INCREMENTAL PROCESSING FOR IMPROVING CONVERSATIONAL GROUNDING IN A CHATBOT

by

Aprajita Shukla

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Science

Boise State University

August 2019

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Aprajita Shukla

Thesis Title: Incremental Processing for Improving Conversational Grounding in a Chatbot

Date of Final Oral Examination: 3 August 2019

The following individuals read and discussed the thesis submitted by student Aprajita Shukla, and they evaluated the presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Dr. Casey Kennington, Ph.D.	Chair, Supervisory Committee
Dr. Jerry Alan Fails, Ph.D.	Member, Supervisory Committee
Dr. Maria Soledad Pera, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by Dr. Casey Kennington, Ph.D., Chair of the Supervisory Committee. The thesis was approved by the Graduate College.

Dedicated to my parents who are highly statistically significant.

"I know you think you understand what you thought I said, but I'm not sure you realize that what you heard is not what I meant."

— Alan Greenspan

ACKNOWLEDGMENTS

I would like to thank the Boise State Computer Science Department for funding this project, and my adviser, Dr. Casey Kennington for taking me under his wing. I appreciate that while he gave me the freedom to decide how to achieve my goals he was also nurturing me throughout my journey, guiding me every time I tended to lose track. I wish to express gratitude to my committee members Dr. Jerry Alan Fails and Dr. Maria Soledad Pera for agreeing to be on my committee and providing prompt help whenever approached, even for caffeine emergencies.

I would like to thank my family and friends back in India for their immense support and for bearing with me when I missed out on important events to work towards completing my degree and not disbarring me when I took undue advantage of the same to intentionally miss those sometimes; the new friends I made here who are like a breath of fresh air in my life, reinvigorating me to give my best on numerous occasions in the course of my masters; our SLIM research group members, for inspiring a lot of light-bulb moments so that my paper, posters and presentations could reach perfection.

I would also like to acknowledge the fine products of Cadbury, Birmingham, UK for keeping me in high spirits in general, Frasier and Marvelous Mrs. Maisel for keeping me entertained during countless hours of writing and proofreading. It is only when one writes that one realizes the power of LaTeX. Thank you Leslie Lamport! Sending heartfelt thanks and good vibes to everyone who supported me in completing this thesis.

ABSTRACT

Current Digital Personal Assistants can be quite efficient while performing routine tasks like setting up reminders and looking up information. However, they do not attempt to establish common ground–the