.15	0.12	0.39
AT-AS-AD				0.39	0.52

Table 6.9: Observed DA classification and joint segmentation+classification performance.

The reported numbers include word-level accuracy of the B and I tags, F-score for each of the B and I tags, and the DA segmentation error rate (DSER) metric of Zimmermann et al. (2006). DSER measures the fraction of reference DAs whose left and right boundaries are not exactly replicated in the hypothesis. For example, in Table 6.7, the reference (a) contains three DAs, but only the boundaries of the second DA (*it's the blue frame*) are exactly replicated in hypothesis (c). This yields a DSER of 2/3 for this example.

We find that our automated pipeline (HT-AS-AD) with all features performs the best among the pipeline methods, with word-level accuracy of 0.91 and DSER of 0.30. Its performance however is worse than an independent human annotator, with double the DSER. This suggests there remains room for improvement at boundary identification. The 1-DA-per-IPU baseline does well on the common case of single-IPU DAs, but it fails ever to segment an IPU into multiple DAs. We use the pipeline with all features in the following sections.

6.6.2 Evaluation of DA Class Labels

The accuracies are calculated based on the correct dialogue act assigned to each of the token.

$$Acc = \frac{\text{num of tokens with correct class labels}}{\text{tokens}}$$

In this evaluation, we consider DA labels assigned to recognized DA segments using several types of metrics. We summarize our results in Table 6.9.

Metrics used for human transcripts. We first compare our reference annotations (HT-HS-HD) to the performance of our automated pipeline *when provided with human transcripts as input.* For this comparison, we use three error rate metrics (Lenient, Strict, and DER) from the DA segmentation literature that are intuitively applied when the token sequence being segmented and labeled is identical (or at least isomorphic) to the annotated token sequence. Lower is better for these. The Lenient and Strict metrics (Ang et al. (2005)) are based on the DA labels assigned to each individual word (by way of the label of the DA segment that contains that word). Lenient is a per-token DA label error rate that ignores DA segment boundaries.[10] In Table 6.9, this error rate is 0.09 when human-annotated boundaries are fed into our DA classifier (HT-HS-AD) and 0.15 when automatically-identified boundaries are used (HT-AS-AD). This shows that incorrect automatic segmentation of human transcripts causes about 6% of words (0.15-0.09) to receive the wrong DA label.

Strict and DER are boundary-sensitive metrics. Strict is a per-token error rate that requires each token to receive the correct DA label and also to be part of a DA segment whose exact boundaries appear in the reference annotation. This is a much higher standard.[11] Our pipeline (HT-AS-AD) achieves a Strict error rate of 0.38. Dialogue Act Error Rate (DER) (Zimmermann et al. (2006)) is the fraction of reference DAs whose left and right boundaries and label are perfectly replicated in the hypothesis. While the reported boundary-sensitive error rate numbers (0.39

[10] E.g., in Table 6.7 (c), the only Lenient error is at word *um*.

[11] E.g., in Table 6.7 (c), only the four words *it's the blue frame* would count as non-errors on the Strict standard.

and 0.72) may appear to be high, many of these boundary errors may be relatively innocuous from a system standpoint.

Alignment-based metrics. We also report two additional metrics that are intuitively applied even when the word sequence being segmented and classified is only a noisy approximation to the word sequence that was annotated, i.e., under an ASR condition such as AT-AS-AD. The Concept Error Rate (CER) is a word error rate (WER) calculation (Chotimongkol & Rudnicky (2001)) based on a minimum edit distance alignment of the DA tags (using one DA tag per DA segment). Our fully automated pipeline (AT-AS-AD) has a CER of 0.52.

We also report an analogous word-level metric which we call 'Levenshtein-Lenient'. To our knowledge this metric has not previously been used in the literature. It replaces each word in the reference and hypothesis with the DA tag that applies to it, and then computes a WER on the DA tag sequence. It is thus a Lenient-like metric that can be applied to DA segmentation based on ASR results. Our automated pipeline (AT-AS-AD) scores 0.39, indicating considerable differences in the hypothesized sequences of word-level DA tags.

DA multiset precision and recall metrics. When ASR is used, the CER and Levenshtein-Lenient metrics give an indication of how well you are doing at replicating the ordered sequence of DA tags. But in building a system, sometimes the sequence is less of a concern, and what is desired is a breakdown in terms of precision and recall per DA tag. One way to achieve this is with a metric like the "F measure" of Zimmermann (2009), which looks at the DAs that receive both the correct boundaries and the correct label, and computes precision and recall. However, for system building, getting the exact boundaries right may be less of a concern, and may be too difficult a standard to meet when noisy automated ASR transcription is involved. Many dialogue systems use policies that are triggered when a certain DA type has occurred in the user's speech (such as an agent that processes yes (A-Y) or no (A-N) answers differently, or a Director agent for the RDG-Image game that moves on when the Matcher performs As-I ("got it")). For such systems, exact DA boundaries and even the order of DAs is not of paramount importance so long as a correct DA label is produced around the time the user performs the DA.

For example, to an agent like Eve, what is likely to matter most for certain DA types is that there is high precision and recall for the occurrence of the DA. For example, if the agent were in the Director role, it would need to be able to reliably recognize the occurrence of As-I ("got it") so that the dialogue or interaction manager could then move onto the next object. The exact DA boundaries and even the order of DA labels is likely not of paramount importance so long as the As-I label is produced around the time the user performs this DA.

We therefore define a more permissive measure that looks only at precision and recall of DA labels within a sample of user speech. As an example, in (a) in Table 6.7, there is one A-N label and two D-T labels. In (d), there are two A-N labels and 3 D-T labels. Ignoring boundaries, we can represent as a multiset the collection of DA labels in a reference A or hypothesis H, and compute standard multiset versions of precision and recall for each DA type.

We report these numbers for our most common DA types in Table 6.10. Here, we continue to use the speech of one speaker during a target image subdialogue as the unit of analysis. The data show that precision and recall generally decline for all DA types as automation increases in the conditions from left to right. We do relatively well with the most frequent DA types, which are D-T and As-I. A particular challenge, even in human transcript+segment condition HT-HS-AD, is the DA tag PFB. In a manual analysis of common error types, we found that the

Condition	HT-HS-AD		HT-AS-AD		AT-AS-AD	
	P	**R**	**P**	**R**	**P**	**R**
D-T	0.98	0.98	0.85	0.95	0.79	0.88
As-I	0.97	0.97	0.74	0.96	0.73	0.68
NG	0.84	0.89	0.72	0.88	0.63	0.50
PFB	0.67	0.65	0.50	0.77	0.42	0.60
ST	0.92	0.92	0.71	0.63	0.41	0.31
Q-YN	0.94	0.85	0.86	0.85	0.55	0.52
AN	0.90	0.90	0.70	0.67	0.42	0.32
A-Y	0.79	0.79	0.65	0.75	0.59	0.58

Table 6.10: DA multiset precision and recall metrics for a sample of higher-frequency DA tags.

different DA labels used for very short utterances like 'okay' (D-M, PFB, As-I) and 'yeah' (A-Y, PFB, As-I) are often confused. We believe this type of error could be reduced through a combination of improved features, collapsed DA categories, and more detailed annotation guidelines. ASR errors also often cause DA errors; see e.g., Table 6.7 (d). The version of Kaldi ASR used in these experiments did not incorporate the latest DNN acoustic models which have lowered word error rates substantially; our automated pipeline's results will likely improve by improving the underlying ASR.

6.6.3 Evaluation of Simulated Agent Dialogues

Motivation. In this experiment, we perform an offline investigation into the potential impact on our agent's image-matching performance if we integrate the incremental DA segmentation pipeline. We take the "fully-incremental" version of Eve from (Paetzel et al. (2015)) as our baseline agent in this experiment, i.e., the CDR version of Eve.

The baseline agent's design focuses on the most common DA types in our RDG-Image corpora: D-T for the Director (constituting 60% of Director DAs), and As-I for the Matcher (constituting 46% of Matcher DAs). Effectively, the baseline agent assumes every word the user says is describing the target, and uses an optimized policy to decide the right moment to commit to a selection (As-I) or ask the user to skip the image (As-S). Eve's typical interaction pattern is illustrated in Figure 6.3.

This experiment is narrowly focused on the impact of using the pipeline to segment out only the D-T DAs and to use only the words from detected D-Ts in the target image classifier and the agent's policy decisions. Changing the agent pipeline from using the Director's full utterance towards only taking the D-T tagged words into account could potentially have a negative impact on the baseline agent's performance. For example, for the fully automated condition AT-AS-AD in Table 6.10, D-T has precision 0.79 and recall 0.88. The 0.88 recall suggests that some D-T words will be lost (in false negative D-Ts) by integrating the new DA segmenter. Additionally, as shown in Figure 6.2, the recognized words and whether they are tagged as D-T can change dynamically as new incremental ASR results arrive, and this instability could undermine some of the advantage of segmentation. On the other hand, by excluding non-D-T text from consideration, there is a potential to decrease noise in the agent's understanding and improve the agent's accuracy or speed.

While this design has enabled the agent to play the Matcher role with some success (achieving approximately human-level game scores in terms of points/sec (Paetzel et al. (2015))), it excludes a substantial range of natural conversational patterns in the game. Indeed, taken together, D-T and As-I only represent 56% of the overall DAs in our annotated corpus. (See Table 6.4 for the full distribution.)

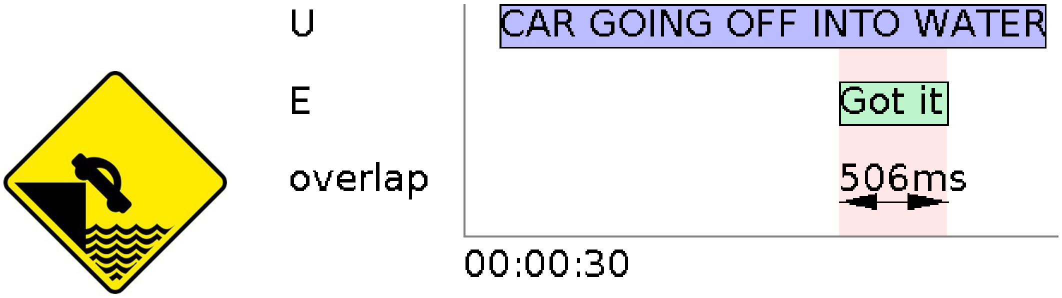


Figure 6.3: Eve (E) identifies a target image.

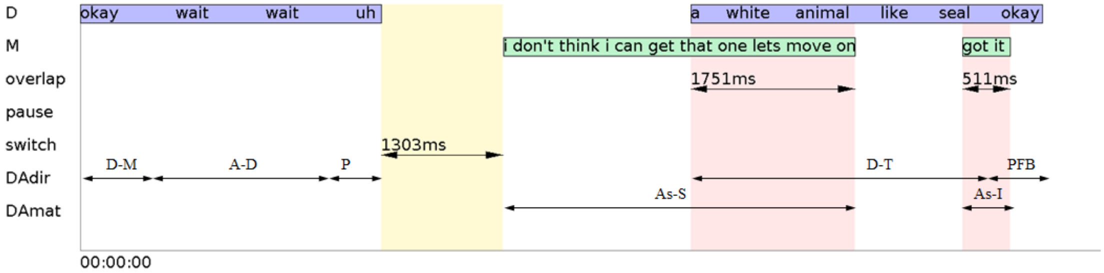


Figure 6.4: Our baseline Eve fails to recognize that the user is requesting more time. Instead, Eve judges that she is failing to understand a description of the target image, and suggests they skip it.

In the post-game survey, users rated baseline agent Eve as significantly less efficient in its gameplay than human partners, and also felt significantly less able to talk to our baseline agent "in the way I normally talk to another person".

An example where Eve fails to respond like most human partners would is given in Figure 6.4. We believe that by developing an incremental DA segmentation component for the agent, and by enabling the agent to detect and respond to a wider variety of DA types, we will eventually be able to increase its repertoire of conversational interaction patterns and thereby improve user experience. In Figure 6.1, the Director's answer *no* (A-N) and continued description *it's green...* appear to be understood incrementally before the Matcher initiates *yes I got it.*

While we are motivated to enable such interactive capabilities, at the same time, it is important that we not reduce the performance of the agent by introducing more complex and error-prone language processing modules. A further consideration is that incremental operation is critical to the baseline agent's current performance and favorable user impressions. As the ASR partials can be unstable, the segmenter's and the DA classifier's outputs are unstable as well. Figure 6.2 shows an example of the evolving output from the pipeline. If the D-T vs. non-D-T determination is highly unstable, it could undermine some of the advantages of the segmentation capability.

The goals of including segmentation and labelling in the dialogue pipeline is to improve the overall performance and acceptance of the agent by distinguishing the DAs for the game partner and ultimately introducing a greater variety of responses. However, this will only improve the overall experience with the agent if it does not decrease the NLU accuracy and the interaction speed as both imply reduction in the scores of the agent. The pipeline operating incrementally is critical for including the system into a real time SDS.

Experiment. As an initial investigation into the issues described above, we adopt the "eavesdropper" framework for policy simulation. In an eavesdropper simulation, the Director's speech from pre-recorded target image dialogues is provided to the agent, and the agent simulates alternative policy decisions as if it were in the Matcher role. We have found that higher cross-validation performance in these offline simulations has translated to higher performance in live interactive human-agent studies (Paetzel et al. (2015)).

We created a modified version of our agent that uses the fully automated pipeline (AT-AS-AD) to pass only word sequences tagged as D-T to the agent's language understanding component (a target image classifier), effectively ignoring other DA types. Tagging is performed every 100 ms on each new incremental output segment published by the ASR. We then compare the performance of our baseline and modified agent in a cross-validation setup, using an eavesdropper simulation to train and test the agents. We use a corpus of human-human gameplay that includes 18 image sets and game data from both the lab-based corpus of 32 games described in Section 6.3.1 and also the web-based corpus of an additional 98 human-human RDG-Image games described in Manuvinakurike & DeVault (2015). Each simulation yields a new trained NLU (target image classifier, based either on all text or only on D-T text) and a new optimized policy for when the agent should perform As-I vs. As-S. Within the simulations, for each target image, we compute whether the agent would score a point and how long it would spend on each image.

	image set	total time(sec)	total points p	p/sec	NLU accuracy	avg sec/image
All DAs	Pets	984.7	182	0.18	0.77	4.15
	Zoo	921.1	203	0.22	0.79	3.60
	Cocktails	1300.3	153	0.12	0.60	5.12
	Bikes	1630.9	126	0.08	0.47	6.12
Only D-T	Pets	992.0	184	0.19	0.78	4.19
	Zoo	932.8	198	0.21	0.77	3.64
	Cocktails	1326.7	155	0.12	0.61	5.22
	Bikes	1678.4	130	0.08	0.49	6.29

Table 6.11: Overall performance of the eavesdropper simulation on the unsegmented data (All DAs) and the automatically segmented data (Only D-T) identified with our pipeline (AT-AS-AD).

Table 6.11 summarizes the observed performance in these simulations for four sample image sets in the two agent conditions. All results are calculated based on leave-one-user-out training and a policy optimized on points per second. A Wilcoxon-Mann-Whitney Test on all 18 image sets indicated that, between the two conditions, there is no significant difference in the total time ($Z = -0.24$, $p = .822$), total points scored ($Z = -0.06$, $p = .956$), points per second ($Z = -0.06$, $p = .956$), average seconds per image ($Z = -0.36$, $p = .725$), or NLU accuracy ($Z = -0.13$, $p = .907$).

These encouraging results suggest that our incremental DA segmenter achieves a performance level that is sufficient for it to be integrated into our agent, enabling the incremental segmentation of other DA types without significantly compromising (or improving) the agent's current performance level. These results provide a complementary perspective on the various DA classification metrics reported in Section 6.6.2.

The current baseline agent (Paetzel et al. (2015)) can only generate As-I and As-S dialogue acts. In future work, the fully automated pipeline presented here will enable us to expand the agent's dialogue policies to support additional patterns of interaction beyond its current skillset. For example, the agent would be better able to understand and react to a multi-DA user utterance like *and handles got it?* in Figure 6.1. By segmenting out and understanding the Q-YN *got it?*, the agent would be able to detect the question and answer with an A-Y like *yeah*. Overall, we believe the ability to understand the natural range of Director's utterances will help the agent to create more natural interaction patterns, which might receive a better subjective rating by the human dialogue partner and in the end might even achieve a better overall game performance, as ambiguities can be resolved quicker and the flow of communication can be more efficient.

6.6.4 Conclusion - RDG-Image

In this chapter, we have defined and evaluated a sequential approach to incremental DA segmentation and classification. Our approach utilizes prosodic, lexico-syntactic and contextual features, and achieves an encouraging level of performance in offline analysis and in policy simulations. We have presented our results in terms of existing metrics for DA segmentation and also introduced additional metrics that may be useful to other system builders. This work has been published in

Manuvinakurike et al. (2016). In future work, we will continue this line of work by incorporating dialogue policies for additional DA types into the interactive agent. We expect that the resulting new capabilities will help improve user perceptions of our agent by enabling richer and more natural patterns of interaction.

6.7 New Domain: Conversational Image editing

Photographs[12] have emerged as a means for sharing information, effective storytelling, preserving memories, and brand marketing among many other applications. The advent of photo-centric social media platforms such as Instagram, Snapchat, etc. along with easy access to high quality photo-taking devices has only made photographs a more powerful medium. Photographs are often edited with the intention of improving their quality (e.g., fixing the lighting), for use in a narrative (e.g., for an ad campaign), for alteration (e.g., removing objects from the image), for preservation of memories (by restoring old photographs), and for other reasons. Social media platforms such as Snapchat, Instagram, etc. support popular and extensively used editing methods called presets (or filters). Such presets can also be found in cameras on many current smartphones, and can be applied to photographs almost instantaneously. However, image editing is far from choosing the right filter or preset values. Photo editing is a complex task often involving diligently and skillfully executed steps that require expertise. Seeking professional help for editing photographs is common, and can be seen in popular forums such as Reddit Photoshop Request (https://www.reddit.com/r/PhotoshopRequest/) and Zhopped (http://zhopped.com/), where users post their photographs and request help from

[12] "Photographs" and "images" as terms are used interchangeably in this work henceforth. They both refer to digital photographs captured in any popular format that can be rendered on the computer screen using popular programs.

professionals. The professionals then either volunteer for free or do the job for a fee. The process typically starts with users publicly posting their request and the photograph they desire to be edited. These requests are formulated in an abstract manner using natural language (Ex: "I love this photo from our trip to Rome. Can someone please remove my ex from this photo? I am the one on the right."), rather than intricate multi-step instructions (Ex: "Free select the person on the left, replace the region with the building on the bottom left using replacement tools, fix the blotch by the healing tool..."). The professionals download these photographs, edit them, and post them back. They have knowledge about the image editing tool used, skills, time, and artistic creativity to perform the changes. If the users are not happy with the results, they post their modification requests, and then the professionals incorporate these changes and post the updated photographs. While these forums are popular, such methods have a few drawbacks. Because the expert editors edit the photographs without the requester being able to see the changes being performed in real time, (i) the users are not able to provide real-time feedback, (ii) it is hard for the users to provide requests for all needed modifications, and (iii) the professional editors cannot ask for minor clarifications while editing the photographs. These drawbacks often result in modifications that do not match the users' expectations. The alternative solution of the users performing the edits themselves is difficult and time consuming as the image editing tools have a steep learning curve.

Our ultimate goal is to develop a conversational agent that can understand the user requests, perform the edits, guide the user by providing suggestions, and respond in real time. In this chapter we present a novel corpus that captures the conversation between the user who wants to edit a photograph and the expert human wizard who performs the edits (playing the role of a future dialogue system).

We introduce a novel annotation scheme for this task, and discuss challenging subtasks in this domain. We focus specifically on incremental dialogue act detection. Conversational image editing combines spoken language, dialogue, and computer vision, and our real-world domain extends the literature on domains that are at the intersection of language and computer vision.

Conversation in the context of visual information has been studied for a long time. Clark & Wilkes-Gibbs (1986) studied reference resolution of simple figures called tangrams. Kennington & Schlangen (2015) performed incremental understanding and incremental reference resolution respectively in a domain of geometric shape descriptions, while Schlangen et al. (2016) resolved references to objects in real-world example images. Much work has been done in the context of gamified scenarios where the interlocutors interact and resolve references to real-world objects (Kazemzadeh et al. (2014); Paetzel et al. (2014); Manuvinakurike & DeVault (2015)). Also, such gamified scenarios have served as platforms for developing/learning incremental dialogue policies regarding whether the system should respond immediately or wait for more information (Paetzel et al. (2015); Manuvinakurike et al. (2017)). Referential domains in the context of dialogue have also been studied using virtual reality technologies and spatial constraints (Stoia et al. (2008); Das et al. (2018)) as well as robots (Whitney et al. (2016); Skantze (2017)).

Other gamified real-world scenarios involve object arrangement (DeVault & Stone (2009)), puzzle completion (Iida et al. (2010); Takenobu et al. (2012)), map navigation (Anderson et al. (1991); Lemon et al. (2001); Johnston et al. (2002)), furniture-buying scenarios (Di Eugenio et al. (2000)), and treasure-hunt tasks in a virtual environment (Byron & Fosler-Lussier (2006)). A multimodal interface for image editing combining speech and direct manipulation was developed by Laput et al. (2013). With this interface a user can for example select a person's hat in an image and say "this is a hat". Then the system learns to associate the tag "hat" with the selected region of the image.

A more recent direction of research involving dialogue and vision has been in the context of answering factual questions on images (Das et al. (2017); Antol et al. (2015)) using the MS COCO dataset (Lin et al. (2014)). The task may also involve a gamified scenario with the interlocutors playing a yes-no question-answer game as in de Vries et al. (2017). In these works the focus is less on the dialogue aspects and more on the factual aspects of the images, i.e., if an object is present or what a certain component of the image is. Mostafazadeh et al. (2017) extended this line of work with conversations grounded on images. Furthermore, Huang et al. (2016) built a dataset of images with corresponding descriptions in sequence, for the task of visual storytelling. Some recent work has started investigating the potential of building dialogue systems that can help users efficiently explore data through visualizations (Kumar et al. (2017)).

The problem of intent recognition or dialogue act (DA) detection has been extensively studied. Below we focus on recent work on DA detection that employs deep learning. People have used recurrent neural networks (RNNs) including long short term memory networks (LSTMs), and CNNs (Kalchbrenner & Blunsom (2013); Li & Wu (2016); Khanpour et al. (2016); Shen & Lee (2016); Ji et al. (2016); Tran et al. (2017)). The works that are most similar to ours are by Lee & Dernoncourt (2016) and Ortega & Vu (2017) who compared LSTMs and CNNs on the same datasets. However, neither Lee & Dernoncourt (2016) nor Ortega & Vu (2017) experimented with incremental DA detection as we do. Petukhova & Bunt (2014) built models for incremental DA recognition. Most related to our work, DeVault et al. (2011a) built models for incremental interpretation and prediction of utterance meaning and showed that incremental processing helps save time. In this work, we add to the literature by lending support to the generality of the result that incremental NLU can save time.

6.8 Data: Conversational Image Editing

The task of image editing is challenging for the following reasons: (i) The user needs to understand whether changes applied to a given image fit the target narrative or not. (ii) Image editing is a time consuming task. The user typically experiments with various features often undoing, redoing, altering in increments, or even completely removing previously performed edits before settling on the final image edit. (iii) Users may know at an abstract level what changes they want to perform, but be unaware of the image editing steps or parameters that would produce the desired outcome. For example, a person's face in a photo may look flushed, but users may not know that adjusting the saturation and the temperature settings to some specific values will change the photo to match their expectations. (iv) Image editing tools are complicated due to the availability of innumerable options, and can have a steep learning curve often requiring months of training. Users may not be fully aware of the features and the functionality that are supported by the given image editing tool.

Our task is particularly well suited for spoken dialogue research, and in particular incremental dialogue processing. Besides understanding the user utterances and mapping them to commands supported by the tool, the task also involves a high degree of interactivity that requires real-time understanding and execution of the user requests. For instance, in a dialogue setting and in order to increase the saturation value, the user can utter "more, more, more" until the desired target value has been set. An annotation scheme should support such incremental changes as well as requests for new changes, updates of ongoing changes (including undoing and redoing), comparing the current version of the image with previous versions, and question-answer exchanges between the user and the wizard (including suggestions, clarifications, and feedback).

6.8.1 Data Collection

We collected spoken dialogues between users (who request image edits) and wizards (who perform the edits); a total of 28 users and 2 wizards participated in the collection. Prior to data collection, our wizards were trained in executing a range of image edits.

We tested several image editing tools and found that very simple tools that did not support a high degree of functionality resulted in extremely restrictive dialogues lacking variety. Conversely, tools with rich functionality, such as Adobe Photoshop or GNU GIMP, resulted in user image edit requests that required hours to complete. Such interactions yielded creative image edit requests but did not yield timely dialogue phenomena. The tool ultimately used for image editing in this study was Adobe Lightroom. This tool produced diverse and highly interactive dialogues for image editing. The tool is popular among photographers and supports a wide variety of functionality. Users were able to make creative requests with few restrictions, and these requests could often be executed rapidly. The wizards were trained expert image editors who knew how to use the tool well.

6.8.2 Experiment Setup

The recruited users were given images (digital photographs) sampled from the Visual Genome dataset (Krishna et al. (2017)) which in turn were sampled from the MS COCO dataset (Lin et al. (2014)). The images used for data collection were selected to reflect real-world scenarios. Thus the images sampled were representative of images regularly edited by users. The photos selected from the sampled image datasets were based on observations of 200 random request submissions from Zhopped and Reddit Photoshop forums. The forum submissions were often about eight high-level categories of images: animals, city scenes, food, nature/landscapes, indoor scenes, people, sports, and vehicles. Thus we selected images from the MS COCO dataset that fit into at least one of these eight categories.

Users were given one photograph from each category in an experiment session. They were given time to think about the changes they wanted to perform before the dialogue session, and were informed about the tool that was going to be used and the fact that it did not support complex functionality. If they were unsure of what functionality was supported they were instructed to ask the wizard. Users were asked to perform as many edits as they desired per image. Participants were encouraged (but not required) to participate for 40 minutes, and communicated via remote voice call. Users did not have the freedom to perform the edits themselves. Any edits they wished to be performed on the image had to be conveyed to the wizard through voice. This was to ensure that all the changes that the users wanted to achieve were captured in spoken language. The wizard responded to the requests in a natural human-like manner. The screen share feature was enabled on the wizard's screen so that the user could see in real time the wizard's edits on the image. While users were not explicitly told that the wizard was human, this was obvious due to the naturalness of the conversation.

The interaction typically started with the user describing a given image to the wizard. The wizard was not aware of the images provided to the user. The wizard chose the image from the available images based on the user description; following

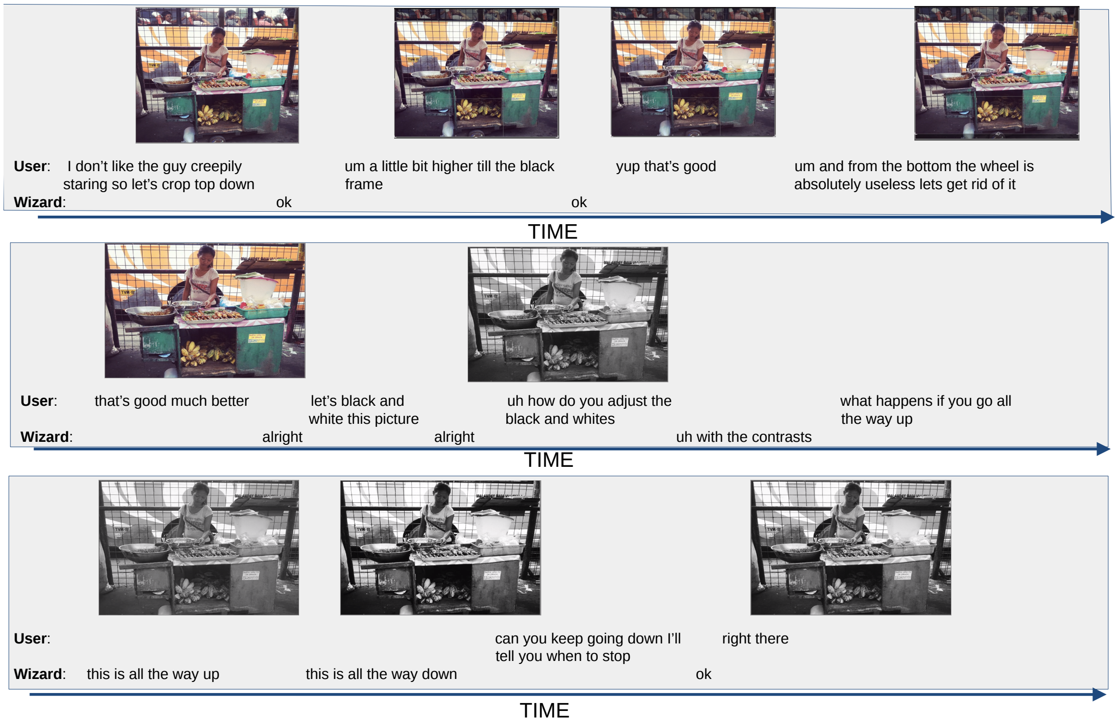
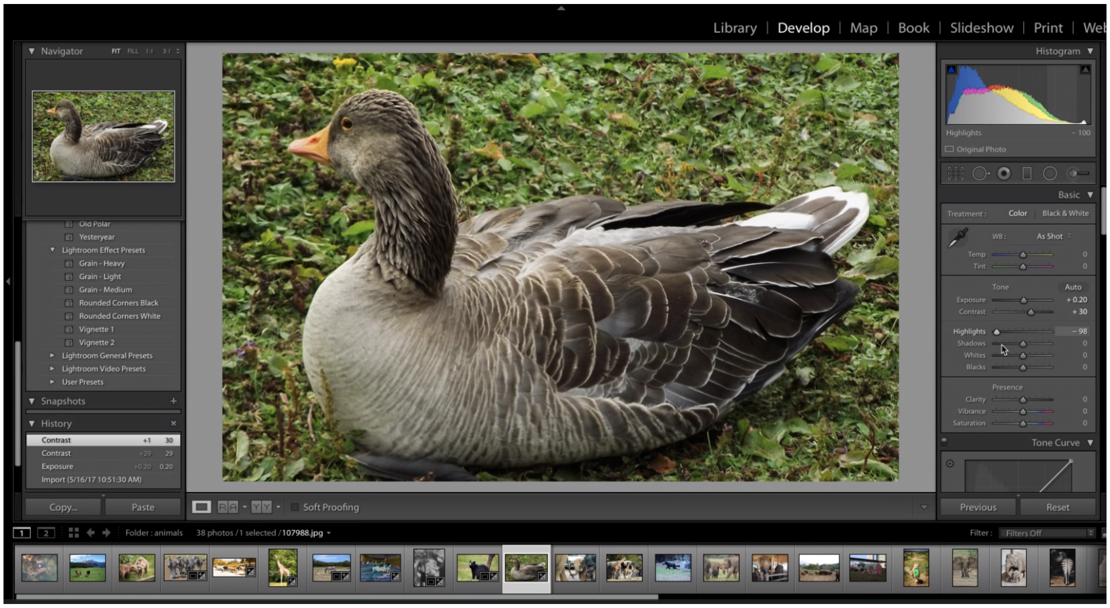

User: I don't like the guy creepily staring so let's crop top down
Wizard: ok
um a little bit higher till the black frame
ok
yup that's good
um and from the bottom the wheel is absolutely useless lets get rid of it
TIME
User: that's good much better
Wizard: alright
let's black and white this picture
alright
uh how do you adjust the black and whites
uh with the contrasts
what happens if you go all the way up
TIME
User:
Wizard: this is all the way up
this is all the way down
can you keep going down I'll tell you when to stop
ok
right there
TIME

Figure 6.5: Sample interaction between the user and the wizard.

user confirmation, the image was then loaded for editing. The image editing session generally began with the user describing desired changes to the image in natural language. The wizard interpreted the request provided by the user and performed these edits on the image. The interaction continued until the user was satisfied with the final outcome. Figure 6.5 shows an example of an interaction between the user and the wizard. Figure 6.6 shows the interface as seen by the user and the wizard.

6.8.3 Data Preparation

The conversations were recorded using the OBS software which is a free open-source program for streaming video and audio. Then the audio data were extracted from the videos. Transcription was done on small audio chunks which was more



Contrast +1 30
Contrast +29 29
Exposure +0.20 0.20
Import (5/16/17 10:51:30 AM)

Folder : animals 38 photos / 1 selected / 107988.jpg

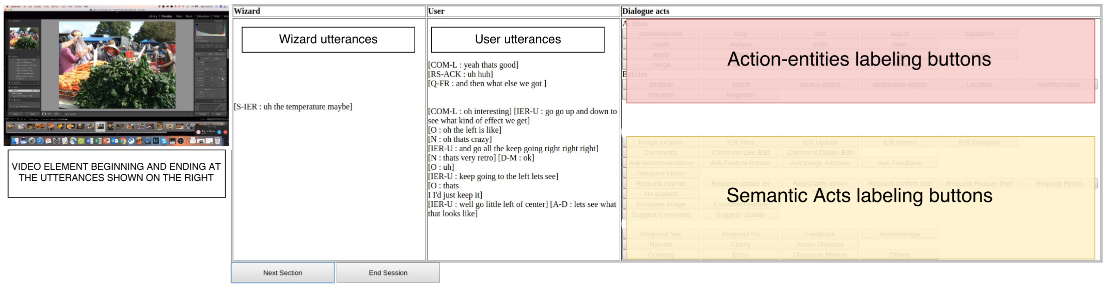
Highlights - 100

Treatment : Color | Black & White
WB : As Shot
Temp 0
Tint 0
Tone Auto
Exposure + 0.20
Contrast + 30
Highlights – 98
Shadows 0
Whites 0
Blacks 0
Presence
Clarity 0
Vibrance 0
Saturation 0

Figure 6.6: The interface as seen by the user and the wizard. We use Adobe Lightroom as the image editing program.

convenient and faster than transcribing long clips. The small audio clips were obtained by splitting the audio at the silence points using the webRTC Voice Activity Detection (VAD)[13]. Transcriptions were performed using the Amazon Mechanical Turk platform. The transcribed audio data were then validated and annotated with DAs, actions, and entities using a custom-built web annotation tool. The annotations were performed by two expert annotators who were well versed with the new annotation scheme that we developed (see 6.8.4). Figure 6.7 shows the tool that was built for annotating the dataset. The tool was web-based, with the annotators being able to see the video, audio interaction, and the transcriptions shown in small chunks (typically around 45 seconds) which were to be annotated by selecting the text and the corresponding DA. In total 28 users contributed to 129 dialogues with 8890 user utterances, 4795 wizard utterances, and 858 minutes of speech. The total number of tokens in the user and wizard utterances is 59653 and

[13] https://pypi.python.org/pypi/webrtcvad


VIDEO ELEMENT BEGINNING AND ENDING AT THE UTTERANCES SHOWN ON THE RIGHT
Wizard
Wizard utterances
User
User utterances
Dialogue acts
Action-entities labeling buttons
Semantic Acts labeling buttons
Next Section
End Session

[S-IER : uh the temperature maybe]

[COM-L : yeah thats good]
[RS-ACK : uh huh]
[Q-FR : and then what else we got]

[COM-L : oh interesting] [IER-U : go go up and down to see what kind of effect we get]
[O : oh the left is like]
[N : oh thats crazy]
[IER-U : and go all the keep going right right right]
[N : thats very retro] [D-M : ok]
[O : uh]
[IER-U : keep going to the left lets see]
[O : thats
I I'd just keep it]
[IER-U : well go little left of center] [A-D : lets see what that looks like]

Figure 6.7: Web annotation tool used to annotate the dialogue. The figure shows the wizard and the user utterance aligned with time. The video element is shown to the left. The annotation is performed by highlighting the text and clicking the buttons corresponding to the DA.

# users	28
# dialogues	129
# user utterances	8890
# Wizard utterances	4795
# time (raw)	858 min
# user tokens	59653
# user unique tokens	2299
# Wizard tokens	26284
# Wizard unique tokens	1310
# total unique tokens	2650

Table 6.12: Data statistics.

26284 respectively. Also, there are 2299 unique user tokens, 1310 unique wizard tokens, and 2650 total unique tokens. Table 6.12 shows the statistics of the data.

6.8.4 Annotation Scheme

We designed a set of 26 DA types, for the ultimate goal of building a conversational agent. Some of the DAs were motivated by (Bunt (2009)), while others are specific to the domain of conversational image editing. DAs apply to segmented utterances, with each segment annotated with one DA. Note that an utterance is defined as a portion of speech preceded and/or followed by a silence interval greater than 300ms. The DA types are summarized in Table 6.13. Utterances from both the wizard and the user were annotated. In order to measure the validity of the annotation scheme we calculated inter-rater reliability for DA labeling by having two expert annotators annotate a single dialogue session of 20 minutes; kappa was 0.81 which indicates high agreement. Below we elaborate on the DA types.

Image Edit Requests (IER). Image edit requests are grouped into four categories. New requests (IER-N) are edits that the users desire to see in the image, which are different from previous requests. These requested changes are either abstract ("it's flushed out, can you fix it?") or exact ("change the saturation to 20%"). The wizard interprets these requests and performs the changes. Update requests (IER-U) are refinements to a previous request (users often request updates until the target is achieved). These include the addition of more details ("change it to 50%") to the IER-N ("change the saturation"), issuing corrections to the IER ("can you reduce the value again?"), modifiers (more, less), etc. Revert requests (IER-R) occur when users want to undo the changes done to the image until a certain point. The IER-R act is used if the user reverts the complete changes performed, compared to only changing the values. For example, if the user is modifying the saturation of the image and across multiple turns changes the value of saturation from 20% to 30% and back to 20%, the user's action is labeled as IER-U. If the user wants all the saturation changes to be undone, the user's action is labeled as IER-R. Compare requests (IER-C) occur when users want to compare the current version of the image to a previous version (before the most recent changes took place). The image edit requests IER-N and IER-U are labeled further with action and entity labels, which specify the nature of the edit request (the use of actions and entities is inspired by the intents and entities of Williams et al. (2015)). These labels serve as an intermediary language to map a user's utterance to executable commands that

Dialogue Act	**Description**
Image Edit Request (IER)	user requests changes to the image (IER-N, IER-U, IER-R, IER-C)
Comment (COM)	user comments on the image or edits (COM-L, COM-D, COM-I)
Request Recommendation (RQR)	user requests recommendation from the wizard on editing ideas
Question Feature (QF)	user asks question on the functionality of the editing tool
Question Image Attribute (QIA)	user asks question about the image
Request Feedback (RF)	user requests feedback about the image edits
Image Location (IL)	user & wizard locate the image at the beginning
Action Directive (AD)	user asks wizard to act on the application, e.g., "click the button"
Finish (FIN)	user wants to end the editing session
Suggestions (S)	wizard suggests ideas for editing the image
Request IER (RQIER)	wizard requests user to provide IER
Confirm Edit (CE)	wizard confirms the edit being performed
Feature Preference (FP)	wizard requests which tool option to use for achieving the user edits
Narrate (N)	wizard gives narration of the steps being performed
Elucidate (E)	these are wizard responses to QF & QIA
No Support (NS)	wizard informs user that the edit is not supported by the tool
Respond Yes/No (RSY/RSN)	yes/no response
Acknowledge (ACK)	acknowledgment
Discourse Marker (DM)	discourse marker
Other (O)	all other cases

Table 6.13: Dialogue act types (Manuvinakurike et al. (2018a,b)).

can be carried out in an image editing program. Actions are a predefined list of 18 functions common to most image editing programs, such as cropping. Each IER contains at most one action. The entities provide additional information without which the action cannot be applied to the given image. The entities are made up

Segments	**Dialogue Act**	**Action**	**Attribute**	**Loc/Obj**	**Mod/Val**
uh	O	-	-	-	-
make the tree brighter	IER-N	Adjust	brightness	tree	-
like a 100	IER-U	Adjust	brightness	tree	100
nope too much	COM-D	-	-	-	-
perfect	COM-L	-	-	-	-
let's work on sharpness	IER-N	Adjust	sharpness	-	-

Table 6.14: Example annotations of dialogue acts, actions, and entities.

of attributes (saturation, contrast, etc.), region/object (location where the image edit action is to be applied), value (modifiers or cardinal values accompanying the action-attribute). Table 6.14 shows example annotations. So when the user says "make the tree brighter to 100", it is important to understand the exact user's intent and to translate this into an action that the image editing tool can perform. For this reason we use action-entities tuples <action, attribute, location/object, value>. The user utterances are mapped to DAs and then to a pre-defined set of image action-entities tuples which are translated into image editing actions. It is beyond the scope of this work to perform the image editing. We focus instead only on the DA detection problem which does not require any information about actions and entities.

Comments (COM). Three types of user comments are annotated: (i) Like comments (COM-L) where users show a positive attitude towards the edits that are being performed ("that looks interesting", "that's cool"). (ii) Dislike comments (COM-D) are the opposite of like comments ("I don't like that", "I don't think it's what I want"). (iii) Image comments (COM-I) are neutral user comments such as comments on the image ("it looks like a painting now", "her hair looks pretty striking").

Suggestions (S). Suggestions were the recommendations issued by the wizards to the users recommending the image editing actions. Suggestions also included the utterances that were issued with the goal of helping the user achieve the final image edits desired. The wizard provides new suggestions (S-N), e.g., “do you want to change the sharpness on this image?”. The wizard could also provide update suggestions for the current request under consideration (S-U), e.g., “sharpness of about 50% was better”.

Requests & Responses. The user may ask the wizard to provide suggestions on the IERs. These are labeled as “Request” acts. “Yes” and “no” responses uttered in response to the wizard’s suggestions are labeled as RS-Y or RS-N.

Other user actions are labeled as questions about the features supported by the image editing tool, clarifications, greetings, and discourse markers. In total there are 26 DA labels, including the DA “Other (O)” which covers all of the cases that do not belong in the other categories. Table 6.15 shows the prevalence of our 26 DAs in the corpus (percentage of words and of utterance segments in the corpus per DA). For the task of incremental DA detection we focus only on the user utterances only, and in particular, in classifying user utterances into one of 10 labels: IER-N, IER-U, IER-R, IER-C, RS-Y, RS-N, COM-L, COM-D, COM-I, and O.

Table 6.16 shows example utterances for some of the most frequently occurring DAs in the corpus. In these examples it can be seen that, with the exception of 3, all the other DAs can be identified with some degree of certainty without waiting for the user to complete the utterance. Also, Figure 6.8 shows example IERs. One of the motivations for our work is to identify the right DA at the earliest time. Not only is this more efficient but also more natural. The human wizard can begin to take action even before the utterance completion, e.g., in utterance 1 (Table 6.16) the wizard clicks the “vignette” feature in the tool before the user has finished uttering their request. Another goal is to measure potential savings in time gained through incremental processing, i.e., how much we save in terms of number of words

Dialogue Act	**% Words**	**% Utterance Segments**	**Dialogue Act**	**% Words**	**% Utterance Segments**
IER-N	19.4	9.2	FIN	1.5	1.0
IER-U	16.3	12.5	S	4.7	4.0
IER-R	1.0	0.8	RQIER	2.1	2.6
IER-C	0.5	0.3	CE	1.6	1.9
COM-L	4.9	6.0	FP	0.1	0.1
COM-D	1.8	1.5	N	3.1	4.2
COM-I	2.5	1.5	E	1.3	0.7
RQR	0.7	0.0	NS	1.0	0.6
QF	1.1	0.6	RSY	2.3	6.8
QIA	0.3	0.2	RSN	0.9	1.2
RF	0.0	0.0	ACK	6.4	17.6
IL	3.0	1.5	DM	2.3	6.5
AD	4.8	3.9	O	16.4	14.8

Table 6.15: Percentages of words and of utterance segments for each DA type; "0.0" values are close to 0.

when we identify the DA earlier rather than waiting until the full completion of the utterance, without sacrificing performance.

The transitions between DAs for the user acts were analyzed (for this analysis we ignore the label "Other-O"). We found that the most common transition was from IER-U to IER-U. This is particularly interesting as it shows that users provide a series of updates before attaining the final image edits. This transition was more common than IER-N to IER-U, which is the second most frequently found transition. Users were found to like the image edits after IER-Us, and after issuing a COM-L (like edit) comment they usually move on to the next IER. We also found that when users disliked the edits (COM-D) they did not entirely cancel the edits but continued updating (IER-U) their requests until the final image version fit their needs. Transitions from IER-N to IER-N were also common; users could issue a complete new image edit request IER-N and then move on to another new image edit request IER-N.

In this section, we described our data collection process and novel DA labeling scheme. The corpus supports research and development in areas such as incremental

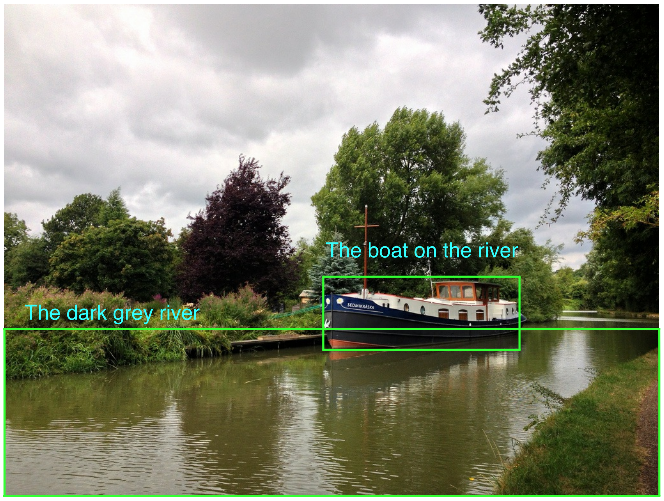
	Utterance	Tag
1	add a vignette since it's also encircled better	IER-N
2	can we go down to fifteen on that	IER-U
3	go back to .5	IER-U
4	actually let's revert back	IER-R
5	can you compare for me before and after	IER-C
6	I like it leave it there please	COM-L
7	no I don't like this color	COM-D

Table 6.16: Examples of some of the most commonly occurring DAs in our corpus.

intent recognition (Manuvinakurike et al. (2018a)), dialogue modeling, and dialogue state tracking. Furthermore, the dataset is constructed using richly annotated images, which makes it an ideal platform for studying reference resolution in images, question answering, image-grounded dialogue modeling, tracking user likeness of images, and user modeling (providing suggestions to users depending on their preferences and knowledge of the tool).

6.9 Model Design: Incremental Dialogue Act Identification

For our experiments we use a training set sampled randomly from 90% of the users (116 dialogues for training, 13 dialogues for testing). We use word embedding features whose construction is described in Section 6.9.1. Figure 6.11 shows the visual presentation of the utterances embeddings using t-SNE (Maaten & Hinton (2008)).



Tag	User Edit Requests
IER-N	I want to um add more focus on the boat
IER-N	can you make the water uh nicer color
IER-N	uh can we crop out uh little bit off the bottom
IER-N	is there a way to add more clarity
IER-N	can we adjust the shadows
IER-U	more [saturation]
IER-U	can we get rid of the hints of green in it
IER-U	bluer
IER-U	little bit more from the left [crop]
IER-R	can you unfocus it
IER-C	can you show me before and after

Figure 6.8: Example user edit requests. Only two bounding boxes are labeled in the image for better reading. The actual images have more extensive object labels.

6.9.1 Constructing Word Embeddings

We convert the words into vector representations to train our deep learning models (and a variation of the random forests). We use out-of-the-box word vectors available in the form of GloVe embeddings (Pennington et al. (2014)) (trained with Wikipedia data), or we employ fastText (Bojanowski et al. (2016)) to construct embeddings using the data from the Visual Genome image region description phrases, the dialogue training set collected during this experiment, and other data related to image editing that we have collected (image edit requests out of a dialogue context). From now on these embeddings trained with fastText will be referred to as “trained embeddings”.

As we can see in Table 6.17, for models E (LSTMs) and I (CNNs) we use word embeddings trained with fastText on the aforementioned datasets. The Vanilla LSTM (model D) does not use GloVe or trained embeddings, i.e., there is no dimensionality reduction. Model H (CNNs) uses GloVe embeddings. The vectors used in this work (both GloVe and trained embeddings) have a dimension of 50. For trained embeddings, the vectors were constructed using skipgrams over 50 epochs with a learning rate of 0.5.

Recent advancements in creating a vector representation for a sentence were also evaluated. We used the Sent2Vec (Pagliardini et al. (2018)) toolkit to get a vector representation of the sentence and then used these vectors as features for models G and J. Note that LSTMs are sequential models where every word needs a vector representation and thus we could not use Sent2Vec.

6.9.2 Model Construction

We use Weka (Hall et al. (2009)) for the Naive Bayes and Random Forest models, Mallet (McCallum (2002)) for the CRF model (linear chain), and TensorFlow for the LSTM and CNN models. The models B, C, D, and F in Table 6.17 use bag-of-words features. The CNN has 2 layers, with the first layer containing 512 filters and the second layer 256 filters. Both layers have a kernel size of 10 and use ReLU activation. The layers are separated by a max pooling layer with a pool size of 10. The dense softmax is the final layer. We use the Adam optimizer with the

	Model	Accur
A	Baseline (Majority) *	0.32
B	Naive Bayes *	0.41
C	Conditional Random Field *	0.51
D	LSTM (Vanilla) *	0.53
E	LSTM (trained word embeddings) *	0.55
F	Random Forest *	0.72
G	Random Forest (with Sent2Vec)	0.73
H	CNN (GloVe embeddings)	0.73
I	CNN (trained word embeddings)	0.74
J	CNN (Sent2Vec)	0.74

Table 6.17: DA classification results for complete speech segments using the transcripts. * indicates significant difference ($p < 0.05$) between the best performing models (I and J) and the other models.

categorical cross entropy loss function. The LSTM cell is made up of 2 hidden layers. We use a dropout with keep_prob = 0.1. We put the logits from the last time steps through the softmax to get the prediction. We use the same optimizer and loss function as for the CNN since they were found to be the best performing.

Table 6.17 shows the DA classification accuracy for all models on our test set. Here we assume that we have the correct utterance segmentation for both the training and the test data. Note that because of the “Other” DA all words in a sentence will belong to a segment and a DA category.

6.9.3 Results

Table 6.18 shows the savings in terms of overall number of words and average number of words saved per sentence, for each DA in the corpus.

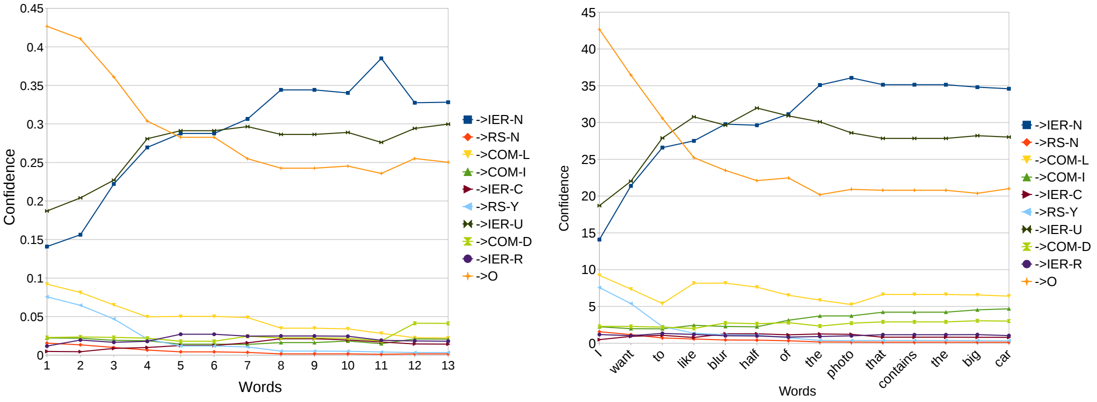


Figure 6.9: Confidence contours based on every word. The correct tag is IER-N. The confidence contours at the word level take time to stabilize.

Figure 6.9 shows the confidence curves for predicting the DA with the progression of every word. From the figure on the right it is clear that after listening to the word “photo” the classifier is confident enough that the user is issuing the IER-N command. Here the notion of incrementality is to predict the right DA as early as possible and evaluate the savings in terms of the number of words. While from this example it is clear that the correct DA can be identified before the user completes the utterance, it is not clear when to commit to a DA. The trade-off involved in committing early is often not clear. Table 6.18 shows the maximum savings that can be achieved in an ideal scenario where an oracle (an entity informing if the prediction is correct or wrong as soon as the prediction is made) identifies the earliest point of predicting the correct DA.

The method used for calculating the savings is shown in Table 6.19. In this example for the utterance “I think that’s good enough”, we feed the classifier the utterances one word at a time and get the classifier confidence. The class label with the highest score is obtained. Here the oracle tells us that we could predict the correct class COM-L as soon as “I think that’s good” was uttered and thus the word savings would be 1 word.

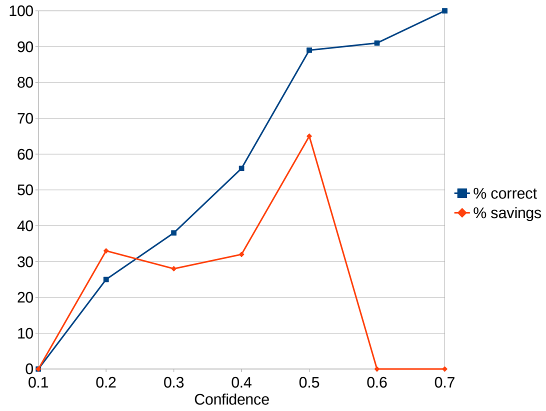


Figure 6.10: % savings (for correct predictions) and accuracy (% correct) of incremental predictions of DAs as a function of confidence level.

Tag	**% Overall Word Savings**	**Average Word Savings per Utterance**
IER-N	37	3.96
IER-U	39	2.72
IER-R	41	1.63
IER-C	40	1.69
COM-L	36	1.13
COM-D	41	1.38
COM-I	37	2.56
RS-Y	28	0.34
RS-N	37	0.69
O	47	3.95

Table 6.18: Percentage of overall word savings and average number of words saved per utterance, for each DA.

However, in real-world scenarios the oracle is not present. We use several confidence thresholds and measure the accuracy and the savings achieved in predicting the DA without the oracle. For the predictions in the test set we get the accuracy for each of the thresholds. Then if the predictions are correct, we calculate the savings. Thus Figure 6.10 shows the word savings for each confidence threshold when the predictions are correct for that threshold.

Utterance	**Max conf**	**Class**
I	0.2	O
I think	0.3	O
I think that's	0.3	O
I think that's good	**0.5**	**COM-L**
I think that's good enough	**0.5**	**COM-L**

Table 6.19: Example incremental prediction. The correct label is COM-L. Columns 2 and 3 show the maximum confidence level and model prediction after each word is uttered.

So in the example of Table 6.19, for a confidence threshold value of 0.4, we extract the class label assigned for the utterance once the max confidence score exceeds 0.4. In this case once the word "good" was uttered by the user the confidence score assigned (0.5) was higher than the threshold value of 0.4 and we take the predicted class as COM-L. The word savings in this case is 1 word and our prediction is correct. But for a confidence threshold value of 0.2, our prediction would be the tag O which would be wrong and there would be no time savings. Figure 6.10 shows that as the confidence threshold values increase the accuracy of the predictions rises but the savings decrease.

Researchers have used simulations (Paetzel et al. (2015)), classifiers (DeVault et al. (2011a)) or a reinforcement learning policy (Manuvinakurike et al. (2017)) to learn the right points of interrupting the user which are dependent on the language understanding confidence scores. Here we do not focus on learning such policies. Instead, our work is a precursor to learning an incremental system dialogue policy.

We presented "conversational image editing", a novel real-world application domain, which combines dialogue, visual information, and the use of computer vision. We discussed why this is a domain particularly well suited for incremental dialogue processing. We built models for incremental intent identification based on

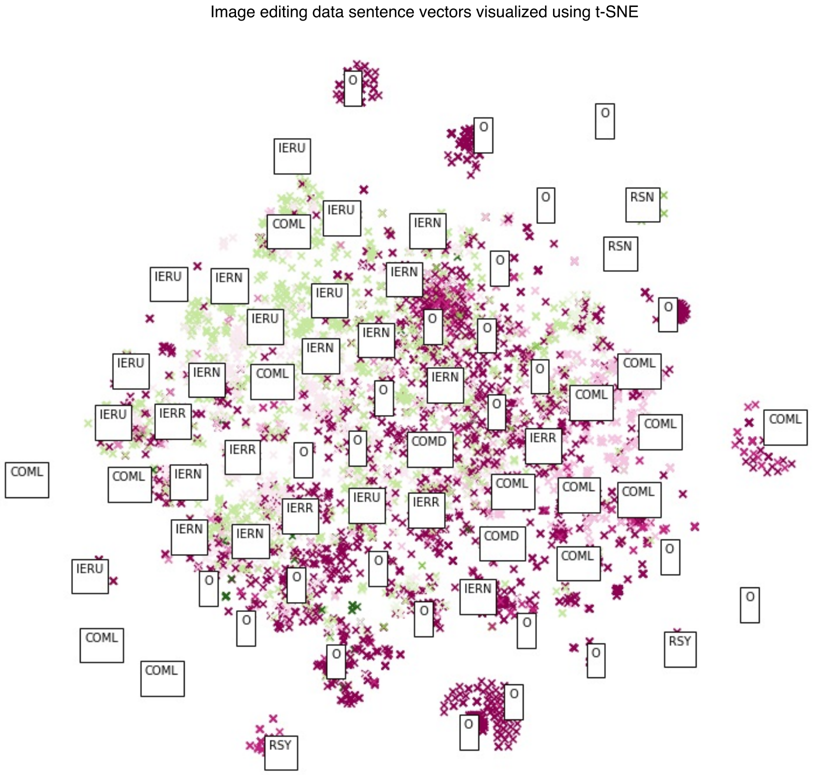


Figure 6.11: Visualization of the sentence embeddings of the user utterances used for training. The t-SNE visualizations after half-way through the utterances are shown. The utterances that have the same DAs can be seen grouping together. This shows that the complete utterance is not always needed to identify the correct DA.

deep learning and traditional classification algorithms. We calculated the impact of varying confidence thresholds (above which the classifier's prediction is considered) on classification accuracy and savings in terms of number of words. Our experiments add to previous evidence (DeVault et al. (2011a)) that incremental intent processing could be more efficient for the user and save time in accomplishing tasks in SDS.

6.10 Contributions

In this chapter, we developed methods for incremental language understanding towards the development of SDS. We developed novel models for incremental segmentation and DA detection. It has not been shown before in the literature what the impact of such an incremental operation (segmentation and DA detection) is in an SDS pipeline. In this chapter, we showed that such an operation in an incremental SDS contributes towards increasing processing capabilities without impacting the speed of processing. We have shown that including incremental language understanding capabilities in the SDS pipeline could help save time in SDS. We then applied incremental segmentation and DA detection to a novel real-world domain (conversational image editing) and showed that such processing helps achieve savings in terms of time when detecting user intentions. Such incremental processing could help develop more efficient dialogue systems in the near future. The discussions in this chapter have been published in (Manuvinakurike et al. (2018a); Manuvirakurike et al. (2018); Manuvinakurike et al. (2018b); Brixey et al. (2018)). In the next chapter, I will discuss performing segmentation and DA detection using visually grounded models.

Chapter 7

Vision, Language & Incrementality

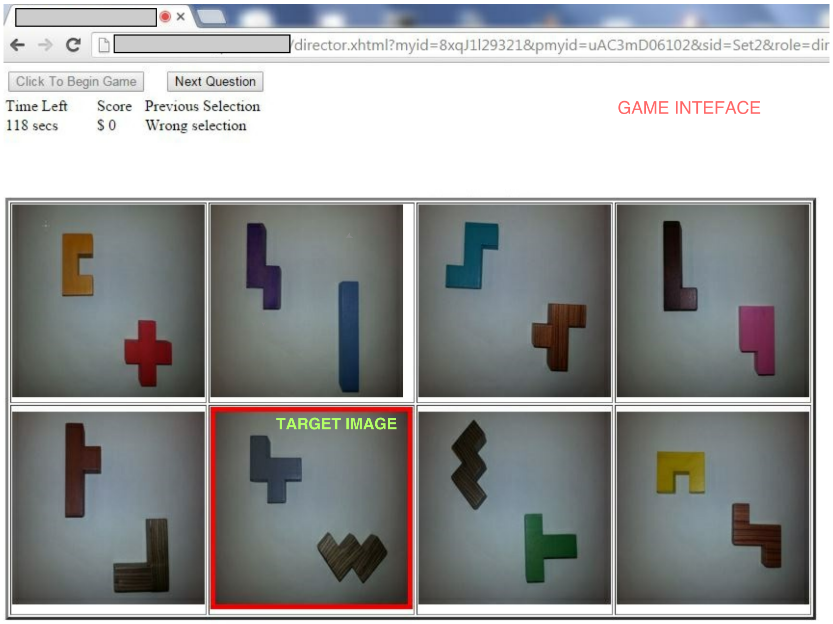
"Only thing worse than being blind is having sight but no vision"

– Helen Keller, *Author, activist, lecturer*

7.1 Introduction

In this chapter, we present and evaluate a language processing pipeline that enables an automated system to detect and understand complex referential language about visual objects depicted on a screen. This is an important practical capability for present and future interactive SDS. There is a trend toward increasing deployment of SDS for smartphones, tablets, automobiles, TVs, and other settings where information and options are presented on-screen along with an interactive speech channel in which visual items can be discussed (Celikyilmaz et al. (2014)). Similarly, for future systems such as smartphones, quadcopters, or self-driving cars that are equipped with cameras, users may wish to discuss objects visible to the system in camera images or video streams.

A challenge in enabling such capabilities for a broad range of applications is that human speakers draw on a diverse set of perceptual and language skills to communicate about objects in situated visual contexts. Consider the example in Figure 7.1, drawn from the corpus of RDG-Pento games (discussed further in



[redacted]

[redacted]/director.xhtml?myid=8xqJ1l29321&pmyid=uAC3mD06102&sid=Set2&role=dir

Time Left 118 secs

Score $ 0

Previous Selection Wrong selection

Director: this one is kind of a uh a blue
T and a wooden w sort of the
T is kind of malformed

Matcher: okay got it

Figure 7.1: In an RDG-Pento game, the Director is describing the image highlighted in red (the *target image*) to the Matcher, who tries to identify this image from among the 8 possible images. The figure shows the game interface as seen by the Director including a transcript of the Director's speech.

Section 7.4). In this example, a human in the *Director* role describes the visual scene highlighted in red (the *target image*) to another human in the *Matcher* role. The scene description is provided in one continuous stream of speech, but it includes three functional segments each providing different referential information: [*this one is kind of a uh a blue T*] [*and a wooden w sort of*] [*the T is kind of malformed*]. The first and third of these three segments refer to the object at the top left of the target image, while the middle segment refers to the object at bottom right. An ability to detect the individual segments of language that carry information about individual referents is an important part of deciphering a scene description like this. Beyond detection, actually understanding these referential segments in context seems to require perceptual knowledge of vocabulary for colors, shapes, materials and hedged descriptions like *kind of a blue T*. In other game scenarios, it is important to be able to understand plural references like *two brown crosses* and relational expressions like *this one has the L on top of the T*.

A variety of vocabulary knowledge is needed, as different speakers may describe individual objects in very different ways (the object described as *kind of a blue T* may also be called *a blue odd-shaped piece* or *a facebook*). When many scenes are described by the same pair of speakers, the pair tends to entrain or align to each other's vocabulary (Garrod & Anderson (1987)), for example by settling on *facebook* as a shorthand description for this object type. Finally, to understand a full scene description, the Matcher needs to combine all the evidence from multiple referential segments involving a group of objects to identify the target image.

In this chapter, we define and evaluate a language processing pipeline that allows many of these perceptual and language skills to be integrated into an automated system for understanding complex scene descriptions. We take the challenging visual reference game RDG-Pento, shown in Figure 7.1, as our testbed, and we evaluate both human-human and automated system performance in a corpus study.

Our automated pipeline, discussed in Section 7.5, includes components for learning perceptually grounded word meanings, segmenting a stream of speech, identifying the type of referential language in each speech segment, resolving the references in each type of segment, and aggregating evidence across segments to select the most likely target image. Our technical approach enables all of these components to be trained in a supervised manner from annotated, in-domain, human-human reference data. Our quantitative evaluation, presented in Section 7.6, looks at the performance of the individual components as well as the overall pipeline, and quantifies the strong importance of segmentation, segment type identification, and speaker-specific vocabulary entrainment for improving performance in this task.

7.2 Related Work

The work described in this chapter directly builds off of (Paetzel et al. (2015)) as the same RDG game scenario was used, however reference was only made to single objects in that work. The work here also builds off of (Kennington & Schlangen (2015)) in the same way in that their work only focused on reference to single objects. The extension of this previous work to handle more complex scene descriptions required substantial composition on the word and segment levels. The segmentation presented here was fairly straight forward (similar in spirit to chunking as in Marcus (1995)). Composition is currently an active area in distributional semantics where word meanings are represented by high-dimensional vectors and composition amounts to some kind of vector operation (see Milajevs et al. (2014) for a comparison of methods). An important difference is that here words and segments are composed at the denotational level (i.e., on the scores given by the WAC (Words-As-Classifiers) model, akin to *referentially afforded concept composition* (Mcnally & Boleda (2015))).

Also related are the recent efforts in automatic image captioning and retrieval, where the task is to generate a description (a caption) for a given image or retrieve one being given a description. A frequently taken approach is to use a convolutional neural network to map the image into a dense vector, and then to condition a neural language model on this to produce an output string or use it to map the description into the same space (Vinyals et al. (2015); Devlin et al. (2015); Socher et al. (2014)). See also (Fang et al. (2015)), which is more directly related to our model in that they use “word detectors” to propose words for image regions.

Han et al. (2015) directly relates to the work here in that the WAC was applied to similar geometric objects and images were retrieved from composed, individual referring expressions.

7.3 Words-As-Classifiers

7.4 The RDG-Pento Game

The RDG-Pento (Rapid Dialogue Game-Pentomino) game is a two player collaborative game. RDG-Pento is a variant of the RDG-Image game. As in RDG-Image, both players see 8 images on their screen in a 2X4 grid as shown in Figure 7.1. One person is assigned the role of Director and the other person that of Matcher. The Director’s screen has a single target image (TI) highlighted with a red border. The goal of the Director is to uniquely describe the TI for the Matcher to identify among the distractor images. The 8 images are shown in a different order on the Director and Matcher screens, so that the TI cannot be identified by grid position. The players can speak freely until the Matcher makes a selection. Once the Matcher indicates a selection, the Director can advance the game.

In RDG-Pento, the individual images are taken from a real-world, tabletop scene containing an arrangement of between one and six physical Pentomino objects. Individual images with varying numbers of objects are illustrated in Figure 7.2. The 8 images at any one time always contain the same number of objects; the number of objects increases as the game progresses. Players play for 5 rounds, alternating roles. Each round has a time limit (about 200 seconds) that creates time pressure for the players, and the time remaining ticks down in a countdown timer. For every correct guess made by the Matcher, both players win a micro-award of $0.02.

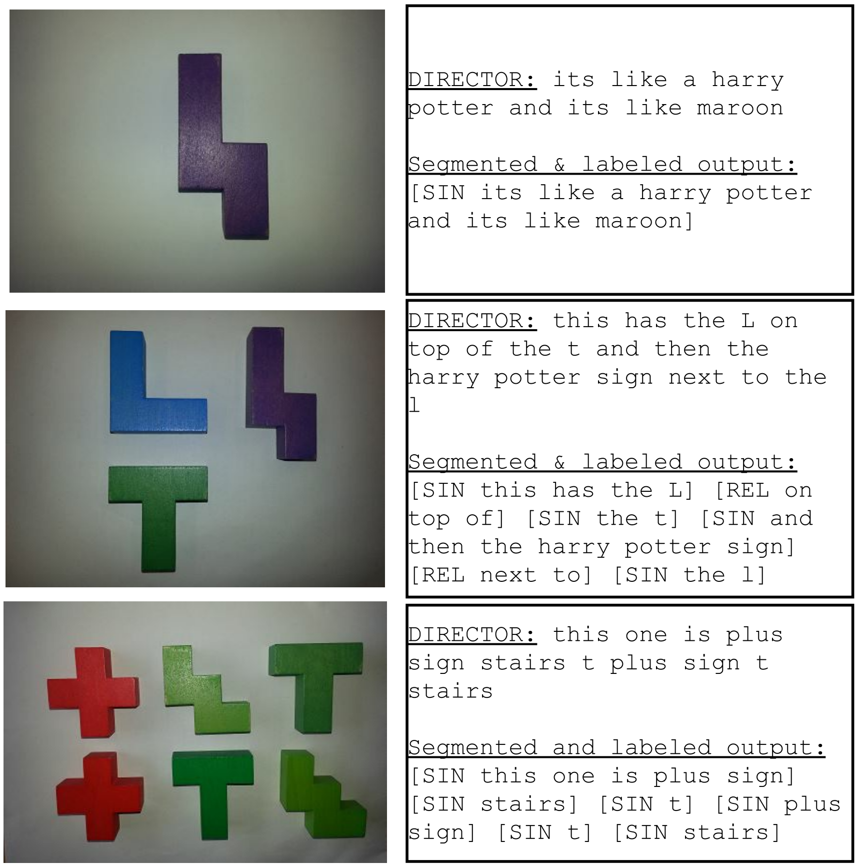
7.4.1 Data Collection

The corpus used here was collected using the Pair Me Up (PMU) web framework. To create this corpus, 42 pairs of native English-speakers located in the U.S. and Canada were recruited using Amazon Mechanical Turk. Game play and audio data were captured for each pair of speakers (who were not co-located and communicated entirely through their web browsers), and the resulting audio data was transcribed and annotated as described in section 7.4.2. 16 pairs completed all 5 game rounds, while the remaining crowd-sourced pairs completed only part of the game for various reasons. As our focus is on understanding individual scene descriptions, our data set here includes data from the 16 complete games as well as partial games. A more complete description and analysis of the corpus can be found in Zarrieß et al. (2016).

7.4.2 Data Annotation

We annotated the transcribed Director and Matcher speech through a process of segmentation, segment type labeling, and referent identification. The segment types are shown in Table 7.1, and example annotations are provided in Figure 7.2. The annotation is carried out on each *target image subdialogue* in which the Director and Matcher discuss an individual target image. The segmentation and labeling steps create a complete partition of each speaker's speech into sequences of words with a related semantic function in our framework.

Sequences of words that ascribe properties to a single object are joined under the SIN label. Our SIN segment type is not a simple syntactic concept like "singular



DIRECTOR: its like a harry potter and its like maroon

Segmented & labeled output:
[SIN its like a harry potter and its like maroon]

DIRECTOR: this has the L on top of the t and then the harry potter sign next to the l

Segmented & labeled output:
[SIN this has the L] [REL on top of] [SIN the t] [SIN and then the harry potter sign] [REL next to] [SIN the l]

DIRECTOR: this one is plus sign stairs t plus sign t stairs

Segmented and labeled output:
[SIN this one is plus sign] [SIN stairs] [SIN t] [SIN plus sign] [SIN t] [SIN stairs]

Figure 7.2: Example scene descriptions for three TIs.

NP referring expression". The SIN type includes not only simple singular NPs like *the blue s* but also clauses like *it's the blue s* and conjoined clauses like *its like a harry potter and its like maroon* (Figure 7.2). The individuation criterion for SIN is that a SIN segment must ascribe properties only to a single object; as such it may contain word sequences of various syntactic types.

Sequences of words such as *the two crosses* that ascribe properties to multiple objects are joined into a segment under the MUL label.

Sequences of words that describe a geometric relation between objects are segmented and given a REL label. These are generally prepositional expressions,

Segment type	**Label**	**Examples**
Singular	SIN	this is a green t, plus sign
Multiple objects	MUL	two Zs at top, they're all green
Relation	REL	above, in a diagonal
Others	OT	that was tough, lets start

Table 7.1: Segment types, labels, and examples.

and include both single-word prepositions (*underneath*, *below*) and multi-word complex prepositions (Quirk et al. (1985)) which include multiple orthographic words ("next to", "left of", "in front of", etc.). The REL segments generally describe geometric relations between objects referred to in SIN and MUL segments. An example would be *[MUL two crosses] [REL above] [MUL two Ts].*

All other word sequences are assigned the type Others and given an OT label. This segment type includes acknowledgments, confirmations, feedback, and laughter, among other dialogue act types not addressed in this work.

For each segment of type SIN, MUL, or REL, the correct referent object or objects within the target image are also annotated. This annotation scheme was designed to work well with the model for word-level understanding, discussed further in Section 7.5.3.

In the dataset, there are a total of 4132 *target image speaker transcripts* in which either the Director's or the Matcher's transcribed speech for a target image is annotated. There are 8030 annotated segments (5451 Director segments and 2579 Matcher segments). There are 1372 word types and 55,238 word tokens.

7.5 Language Processing Pipeline

In this section, we present our language processing pipeline for segmentation and understanding of complex scene descriptions. The modules, decision-making, and

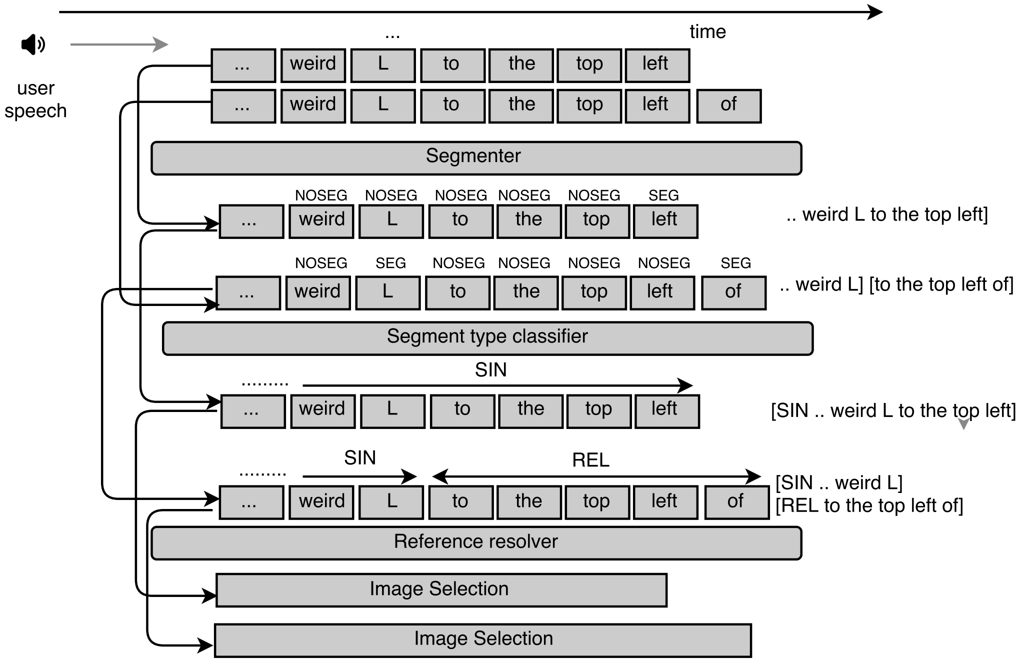

time
...
user
speech
... weird L to the top left
... weird L to the top left of
Segmenter
NOSEG NOSEG NOSEG NOSEG NOSEG SEG
.. weird L to the top left]
NOSEG SEG NOSEG NOSEG NOSEG NOSEG SEG
.. weird L] [to the top left of]
Segment type classifier
SIN
[SIN .. weird L to the top left]
SIN
REL
[SIN .. weird L]
[REL to the top left of]
Reference resolver
Image Selection
Image Selection

Figure 7.3: Information flow during processing of an utterance. The modules operate incrementally, word-by-word; as shown here, this can lead to revisions of decisions.

information flow for the pipeline are visualized in Figure 7.3. The pipeline modules include a Segmenter (Section 7.5.1), a Segment Type Classifier (Section 7.5.2), and a Reference Resolver (Section 7.5.3). In the coming subsections, we describe each of these modules, while the evaluation of each module and of the full pipeline is presented in Section 7.6.

In this chapter, we focus on how our pipeline could be used to automate the role of the Matcher in the RDG-Pento game. We consider the task of selecting the correct target image based on a human Director's transcribed speech drawn from our RDG-Pento corpus.

The pipeline is designed however for eventual real-time operation using incremental automatic speech recognition results, so that in the future it can be incorporated into a real-time interactive dialogue system. We view it as a crucial design constraint on our pipeline modules that the resolution process must take place *incrementally*; i.e., processing must not be deferred until the end of the user's speech arrives. This is because humans resolve (i.e., comprehend) speech as it unfolds, and incremental processing (i.e., processing word by word) is important to developing an efficient and natural speech channel for interactive systems (Skantze & Schlangen (2009); Paetzel et al. (2015); DeVault et al. (2009); Aist et al. (2007b)). In the current study, we have therefore provided the human Director's correctly transcribed speech as input to our pipeline on a word-by-word basis, as visualized in Figure 7.3. Our results in Section 7.6 evaluate the performance of the pipeline at the end of the Director's speech for each target image.

7.5.1 Segmenter

The segmenter module is tasked with identifying the boundary points between segments. In our pipeline, this task is performed independently of the determination of segment types, which is handled by a separate classifier (Section 7.5.2).

Our approach to segmentation is similar to (Celikyilmaz et al. (2014)) which used CRFs for a similar task. Our pipeline currently uses linear-chain CRFs to find the segment boundaries, implemented with Mallet (McCallum (2002)). Using a CRF trained on the annotated RDG-Pento data set, we identify the most likely sequence of word-level boundary tags, where each tag indicates if the current word ends the previous segment or not.[1] An example segmentation is shown in Figure 7.3, where the word sequence *weird L to the top left of* is segmented into two segments, *[weird L]* and *[to the top left of]*.

[1] We currently adopt this two-tag approach rather than BIO tagging as our tag-set provides a complete partition of each speaker's speech.

The features provided to the CRF include unigrams[2], the speaker's role, part-of-speech (POS) tags obtained using the Stanford POS tagger (Toutanova & Manning (2000)), and information about the scene such as the number of objects.

7.5.2 Segment Type Classifier

The segment type classifier assigns each detected segment with one of the type labels in Table 7.1 (SIN, MUL, REL, OT). This label informs the Reference Resolver module in how to proceed with the resolution process, as explained below.

The segment type labeler is an SVM classifier implemented in LIBSVM. Features used include word unigrams, word POS, user role, number of objects in the TI, and the top-level syntactic category of the segment as obtained from the Stanford parser (Manning et al. (2014)). Figure 7.3 shows two examples of output from the segment type classifier, which assigns SIN to *[weird L]* and REL to *[to the top left of]*.

7.5.3 Reference Resolver

We introduce some notation to help explain the operation of the reference resolver (RR) module. When a scene description is to be resolved, there is a visual context in the game which we encode as a context set $\mathcal{C} = I_1, ..., I_8$ containing the eight visible images (see Figure 7.1). Each image I_k contains n objects $\{o_1^k, \dots, o_n^k\}$, where n is fixed per context set, but varies across context sets from $n = 1$ to $n = 6$. The set of all objects in all images is $\mathcal{O} = \{o_l^k\}$, with $0 < k \leq 8, 0 < l \leq n$.

When the RR is invoked, the Director has spoken some sequence of words which has been segmented by earlier modules into one or more segments $S_j =$

[2] Words of low frequency (i.e., <5) are replaced with a fixed symbol.

$w_{1:m_j}$, and where each segment has been assigned a segment type $\text{type}(S_j) \in \{\text{SIN}, \text{MUL}, \text{REL}, \text{OT}\}$. For example, $S_1 = \langle \text{weird}, \text{L} \rangle, S_2 = \langle \text{to}, \text{the}, \text{top}, \text{left}, \text{of} \rangle$ and $\text{type}(S_1) = \text{SIN}, \text{type}(S_2) = \text{REL}$.

The RR then tries to understand the individual words, typed segments, and the full scene description in terms of the visible objects o_l^k and the images I_k in the context set. We describe how words, segments, and scene descriptions are understood in the following three sections.

Understanding words

We understand individual words using the Words-as-Classifiers (WAC) model of Kennington & Schlangen (2015). In this model, a classifier is trained for each word w_p in the vocabulary. The model constructs a function from the perceptual features of a given object to a judgment about how well those features "fit" together with the word being understood. Such a function can be learned using a logistic regression classifier, separately for each word.

The inputs to the classifier are the low-level continuous features that represent the object (e.g., RGB/HSV values, number of detected edges, x/y coordinates, etc.) extracted using OpenCV.[3] These classifiers are learned from instances of language use, i.e., by observing referring expressions paired with the object referred to. Crucially, once learned, these word classifiers can be applied to any number of objects in a scene.

We trained a WAC model for each of the (non-relational) words in our RDG-Pento corpus, using the annotated correct referent information for our segmented data (Section 7.4.2).

After training, words can be applied to objects to yield a score:

[3] `http://opencv.org`

$$score(w_p, o_l^k) = w_p(o_l^k) \tag{7.1}$$

(Technically, the score is the response of the classifier associated with word w_p applied to the feature representation of object o_l^k.).

Note that relational expressions are trained slightly differently than non-relational words. Examples of relational expressions include *underneath*, *below*, *next to*, *left of*, *right of*, *above*, and *diagonal*. A WAC classifier is trained for each full relational expression e_q (treated as a single token), and the 'fit' for a relational expression's classifier is a fit for a *pair* of objects:[4]

$$score_{rel}(e_q, o_{l_1}^k, o_{l_2}^k) = e_q(o_{l_1}^k, o_{l_2}^k) \tag{7.2}$$

There are about 300 of these expressions in RDG-Pento. See (Kennington & Schlangen (2015)) for details on this training.

Understanding segments

Consider an arbitrary segment $S_j = w_{1:m_j}$ such as $S_1 = \langle \text{weird}, \text{L} \rangle$. For a segment (SIN or MUL), we attempt to understand the segment as referring to some object or set of objects. To do so, we combine the word-level scores for all the words in the segment to yield a segment-level score[5] for each object o_l^k:

[4] The features used for such a classifier are comparative features, such as the euclidean distance between the two objects, as well as x and y distances.

[5] The composition operator $\odot$ is left-associative and hence incremental. In this chapter, word-level scores are composed by multiplying them.

$$\begin{aligned} score(S_j, o_l^k) = score(w_1, o_l^k) \odot \ \dots \odot \\ score(w_{m_j}, o_l^k) \end{aligned} \tag{7.3}$$

Each segment $S_j = w_{1:m_j}$ hence induces an order R_j on the object set $\mathcal{O}$, through the scores assigned to each object o_l^k. With these ranked scores, we look at the type of segment to compute a final score $score_k^*(S_j)$ for each image I_k. For SIN segments, $score_k^*(S_j)$ is the score of the top-scoring object in I_k. For MUL segments with a cardinality of two (e.g., *two red crosses*), $score_k^*(S_j)$ is the sum of the scores of the top two objects in I_k, and so on.

Obtaining the final score $score_k^*(S_j)$ for REL segments is done in a similar manner with some minor differences. Because REL segments express a relation between *pairs* of objects (referred to in neighboring segments), a score for the relational expression in S_j can be computed for any pair of distinct objects $o_{l_1}^k$ and $o_{l_2}^k$ in image I_k using Eq. (7.2). We let $score_k^*(S_j)$ equal the score computed for the top-scoring objects $o_{l_1}^k$ and $o_{l_2}^k$ of the neighboring segments.

Understanding scene descriptions

In general, a scene description consists of segments $S_1, ..., S_z$. Composition takes segments $S_1, ..., S_z$ and produces a ranking over images. For this particular task, we make the following assumption: in each segment, the speaker is attempting to refer to a specific object (or set of objects), which from our perspective as matcher could be in any of the images. A good candidate I_k for the target image will have

Label	Precision	Recall	F-Score
SEG	0.85	0.74	0.79
NOSEG	0.93	0.97	0.95

Table 7.2: Segmenter performance.

high scoring objects, all drawn from the same image, for all the segments $S_1, ..., S_z$. We therefore obtain a final score for each image in the following manner:[6]

$$score(I_k) = \sum_{j=1}^{z} score_k^*(S_j) \tag{7.4}$$

The image I_k^* selected by our pipeline for a full scene description is then given by:

$$I_k^* = \operatorname*{argmax}_k score(I_k) \tag{7.5}$$

7.6 Experiments & Evaluations

We first evaluate the segmenter and segment type classifier as individual modules. We then evaluate the entire processing pipeline and explore the impact of several factors on pipeline performance.

7.6.1 Segmenter Evaluation

Task & Data. We used the annotated RDG-Pento data to perform a hold-one-dialogue-pair-out cross-validation of the segmenter. The task is to segment each

[6] This is a fairly naive approach to composing the segments; we leave more complex approaches, for example using transition models to capture regular patterns of reference across segments, such as describing objects in a left-to-right and top-to-bottom order, for future work.

Label	Precision	Recall	F-score	% of segments
SIN	0.91	0.96	0.93	57
REL	0.97	0.85	0.91	6
MUL	0.86	0.60	0.71	3
OT	0.96	0.97	0.96	34

Table 7.3: Segment type classifier performance.

speaker's speech for each target image by tagging each word using the tags SEG and NOSEG. The SEG tag here indicates the last word in the current segment. Figure 7.3 gives an example of the tagging.

Results. The results are presented in Table 7.2. These results show that the segmenter is working with some success, with precision 0.85 and recall 0.74 for the SEG tag indicating a word boundary. Note that occasional errors in segment boundaries may not be overly problematic for the overall pipeline, as what we ultimately care most about is accurate target image selection. We evaluate the overall pipeline below (section 7.6.3).

7.6.2 Segment Type Classifier Evaluation

Task & Data. We used the annotated RDG-Pento data to perform a hold-one-pair-out cross-validation of the segment type classifier, training a SVM classifier to predict labels SIN, MUL, REL, and OT using the features described in section 7.5.2.

Results. The results are given in Table 7.3. We also report the percentage of segments that have each label in the corpus. The segment type classifier performs well on most of the class labels. Of slight concern is the low-frequency MUL label. One factor here is that people use number words like *two* not just to refer to multiple objects, but also to describe individual objects, e.g., *the two red crosses* (a

MUL segment) vs. *the one with two sides* (a SIN segment). Our current feature set does not distinguish these two cases very well.

7.6.3 Pipeline Evaluation

We evaluated our pipeline under varied conditions to understand how well it works when segmentation is not performed at all, when the segmentation and type classifier modules produce perfect output (using oracle annotations), and when entrainment to a specific speaker is possible.

Task. We evaluate our pipeline on the task of image retrieval: given a scene description from our data set, how accurately does the pipeline select the correct target image?

Three baselines. We compare against a weak random baseline (1/8 = 0.125) as well as a rather strong one, namely the accuracies of the human-human pairs in the RDG-Pento corpus. As Table 7.4 shows, in the simplest case, with only one object per image, the average human success rate is 85%, but this decreases to 60% when there are four objects/image. It then increases to 68% when 6 objects are present, possibly due to the use of a more structured description ordering in the six object scenes. We leave further analysis of the human strategies for future work. These numbers show that the game is challenging for humans.

We also include in Table 7.4 a simple Naive Bayes classification approach as an alternative to our entire pipeline. In our study, there were only 40 possible image sets that were fixed in advance. For each possible image set, a different Naive Bayes classifier is trained using Weka (Hall et al. (2009)) in a hold-one-pair-out cross-validation. The eight images are treated as atomic classes to be predicted, and unigram features drawn from the union of all (unsegmented) Director speech are used to predict the target image. This method is broadly comparable to the NLU model used in (Paetzel et al. (2015)) to achieve high performance in resolving references to pictures of single objects. As can be seen, the accuracy for this method is as high as 43% for single object TIs in the RDG-Pento data set, but the accuracy rapidly falls to near the random baseline as the number of objects/image increases. This weak performance for a classifier without segmentation confirms the importance of segmenting complex descriptions into references to individual objects in the RDG-Pento game.

Five versions of the pipeline. Table 7.4 includes results for 5 versions of our pipeline. The versions differ in terms of which segment boundaries and segment type labels are used, and in the type of cross-validation performed. A first version (I) explores how well the pipeline works if unsegmented scene descriptions are provided and a SIN label is assumed to cover the entire scene description. This model is broadly comparable to the Naive Bayes baseline, but substitutes a WAC-based NLU component. The evaluation of version (I) uses a hold-one-pair-out (HOPO) cross-validation, where all modules are trained on every pair except for the one being used for testing. A second version (II) uses automatically determined segment boundaries and segment type labels, in a HOPO cross-validation, and represents our pipeline as described in Section 7.5. A third version (III) substitutes in human-annotated or "oracle" segment boundaries and type labels, allowing us to observe the performance loss associated with imperfect segmentation and type labeling in our pipeline. The fourth and fifth versions of the pipeline switch to a hold-one-episode-out (HOEO) cross-validation, where only the specific scene description ("episode") being tested is held out from training. When compared with a HOPO cross-validation, the HOEO setup allows us to investigate the value

		#objects per TI				
		1	2	3	4	6
Random baseline		0.13	0.13	0.13	0.13	0.13
Naive Bayes baseline		0.43	0.20	0.14	0.14	0.13
Seg+lab	X-validation					
(I) None	HOPO	0.47	0.20	0.24	0.13	0.15
(II) Auto	HOPO	0.52	0.40	0.31	0.24	0.23
(III) Oracle	HOPO	0.54	0.42	0.32	0.30	0.26
(IV) Auto	HOEO	0.60	0.46	0.37	0.25	0.23
(V) Oracle	HOEO	0.64	0.50	0.41	0.34	0.44
Human-human baseline		0.85	0.73	0.66	0.60	0.68

Table 7.4: Image retrieval accuracies for five versions of the pipeline and three baselines.

of learning from and entraining to the specific speaker's vocabulary and speech patterns (such as calling the purple object in Figure 7.2 a "harry potter").

Results. Table 7.4 summarizes the image retrieval accuracies for our three baselines and five versions of our pipeline. We discuss here some observations from these results. First, in comparing pipeline versions (I) and (II), we observe that the use of automated segmentation and a segment type classifier in (II) leads to a substantial increase in accuracy of 5-20% depending on the number of objects/image. Comparing (II) and (III), we see that if our segmenter and segment type classifier could reproduce the human segment annotations perfectly, an additional improvement of 1-6% accuracy would be possible. Comparing (II) to (IV), we see that exposing our pipeline training to the idiosyncratic speech and vocabulary of a given speaker would hypothetically enable an increase in accuracy of up to 8%. Note however that this setup cannot easily be replicated in a real-time system, as our HOEO training provides not only samples of the transcribed speech of the same speaker, but also human annotations of the segment boundaries, segment types, and correct referents for this speech (which would not generally be available for immediate use in a run-time system). Comparing (IV) to (V), we see that oracle segment boundaries and types also improve accuracies in a HOEO evaluation between 4-19%.

Comparing our fully automated HOPO pipeline (II) to the baselines, we see that our pipeline performs considerably better than the random and Naive Bayes baselines. At the same time, there is still much room for improvement when we compare to human-human accuracy.

7.6.4 Evaluation of Object Retrieval

Table 7.4 shows that even when there is just one object in each of the eight images, our pipeline (II) only selects the correct image 52% of the time given the complete scene description, while humans succeed 85% of the time. We further investigated our performance at understanding descriptions of individual objects by defining a constructed “object retrieval” problem. In this problem, individual SIN segments from the RDG-Pento corpus are considered one at a time, and the correct target image is provided by an oracle. The only task is to use the WAC model to select the correct referent object within the image for a single SIN segment. An example of the object retrieval problem is to select the correct referent for the SIN segment *and a wooden w sort of* in the known target image of Figure 7.1.

The results are shown in Table 7.5. We can observe that object retrieval is by itself a non-trivial problem for our WAC model, especially as the number of objects increases. This is somewhat by design in that the multiple objects present within an image are often selected to be fairly similar in their properties, and multiple objects may match ambiguous SIN segments such as *the T* or *the plus sign.* We speculate that we could gain here from factoring in positional information implicit in description strategies such as going from top left to bottom right in describing the objects.

n	1	2	3	4	6
accuracy	1	.88	.77	.60	.66

Table 7.5: Accuracy for object retrieval in target images with n objects.

7.7 Contributions

We have presented an approach to understanding complex, multi-utterance references to images containing spatially complex scenes. The approach by design works incrementally, and hence is ready to be used in an interactive system. We presented evaluations that go end-to-end from utterance input to resolution decision (but not yet taking in speech). We have shown that segmentation is a critical component for understanding complex visual scene descriptions.

This work opens avenues for future explorations in various directions. Intra- and inter-segment composition (through multiplication and addition, respectively) are approached somewhat simplistically, and we want to explore the consequences of these decisions more deeply in future work. Additionally, as discussed above, there seems to be much implicit information in how speakers go from one reference to the next, which might be possible to capture in a transition model. Finally, in an online setting, there is more than just the decision "this is the referent"; one must also decide when and how to act based on the confidence in the resolution. Lastly, our results have shown that human pairs do align on their conceptual description frames (Garrod & Anderson (1987)). Whether human users would also do this with an artificial interlocutor, if it were able to do the required kind of online learning, is another exciting question for future work, enabled by the work presented here. This discussions covered in this chapter have been published in (Manuvinakurike et al. (2016); Zarrieß et al. (2016)).

Chapter 8

Contributions & Future Work

> *"If you're not failing, you're not pushing your limits, and if you're not pushing your limits, you're not maximizing your potential"*
>
> – Ray Dalio, *Principles: Life and Work*

8.1 Contributions

In this chapter, I summarize the contributions made in this work. In Chapter 3, a framework for crowd-sourced spoken dialogue data collection is developed in support of incremental SDS development. We additionally show the advantages and trade-offs of collecting a crowd-sourced dataset by comparing with an in-lab dataset. This is the first such comprehensive comparison in the literature. The framework also supports the development of an incremental SDS. To our knowledge, this is the only framework in the literature that supports data collection for building incremental dialogue systems.

In Chapter 4, we develop an incremental SDS which performs about as well as humans in scoring points. This is the first such incremental SDS that was developed to operate on a web platform and that achieves such high performance when compared to humans. We then extend this work in Chapter 5, by utilizing Reinforcement Learning (RL) for the development of an incremental dialogue policy. Such a policy also learns automatically to score about as well as humans and achieves a better qualitative performance than the previously developed agent (in

Chapter 4). This comparison of the RL approach with a strong baseline is the first such study in the literature showing the advantages and limitations of the RL approach. We also find that there is no significant difference in terms of perceived game efficiency when we compare our RL agent with human behavior. We later extend this work by showing that transfer learning approaches can help learn better dialogue policies with fewer samples. This is the first time in the literature that transfer learning is used for building incremental dialogue policies.

In Chapter 6, we then extend this work by including incremental segmentation and Dialogue Act (DA) understanding which improves the semantic processing capabilities in SDS. We show the advantages of incremental segmentation and DA identification in the SDS pipeline in a real-world application that helps achieve substantial savings in processing time in the task of user intent detection. We extend this work further in Chapter 7 and show that such incremental processing capabilities can be achieved with visually grounded models. Figure 8.1 shows the summary of this work in a flow diagram.

8.2 Future Work

In this section, I discuss possible future directions for this work.

Incremental dialogue processing is an approach that is expected to find wide applications in the future. It will find use in problems that involve time critical feedback such as autonomous self-driving cars. A preliminary analysis has begun in Karkada et al. (2018) where we collect and annotate data for language understanding. Similarly, the work on incremental segmentation and DA labeling will be useful in health care intervention domains. We have begun preliminary work in this area in

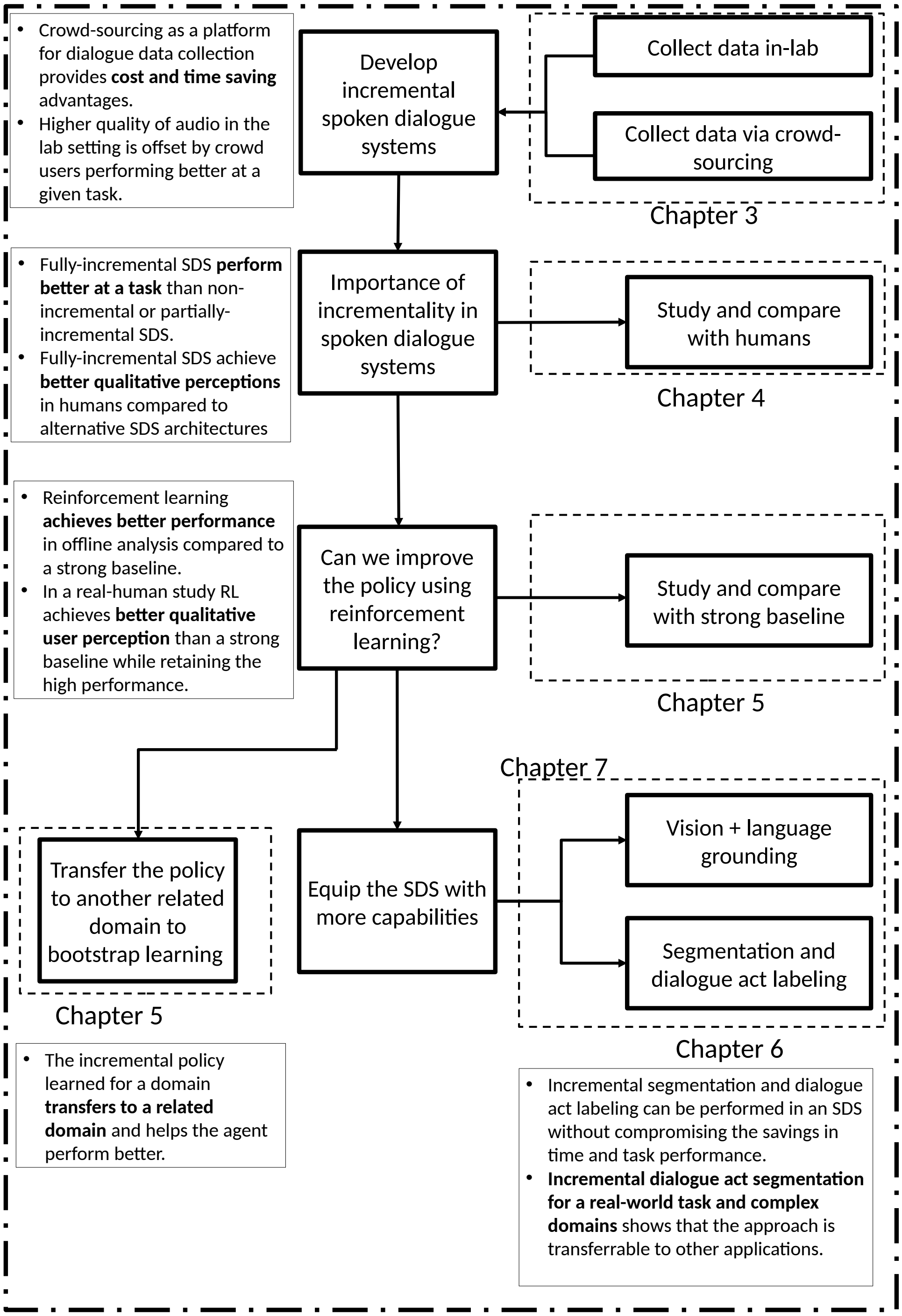
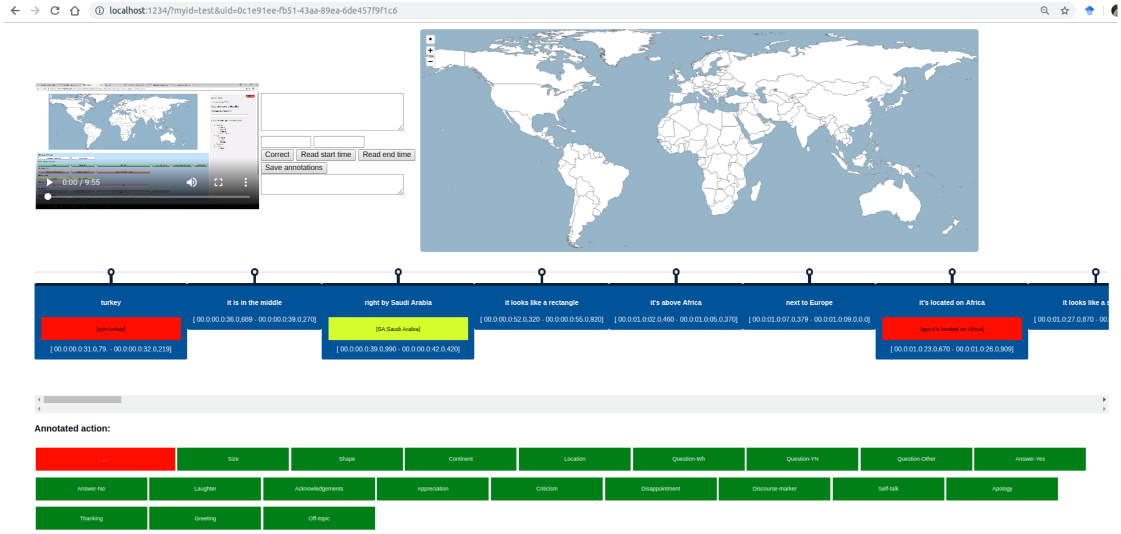

Crowd-sourcing as a platform for dialogue data collection provides cost and time saving advantages.
Higher quality of audio in the lab setting is offset by crowd users performing better at a given task.
Develop incremental spoken dialogue systems
Collect data in-lab
Collect data via crowd-sourcing
Chapter 3
Fully-incremental SDS perform better at a task than non-incremental or partially-incremental SDS.
Fully-incremental SDS achieve better qualitative perceptions in humans compared to alternative SDS architectures
Importance of incrementality in spoken dialogue systems
Study and compare with humans
Chapter 4
Reinforcement learning achieves better performance in offline analysis compared to a strong baseline.
In a real-human study RL achieves better qualitative user perception than a strong baseline while retaining the high performance.
Can we improve the policy using reinforcement learning?
Study and compare with strong baseline
Chapter 5
Chapter 7
Transfer the policy to another related domain to bootstrap learning
Equip the SDS with more capabilities
Vision + language grounding
Segmentation and dialogue act labeling
Chapter 5
Chapter 6
The incremental policy learned for a domain transfers to a related domain and helps the agent perform better.
Incremental segmentation and dialogue act labeling can be performed in an SDS without compromising the savings in time and task performance.
Incremental dialogue act segmentation for a real-world task and complex domains shows that the approach is transferrable to other applications.

Figure 8.1: The figure shows the contributions of this work.

Manuvinakurike et al. (2014, 2018) and will continue further work in this area in the future.

As we have seen in this work, data collection and preparation is an important step prior to the development of an SDS. The steps that are typically involved in the development of an SDS are, i) human data collection, ii) data transcription, iii) data annotation, iv) agent development and deployment. In this work, we presented a framework for collecting data in a human-human setting using a crowd-sourcing approach. Collecting data in a human-wizard setting is also popular for data collection. Our framework has been adopted for collecting data in a wizard-based setting. The user is paired with a wizard who controls the responses to the user utterances. One of the limitations of this framework is the long wait times for pairing the users. While this is due to the fluctuating number of users present throughout the day, the whole process can be better streamlined with efficient load balancing. In future work, we aim to streamline the process of crowd-sourced data collection in this way.

The next step in the process of data preparation is transcription and annotation. The transcription process can be automated with accurate ASR, but the transcripts still need to be corrected. Once the audio is transcribed, it needs to be annotated. Currently, the dialogue data is annotated by experts as it requires an understanding of the annotation scheme and linguistic knowledge. While crowd-sourcing the annotation process enables the collection of large-scale dialogue annotation data, these challenges make the task of crowd-sourcing annotation challenging. In the future, it will be interesting to explore these directions. Another fruitful direction will be to leverage existing dialogue resources to aid the process of SDS building and annotation. We make use of these principles to build a preliminary transfer learning based annotation toolkit, as shown in Figure 8.2.



localhost:1234/?myid=test&uid=0c1e91ee-fb51-43aa-89ea-6de457f9f1c6

turkey
[qyn:turkey]
[00.0:00.0:31.0,79. - 00.0:00.0:32.0,219]

it is in the middle
[00.0:00.0:36.0,689 - 00.0:00.0:39.0,270]

right by Saudi Arabia
[SA:Saudi Arabia]
[00.0:00.0:39.0,990 - 00.0:00.0:42.0,420]

it looks like a rectangle
[00.0:00.0:52.0,320 - 00.0:00.0:55.0,920]

it's above Africa
[00.0:01.0:02.0,460 - 00.0:01.0:05.0,370]

next to Europe
[00.0:01.0:07.0,379 - 00.0:01.0:09.0,0.0]

it's located on Africa
[qyn:it's located on Africa]
[00.0:01.0:23.0,670 - 00.0:01.0:26.0,909]

it looks like a s
[00.0:01.0:27.0,870 - 00.

Annotated action:

Figure 8.2: The figure shows the semi-automated transcription correction and annotation interface.

Another direction that will be interesting to follow is extending the models of incrementality to other problems which will benefit from such models. Extending the segmentation and dialogue act understanding models using deep learning, which has been shown to yield high accuracy, is another fruitful direction. Zhao & Kawahara (2019) have explored such a direction by building deep learning models for segmentation and dialogue act understanding in a non-incremental setting with a limited set of dialogue acts. It will be interesting to extend such work in the future for incremental processing with more data collected using the methods presented.

For building dialogue systems that are multimodal in nature, models that include vision and language inputs are an exciting direction to pursue in the future. Also, an interesting direction is end-to-end incremental SDS, which eliminates the need for building separate models. However, such models need further exploration.

End

Reference List

3rd Generation Partnership Project (2008) Technical specification group services and system aspects; ansi-c code for the adaptive multi rate (amr) speech codec (release 12).

Aist G, Allen J, Campana E, Gallo CG (2007a) Incremental understanding in human-computer dialogue and experimental evidence for advantages over nonincremental methods. *Decalog 2007* p. 149.

Aist G, Allen J, Campana E, Gallo CG, Stoness S, Swift M, Tanenhaus MK (2007b) Incremental dialogue system faster than and preferred to its nonincremental counterpart In *the 29th Annual Conference of the Cognitive Science Society.*

Allen J, Ferguson G, Stent A (2001) An architecture for more realistic conversational systems In *Proceedings of the 6th international conference on Intelligent user interfaces*, pp. 1–8. ACM.

Amodei D, Ananthanarayanan S, Anubhai R, Bai J, Battenberg E, Case C, Casper J, Catanzaro B, Cheng Q, Chen G et al. (2016) Deep speech 2: End-to-end speech recognition in english and mandarin In *International Conference on Machine Learning*, pp. 173–182.

Anderson AH, Bader M, Bard EG, Boyle E, Doherty G, Garrod S, Isard S, Kowtko J, McAllister J, Miller J et al. (1991) The HCRC map task corpus. *Language and Speech* 34:351–366.

Ang J, Liu Y, Shriberg E (2005) Automatic dialog act segmentation and classification in multiparty meetings. In *ICASSP*, pp. 1061–1064.

Antol S, Agrawal A, Lu J, Mitchell M, Batra D, Lawrence Zitnick C, Parikh D (2015) Vqa: Visual question answering In *Proceedings of the IEEE international conference on computer vision*, pp. 2425–2433.

Atterer M, Baumann T, Schlangen D (2008) Towards incremental end-of-utterance detection in dialogue systems In *Proceedings of the 22nd International Conference on Computational Linguistics.*

Baumann T, Kennington C, Hough J, Schlangen D (2017) Recognising conversational speech: What an incremental asr should do for a dialogue system and how to get there In *Dialogues with Social Robots*, pp. 421–432. Springer.

Baumann T, Schlangen D (2012) The inprotk 2012 release In *NAACL-HLT Workshop on Future Directions and Needs in the Spoken Dialog Community: Tools and Data*, pp. 29–32.

Baumann T, Schlangen D (2013) Open-ended, extensible system utterances are preferred, even if they require filled pauses In *SigDial*, pp. 280–283.

Bell L, Boye J, Gustafson J (2001) Real-time handling of fragmented utterances In *The NAACL Workshop on Adaption in Dialogue Systems*, pp. 2–8.

Bickmore TW, Caruso L, Clough-Gorr K, Heeren T (2005) âĂŸitâĂŹs just like you talk to a friendâĂŹrelational agents for older adults. *Interacting with Computers* 17:711–735.

Bojanowski P, Grave E, Joulin A, Mikolov T (2016) fasttext.

Brixey J, Manuvinakurike R, Le N, Lai T, Chang W, Bui T (2018) A system for automated image editing from natural language commands. *arXiv preprint arXiv:1812.01083* .

Bunt H (2009) The dit++ taxonomy for functional dialogue markup In *AAMAS 2009 Workshop, Towards a Standard Markup Language for Embodied Dialogue Acts*, pp. 13–24.

Bunt H, Alexandersson J, Carletta J, Choe JW, Fang AC, Hasida K, Lee K, Petukhova V, Popescu-Belis A, Romary L et al. (2010) Towards an iso standard for dialogue act annotation In *Seventh conference on International Language Resources and Evaluation (LREC'10)*.

Buß O, Schlangen D (2010) Modelling sub-utterance phenomena in spoken dialogue systems In *Proceedings of the 14th International Workshop on the Semantics and Pragmatics of Dialogue (Pozdial 2010)*.

Byron DK, Fosler-Lussier E (2006) The OSU Quake 2004 corpus of two-party situated problem-solving dialogs In *Proceedings of LREC*, pp. 395–400, Genoa, Italy.

Celikyilmaz A, Feizollahi Z, Hakkani-Tur D, Sarikaya R (2014) Resolving referring expressions in conversational dialogs for natural user interfaces In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2094–2104.

Chotimongkol A, Rudnicky AI (2001) N-best speech hypotheses reordering using linear regression .

Clark HH, Wilkes-Gibbs D (1986) Referring as a collaborative process. *Cognition* 22:1–39.

Core MG, Allen JF (1997) Coding dialogues with the DAMSL annotation scheme In *AAAI Fall Symposium on Communicative Action in Humans and Machines*, pp. 28–35.

Das A, Datta S, Gkioxari G, Lee S, Parikh D, Batra D (2018) Embodied question answering In *Proceedings of CVPR*, Salt Lake City, Utah, USA.

Das A, Kottur S, Gupta K, Singh A, Yadav D, Moura JMF, Parikh D, Batra D (2017) Visual dialog In *Proceedings of CVPR*, pp. 326–335, Honolulu, Hawaii, USA.

de Kok I, Hough J, Hülsmann F, Botsch M, Schlangen D, Kopp S (2015) A multimodal system for real-time action instruction in motor skill learning In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, pp. 355–362. ACM.

de Vries H, Strub F, Chandar S, Pietquin O, Larochelle H, Courville A (2017) GuessWhat?! visual object discovery through multi-modal dialogue In *Proceedings of CVPR*, pp. 5503–5512, Honolulu, Hawaii, USA.

Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: A large-scale hierarchical image database .

Dethlefs N, Hastie H, Cuayáhuitl H, Yu Y, Rieser V, Lemon O (2016) Information density and overlap in spoken dialogue. *Computer Speech & Language* 37:82–97.

Dethlefs N, Hastie H, Riser V, Lemon O (2012) Optimising incremental generation for spoken dialogue systems: Reducing the need for fillers In *Seventh International Natural Language Generation Conference (INLG).*

DeVault D, Artstein R, Benn G, Dey T, Fast E, Gainer A, Georgila K, Gratch J, Hartholt A, Lhommet M et al. (2014) Simsensei kiosk: A virtual human interviewer for healthcare decision support In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pp. 1061–1068. International Foundation for Autonomous Agents and Multiagent Systems.

DeVault D, Sagae K, Traum D (2009) Can i finish?: learning when to respond to incremental interpretation results in interactive dialogue In *Proceedings of the SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest*

Group on Discourse and Dialogue, pp. 11–20. Association for Computational Linguistics.

DeVault D, Sagae K, Traum D (2011a) Detecting the status of a predictive incremental speech understanding model for real-time decision-making in a spoken dialogue system In *Twelfth Annual Conference of the International Speech Communication Association.*

DeVault D, Sagae K, Traum D (2011b) Incremental interpretation and prediction of utterance meaning for interactive dialogue. *Dialogue and Discourse* 2:143–70.

DeVault D, Stone M (2003) Domain inference in incremental interpretation In *Proc. ICoS*, pp. 73–87. Citeseer.

DeVault D, Stone M (2009) Learning to interpret utterances using dialogue history In *Proceedings of EACL*, pp. 184–192, Athens, Greece.

DeVault D, Traum D (2012) Incremental speech understanding in a multi-party virtual human dialogue system In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstration Session*, pp. 25–28. Association for Computational Linguistics.

Devlin J, Cheng H, Fang H, Gupta S, Deng L, He X, Zweig G, Mitchell M (2015) Language models for image captioning: The quirks and what works In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 100–105, Beijing, China. Association for Computational Linguistics.

Di Eugenio B, Jordan PW, Thomason RH, Moore JD (2000) The agreement process: An empirical investigation of human–human computer-mediated collaborative dialogs. *International Journal of Human-Computer Studies* 53:1017–1076.

Eshghi A, Lemon O (2014) How domain-general can we be? learning incremental dialogue systems without dialogue acts. *Proceedings of SemDial* .

Fang H, Gupta S, Iandola F, Srivastava R, Deng L, Dollar P, Gao J, He X, Mitchell M, Platt J, Zitnick L, Zweig G (2015) From captions to visual concepts and back In *Proceedings of CVPR*, Boston, MA, USA. IEEE.

Fernandez R, Picard RW (2002) Dialog act classification from prosodic features using support vector machines In *Speech Prosody 2002, International Conference.*

Ferrer L, Shriberg E, Stolcke A (2003) A prosody-based approach to end-of-utterance detection that does not require speech In *Proc. IEEE ICASSP*, pp. 608–611.

Gales M, Young S et al. (2008) The application of hidden markov models in speech recognition. *Foundations and Trends® in Signal Processing* 1:195–304.

Garrod S, Anderson A (1987) Saying what you mean in dialogue: A study in conceptual and semantic co-ordination. *Cognition* 27:181–218.

Georgila K, Black AW, Sagae K, Traum DR (2012) Practical evaluation of human and synthesized speech for virtual human dialogue systems. In *LREC*, pp. 3519–3526.

Georgila K, Henderson J, Lemon O (2005) Learning user simulations for information state update dialogue systems In *Ninth European Conference on Speech Communication and Technology*.

Georgila K, Traum D (2011) Reinforcement learning of argumentation dialogue policies in negotiation In *Twelfth Annual Conference of the International Speech Communication Association*.

Ghigi F, Eskenazi M, Torres MI, Lee S (2014) Incremental dialog processing in a task-oriented dialog In *Fifteenth Annual Conference of the International Speech Communication Association*.

Gorniak P, Roy D (2004) Grounded semantic composition for visual scenes. *Journal of Artificial Intelligence Research* pp. 429–470.

Graesser AC, Chipman P, Haynes BC, Olney A (2005) Autotutor: An intelligent tutoring system with mixed-initiative dialogue. *Education, IEEE Transactions on* 48:612–618.

Gratch J, Okhmatovskaia A, Lamothe F, Marsella S, Morales M, van der Werf R, Morency LP (2006) Virtual Rapport In Gratch J, Young M, Aylett R, Ballin D, Olivier P, editors, *Intelligent Virtual Agents*, Vol. 4133, chapter 2, pp. 14–27. Springer Berlin Heidelberg, Berlin, Heidelberg.

Graves A, Mohamed Ar, Hinton G (2013) Speech recognition with deep recurrent neural networks In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pp. 6645–6649. IEEE.

Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009) The weka data mining software: an update. *ACM SIGKDD explorations newsletter* 11:10–18.

Han T, Kennington C, Schlangen D (2015) Building and Applying Perceptually-Grounded Representations of Multimodal Scene Descriptions In *Proceedings of SEMDial*, Gothenburg, Sweden.

Hastie H, Aufaure MA, Alexopoulos P, Cuayáhuitl H, Dethlefs N, Gasic M, Henderson J, Lemon O, Liu X, Mika P et al. (2013) Demonstration of the parlance system: a data-driven, incremental, spoken dialogue system for interactive search. *Proc SIGDIAL* pp. 154–156.

Henderson J, Lemon O, Georgila K (2008) Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. *Computational Linguistics* 34:487–511.

Hochreiter S, Schmidhuber J (1997) Long short-term memory. *Neural computation* 9:1735–1780.

Hough J, Kennington C, Schlangen D, Ginzburg J (2015) Incremental semantics for dialogue processing: Requirements, and a comparison of two approaches In *Proceedings of the 11th International Conference on Computational Semantics*, pp. 206–216.

Hough J, Purver M (2014) Probabilistic type theory for incremental dialogue processing In *Proceedings of the EACL 2014 Workshop on Type Theory and Natural Language Semantics (TTNLS)*, pp. 80–88.

Hough J, Schlangen D (2017) Joint, incremental disfluency detection and utterance segmentation from speech In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, Vol. 1, pp. 326–336.

Huang THK, Ferraro F, Mostafazadeh N, Misra I, Agrawal A, Devlin J, Girshick R, He X, Kohli P, Batra D, Zitnick CL, Parikh D, Vanderwende L, Galley M, Mitchell M (2016) Visual storytelling In *Proceedings of NAACL–HLT*, pp. 1233–1239, San Diego, California, USA.

Huggins-Daines D, Kumar M, Chan A, Black AW, Ravishankar M, Rudnicky AI (2006) Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, Vol. 1, pp. I–I. IEEE.

Iida R, Kobayashi S, Tokunaga T (2010) Incorporating extra-linguistic information into reference resolution in collaborative task dialogue In *Proceedings of ACL*, pp. 1259–1267, Uppsala, Sweden.

Ji Y, Haffari G, Eisenstein J (2016) A latent variable recurrent neural network for discourse relation language models In *Proceedings of NAACL–HLT*, San Diego, California, USA.

Jiang R, Banchs RE, Kim S, Yeo KH, Niswar A, Li H (2014) Web-based multimodal multi-domain spoken dialogue system In *Proceedings of 5th International Workshop on Spoken Dialog Systems*.

Johnston M, Bangalore S, Vasireddy G, Stent A, Ehlen P, Walker M, Whittaker S, Maloor P (2002) MATCH: An architecture for multimodal dialogue systems In *Proceedings of ACL*, pp. 376–383, Philadelphia, Pennsylvania, USA.

Kalchbrenner N, Blunsom P (2013) Recurrent convolutional neural networks for discourse compositionality In *Proceedings of the ACL Workshop on Continuous Vector Space Models and their Compositionality*, Sofia, Bulgaria.

Kalpathy-Cramer J, Müller H, Bedrick S, Eggel I, de Herrera AGS, Tsikrika T (2011) Overview of the clef 2011 medical image classification and retrieval tasks. In *CLEF (notebook papers/labs/workshop)*, pp. 97–112.

Karkada D, Manuvirakurike R, Georgila K (2018) Towards understanding end-of-trip instructions in a taxi ride scenario In *Proceedings 14th Joint ACL-ISO Workshop on Interoperable Semantic Annotation*, pp. 98–107.

Kazemzadeh S, Ordonez V, Matten M, Berg T (2014) ReferItGame: Referring to objects in photographs of natural scenes In *Proceedings of EMNLP*, pp. 787–798, Doha, Qatar.

Kelleher JD, Kruijff GJM (2006) Incremental generation of spatial referring expressions in situated dialog In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pp. 1041–1048. Association for Computational Linguistics.

Kennington C, Schlangen D (2015) Simple learning and compositional application of perceptually grounded word meanings for incremental reference resolution In *Proceedings of the Conference for the Association for Computational Linguistics (ACL)*.

Kenny P, Parsons T, Gratch J, Rizzo A (2008) Virtual humans for assisted health care In *Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments*, p. 6. ACM.

Khanpour H, Guntakandla N, Nielsen R (2016) Dialogue act classification in domain-independent conversations using a deep recurrent neural network In *Proceedings of COLING*, Osaka, Japan.

Khouzaimi H, Laroche R, Lefevre F (2015) Optimising turn-taking strategies with reinforcement learning In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pp. 315–324.

Khouzaimi H, Laroche R, Lefèvre F (2016) Reinforcement learning for turn-taking management in incremental spoken dialogue systems. In *IJCAI*, pp. 2831–2837.

Kim D, Breslin C, Tsiakoulis P, Gašić M, Henderson M, Young S (2014) Inverse reinforcement learning for micro-turn management In *INTERSPEECH*, pp. 328–332, Singapore.

Koiso H, Horiuchi Y, Tutiya S, Ichikawa A, Den Y (1998) An analysis of turn-taking and backchannels based on prosodic and syntactic features in japanese map task dialogs. *Language and Speech* 41:295–321.

Kolář J, Shriberg E, Liu Y (2006) On speaker-specific prosodic models for automatic dialog act segmentation of multi-party meetings In *Interspeech*, Vol. 1.

Komatani K, Hotta N, Sato S, Nakano M (2015) User adaptive restoration for incorrectly-segmented utterances in spoken dialogue systems In *SIGDIAL*.

Kousidis S, Kennington C, Baumann T, Buschmeier H, Kopp S, Schlangen D (2014) A multimodal in-car dialogue system that tracks the driver's attention In *Proceedings of the 16th International Conference on Multimodal Interaction*, pp. 26–33. ACM.

Krishna R, Zhu Y, Groth O, Johnson J, Hata K, Kravitz J, Chen S, Kalantidis Y, Li LJ, Shamma DA et al. (2017) Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision* 123:32–73.

Kruijff GJM, Lison P, Benjamin T, Jacobsson H, Hawes N (2007) Incremental, multi-level processing for comprehending situated dialogue in human-robot interaction In *Symposium on Language and Robots*.

Kulkarni G, Premraj V, Ordonez V, Dhar S, Li S, Choi Y, Berg AC, Berg TL (2013) Babytalk: Understanding and generating simple image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35:2891–2903.

Kumar A, Eugenio BD, Aurisano J, Johnson A, Alsaiari A, Flowers N, Gonzalez A, Leigh J (2017) Towards multimodal coreference resolution for exploratory data visualization dialogue: Context-based annotation and gesture identification In *Proceedings of SemDial*, Saarbrücken, Germany.

Lafferty JD, McCallum A, Pereira FCN (2001) Conditional random fields: Probabilistic models for segmenting and labeling sequence data In *ICML*, pp. 282–289.

Lagoudakis MG, Parr R (2003) Least-squares policy iteration. *Journal of machine learning research* 4:1107–1149.

Laput GP, Dontcheva M, Wilensky G, Chang W, Agarwala A, Linder J, Adar E (2013) PixelTone: A multimodal interface for image editing In *Proceedings of the*

SIGCHI Conference on Human Factors in Computing Systems, pp. 2185–2194, Paris, France.

Lasecki W, Kamar E, Bohus D (2013) Conversations in the crowd: Collecting data for task-oriented dialog learning In *Human Computation Workshop on Scaling Speech and Language Understanding and Dialog through Crowdsourcing.*

Laskowski K, Shriberg E (2010) Comparing the contributions of context and prosody in text-independent dialog act recognition In *ICASSP*, pp. 5374–5377. IEEE.

Lee JY, Dernoncourt F (2016) Sequential short-text classification with recurrent and convolutional neural networks In *Proceedings of NAACL–HLT*, San Diego, California, USA.

Lemon O, Bracy A, Gruenstein A, Peters S (2001) Information states in a multi-modal dialogue system for human-robot conversation In *Proceedings of the 5th Workshop on Formal Semantics and Pragmatics of Dialogue (Bi-Dialog)*, pp. 57–67.

Lemon O, Georgila K, Henderson J, Stuttle M (2006) An isu dialogue system exhibiting reinforcement learning of dialogue policies: generic slot-filling in the talk in-car system In *Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics: Posters & Demonstrations*, pp. 119–122. Association for Computational Linguistics.

Lendvai P, Geertzen J (2007) Token-based chunking of turn-internal dialogue act sequences In *SIGDIAL*, pp. 174–181.

Li W, Wu Y (2016) Multi-level gated recurrent neural network for dialog act classification In *Proceedings of COLING*, Osaka, Japan.

Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft coco: Common objects in context In *European conference on computer vision*, pp. 740–755. Springer.

Litman DJ, Rotaru M, Nicholas G (2009) Classifying turn-level uncertainty using word-level prosody. In *INTERSPEECH*, pp. 2003–2006.

Litman DJ, Silliman S (2004) Itspoke: An intelligent tutoring spoken dialogue system In *Demonstration papers at HLT-NAACL 2004*, pp. 5–8. Association for Computational Linguistics.

Liu S, Seneff S, Glass J (2010) A collective data generation method for speech language models In *Spoken Language Technologies Workshop (SLT).*

Maaten Lvd, Hinton G (2008) Visualizing data using t-sne. *Journal of machine learning research* 9:2579–2605.

Maess B, Mamashli F, Obleser J, Helle L, Friederici AD (2016) Prediction signatures in the brain: semantic pre-activation during language comprehension. *Frontiers in human neuroscience* 10:591.

Manning CD, Surdeanu M, Bauer J, Finkel J, Bethard SJ, McClosky D (2014) The stanford corenlp natural language processing toolkit In *ACL: System Demonstrations*, pp. 55–60.

Manuvinakurike R, Bharadwaj S, Georgila K (2018) A dialogue annotation scheme for weight management chat using the trans-theoretical model of health behavior change In *Proceedings 14th Joint ACL-ISO Workshop on Interoperable Semantic Annotation (isa-14)*, p. 60.

Manuvinakurike R, Brixey J, Bui T, Chang W, Kim DS, Artstein R, Georgila K (2018a) Edit me: A corpus and a framework for understanding natural language image editing In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018).*

Manuvinakurike R, Bui T, Chang W, Georgila K (2018b) Conversational image editing: Incremental intent identification in a new dialogue task In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pp. 284–295, Melbourne, Australia. Association for Computational Linguistics.

Manuvinakurike R, DeVault D (2015) *Natural Language Dialog Systems and Intelligent Assistants*, chapter Pair Me Up: A Web Framework for Crowd-Sourced Spoken Dialogue Collection, pp. 189–201.

Manuvinakurike R, DeVault D, Georgila K (2017) Using reinforcement learning to model incrementality in a fast-paced dialogue game In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pp. 331–341.

Manuvinakurike R, Kennington C, DeVault D, Schlangen D (2016) Real-time understanding of complex discriminative scene descriptions In *Proceedings of the 17th Annual SIGdial Meeting on Discourse and Dialogue.*

Manuvinakurike R, Paetzel M, DeVault D (2015) Reducing the cost of dialogue system training and evaluation with online, crowd-sourced dialogue data collection. *SEMDIAL 2015 goDIAL* p. 113.

Manuvinakurike R, Paetzel M, Qu C, Schlangen D, DeVault D (2016) Toward incremental dialogue act segmentation in fast-paced interactive dialogue systems In *Proceedings of the 17th Annual SIGdial Meeting on Discourse and Dialogue.*

Manuvinakurike R, Velicer WF, Bickmore TW (2014) Automated indexing of internet stories for health behavior change: weight loss attitude pilot study. *Journal of medical Internet research* 16.

Manuvirakurike R, Brixey J, Bui T, Chang W, Artstein R, Georgila K (2018) DialEdit: Annotations for spoken conversational image editing In *Proceedings 14th Joint ACL - ISO Workshop on Interoperable Semantic Annotation*, pp. 1–9, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Marcus MP (1995) Text Chunking using Transformation-Based Learning In *In Pro- ceedings of the Third Workshop on Very Large Cor- pora*, pp. 82–94.

Marslen-Wilson W (1973) Linguistic structure and speech shadowing at very short latencies. *Nature* 244:522.

Matheson C, Poesio M, Traum D (2000) Modelling grounding and discourse obligations using update rules In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pp. 1–8. Association for Computational Linguistics.

McCallister E, Grance T, Scarfone KA (2010) Sp 800-122. guide to protecting the confidentiality of personally identifiable information (pii) .

McCallum AK (2002) Mallet: A machine learning for language toolkit http://mallet.cs.umass.edu.

Mcnally L, Boleda G (2015) Conceptual vs. Referential Affordance in Concept Composition In *Compositionality and Concepts in Linguistics and Psychology*, pp. 1–20.

Meena R, Boye J, Skantze G, Gustafson J (2014) Crowdsourcing street-level geographic information using a spoken dialogue system In *The 15th Annual SIGdial Meeting on Discourse and Dialogue (SIGDIAL).*

Milajevs D, Kartsaklis D, Sadrzadeh M, Purver M, Science C, Road ME (2014) Evaluating Neural Word Representations in Tensor-Based Compositional Settings In *EMLNP*, pp. 708–719.

Miller AH, Feng W, Fisch A, Lu J, Batra D, Bordes A, Parikh D, Weston J (2017) Parlai: A dialog research software platform. *arXiv preprint arXiv:1705.06476* .

Mills D, Martin J, Burbank J, Kasch W (2010) Network time protocol version 4: Protocol and algorithms specification.

Mitchell M, Bohus D, Kamar E (2014) Crowdsourcing language generation templates for dialogue systems In *the 8th International Natural Language Generation Conference (INLG).*

Morbini F, Audhkhasi K, Sagae K, Artstein R, Can D, Georgiou P, Narayanan S, Leuski A, Traum D (2013) Which ASR should I choose for my dialogue system? In *SIGDIAL*, Metz, France.

Morbini F, Sagae K (2011) Joint identification and segmentation of domain-specific dialogue acts for conversational dialogue systems In *Proceedings of ACL: Human Language Technologies: short papers*, pp. 95–100.

Mostafazadeh N, Brockett C, Dolan B, Galley M, Gao J, Spithourakis G, Vanderwende L (2017) Image-grounded conversations: Multimodal context for natural question and response generation In *Proceedings of IJCNLP*, pp. 462–472, Taipei, Taiwan.

Nakano M, Miyazaki N, Hirasawa Ji, Dohsaka K, Kawabata T (1999) Understanding unsegmented user utterances in real-time spoken dialogue systems In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pp. 200–207. Association for Computational Linguistics.

Ortega D, Vu NT (2017) Neural-based context representation learning for dialog act classification In *Proceedings of SIGDIAL*, Saarbrücken, Germany.

Paetzel M, Manuvinakurike R, DeVault D (2015) "So, which one is it?" The effect of alternative incremental architectures in a high-performance game-playing agent In *Proceedings of the 16th Annual SIGdial Meeting on Discourse and Dialogue.*

Paetzel M, Racca DN, DeVault D (2014) A multimodal corpus of rapid dialogue games In *LREC.*

Pagliardini M, Gupta P, Jaggi M (2018) Unsupervised learning of sentence embeddings using compositional n-gram features In *Proceedings of NAACL–HLT*, New Orleans, Louisiana, USA.

Panayotov V, Chen G, Povey D, Khudanpur S (2015) Librispeech: an asr corpus based on public domain audio books In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210. IEEE.

Parent G, Eskenazi M (2010) Toward better crowdsourced transcription: transcription of a year of the let's go bus information system data In *IEEE Workshop on Spoken Language Technology.*

Pennington J, Socher R, Manning C (2014) Glove: Global vectors for word representation In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543.

Petukhova V, Bunt H (2011) Incremental dialogue act understanding In *Proceedings of the Ninth International Conference on Computational Semantics*, pp. 235–244. Association for Computational Linguistics.

Petukhova V, Bunt H (2014) Incremental recognition and prediction of dialogue acts In *Computing Meaning*, pp. 235–256. Springer.

Pickering MJ, Gambi C (2018) Predicting while comprehending language: a theory and review. *Psychological Bulletin* .

Pieraccini R, Suendermann D, Dayanidhi K, Liscombe J (2009) Are we there yet? research in commercial spoken dialog systems In *Text, Speech and Dialogue*, pp. 3–13. Springer.

Plátek O, Jurčíček F (2014) Free on-line speech recogniser based on Kaldi ASR toolkit producing word posterior lattices In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pp. 108–112, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

Povey D, Ghoshal A, Boulianne G, Burget L, Glembek O, Goel N, Hannemann M, Motlicek P, Qian Y, Schwarz P et al. (2011) The kaldi speech recognition toolkit In *IEEE 2011 workshop on automatic speech recognition and understanding.*

Quirk R, Greenbaum S, Leech G, Svartvik J (1985) *A Comprehensive grammar of the English language* General Grammar Series. Longman.

Raux A, Eskenazi M (2008) Optimizing endpointing thresholds using dialogue features in a spoken dialogue system In *SIGDIAL*, pp. 1–10.

Raux A, Langner B, Bohus D, Black A, Eskenazi M (2005) Let's go public! taking a spoken dialog system to the real world In *Interspeech.*

Rieser V, Lemon O, Keizer S (2014) Natural language generation as incremental planning under uncertainty: adaptive information presentation for statistical dialogue systems. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22:979–994.

Ruan S, Wobbrock JO, Liou K, Ng A, Landay J (2016) Speech is 3x faster than typing for english and mandarin text entry on mobile devices. *arXiv preprint arXiv:1608.07323* .

Salverda AP, Kleinschmidt D, Tanenhaus MK (2014) Immediate effects of anticipatory coarticulation in spoken-word recognition. *Journal of memory and language* 71:145–163.

Sato R, Higashinaka R, Tamoto M, Nakano M, Aikawa K (2002) Learning decision trees to determine turntaking by spoken dialogue systems In *Proceedings of ICSLP-02*, pp. 861–864.

Schalkwyk J, Beeferman D, Beaufays F, Byrne B, Chelba C, Cohen M, Kamvar M, Strope B (2010) "our word is my comman": google search by voice: A case study In *Advances in speech recognition*, pp. 61–90. Springer.

Schlangen D, Baumann T, Atterer M (2009) Incremental reference resolution: The task, metrics for evaluation, and a Bayesian filtering model that is sensitive to disfluencies In *The 10th Annual SIGDIAL Meeting on Discourse and Dialogue (SIGDIAL 2009).*

Schlangen D, Diekmann T, Ilinykh N, Zarrieß S (2018) slurk–a lightweight interaction server for dialogue experiments and data collection In *Proceedings of the 22nd Workshop on the Semantics and Pragmatics of Dialogue (AixDial/semdial 2018).*

Schlangen D, Skantze G (2011) A general, abstract model of incremental dialogue processing. *Dialogue and Discourse* 2:83–111.

Schlangen D, Zarrieß S, Kennington C (2016) Resolving references to objects in photographs using the words-as-classifiers model In *Proceedings of ACL*, pp. 1213–1223, Berlin, Germany.

Sedivy JC, Tanenhaus MK, Chambers CG, Carlson GN (1999) Achieving incremental semantic interpretation through contextual representation. *Cognition* 71:109–147.

Selfridge E, Arizmendi I, Heeman P, Williams J (2013) Continuously predicting and processing barge-in during a live spoken dialogue task In *Proceedings of the SIGDIAL 2013 Conference*, pp. 384–393.

Selfridge E, Heeman P (2010) Importance-driven turn-bidding for spoken dialogue systems In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 177–185, Uppsala, Sweden.

Selfridge EO, Arizmendi I, Heeman PA, Williams JD (2012) Integrating incremental speech recognition and pomdp-based dialogue systems In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pp. 275–279, Seoul, South Korea. Association for Computational Linguistics.

Shen Ss, Lee Hy (2016) Neural attention models for sequence classification: Analysis and application to key term extraction and dialogue act detection In *arXiv preprint arXiv:1604.00077*.

Shriberg E, Stolcke A, Hakkani-Tür D, Tür G (2000) Prosody-based automatic segmentation of speech into sentences and topics. *Speech communication* 32:127–154.

Skantze G (2017) Predicting and regulating participation equality in human-robot conversations: Effects of age and gender In *Proceedings of HRI*, pp. 196–204, Vienna, Austria.

Skantze G, Hjalmarsson A (2010) Towards incremental speech generation in dialogue systems In *Proceedings of the SIGDIAL 2010 Conference*, pp. 1–8, Tokyo, Japan. Association for Computational Linguistics.

Skantze G, Schlangen D (2009) Incremental dialogue processing in a micro-domain In *EACL*, pp. 745–753.

Socher R, Karpathy A, Le QV, Manning CD, Ng AY (2014) Grounded Compositional Semantics for Finding and Describing Images with Sentences. *Transactions of the ACL (TACL)* .

Stent A, Bangalore S (2014) *Natural language generation in interactive systems* Cambridge University Press.

Stoia L, Shockley DM, Byron DK, Fosler-Lussier E (2008) SCARE: A situated corpus with annotated referring expressions In *Proceedings of LREC*, pp. 650–653, Marrakech, Morocco.

Stolcke A, Coccaro N, Bates R, Taylor P, Van Ess-Dykema C, Ries K, Shriberg E, Jurafsky D, Martin R, Meteer M (2000) Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics* 26:339–373.

Stratou G, Morency LP, DeVault D, Hartholt A, Fast E, Lhommet M, Lucas G, Morbini F, Georgila K, Scherer S et al. (2015) A demonstration of the perception system in simsensei, a virtual human application for healthcare interviews In *2015 International Conference on Affective Computing and Intelligent Interaction (ACII)*, pp. 787–789. IEEE.

Suendermann D, Liscombe J, Bloom J, Li G, Pieraccini R (2011) Large-scale experiments on data-driven design of commercial spoken dialog systems In *Interspeech*.

Suendermann-Oeft D, Ramanarayanan V, Teckenbrock M, Neutatz F, Schmidt D (2015) Halef: An open-source standard-compliant telephony-based modular spoken dialog system: A review and an outlook In *Natural language dialog systems and intelligent assistants*, pp. 53–61. Springer.

Sutton RS, Barto AG (1998) *Reinforcement learning: An introduction*, Vol. 1 MIT press Cambridge.

Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9.

Takenobu T, Ryu I, Asuka T, Naoko K (2012) The REX corpora: A collection of multimodal corpora of referring expressions in collaborative problem solving dialogues In *Proceedings of LREC*, pp. 422–429, Istanbul, Turkey.

Takeuchi M, Kitaoka N, Nakagawa S (2004) Timing detection for realtime dialog systems using prosodic and linguistic information In *Speech Prosody 2004*.

Tanenhaus MK, Spivey-Knowlton MJ, Eberhard KM, Sedivy JC (1995) Integration of visual and linguistic information in spoken language comprehension. *Science* 268:1632–1634.

Taylor ME, Stone P (2009) Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research* 10:1633–1685.

Torrey L, Shavlik J (2010) Transfer learning In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, pp. 242–264. IGI Global.

Toutanova K, Manning CD (2000) Enriching the knowledge sources used in a maximum entropy part-of-speech tagger In *In EMNLP/VLC 2000*, pp. 63–70.

Tran QH, Zukerman I, Haffari G (2017) A generative attentional neural network model for dialogue act classification In *Proceedings of ACL – Short Papers*, Vancouver, Canada.

Traum D, Georgila K, Artstein R, Leuski A (2015) Evaluating spoken dialogue processing for time-offset interaction In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pp. 199–208.

Vinyals O, Toshev A, Bengio S, Erhan D (2015) Show and tell: A neural image caption generator In *Computer Vision and Pattern Recognition*.

Visser T, Traum D, DeVault D, op den Akker R (2014) A model for incremental grounding in spoken dialogue systems. *Journal on multimodal user interfaces* 8:61–73.

Wang W, Bohus D, Kamar E, Horvitz E (2012) Crowdsourcing the acquisition of natural language corpora: Methods and observations In *Spoken Language Technology Workshop (SLT)*, pp. 73–78.

Warnke V, Kompe R, Niemann H, Nï£¡th E (1997) Integrated dialog act segmentation and classification using prosodic features and language models In *Proc. 5th Europ. Conf. on Speech, Communication, and Technology*, Vol. 1.

Wen TH, Gasic M, Mrkšić N, Su PH, Vandyke D, Young S (2015) Semantically conditioned lstm-based natural language generation for spoken dialogue systems In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1711–1721.

Whitney D, Eldon M, Oberlin J, Tellex S (2016) Interpreting multimodal referring expressions in real time In *Proceedings of ICRA*, pp. 3331–3338, Stockholm, Sweden.

Williams JD (2011) An empirical evaluation of a statistical dialog system in public use In *Proceedings of the SIGDIAL 2011 Conference*, pp. 130–141. Association for Computational Linguistics.

Williams JD, Kamal E, Ashour M, Amr H, Miller J, Zweig G (2015) Fast and easy language understanding for dialog systems with microsoft language understanding intelligent service (luis) In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pp. 159–161.

Xiong W, Wu L, Alleva F, Droppo J, Huang X, Stolcke A (2018) The microsoft 2017 conversational speech recognition system In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5934–5938. IEEE.

Yang Z, Li B, Zhu Y, King I, Levow G, Meng H (2010) Collection of user judgments on spoken dialog system with crowdsourcing In *Spoken Language Technologies Workshop (SLT)*.

Zarrieß S, Gambino MSL, Schlangen D (2017) Refer-itts: A system for referring in spoken installments to objects in real-world images In *Proceedings of the 10th International Conference on Natural Language Generation*, pp. 72–73.

Zarrieß S, Hough J, Kennington C, Manuvinakurike R, DeVault D, Schlangen D (2016) Pentoref: A corpus of spoken references in task-oriented dialogues. .

Zhao T, Kawahara T (2019) Joint dialog act segmentation and recognition in human conversations using attention to dialog context. *Computer Speech & Language* .

Zimmermann M (2009) Joint segmentation and classification of dialog acts using conditional random fields In *Interspeech*.

Zimmermann M, Liu Y, Shriberg E, Stolcke A (2006) *Second International Workshop, MLMI 2005*, chapter Toward Joint Segmentation and Classification of Dialog Acts in Multiparty Meetings, pp. 187–193.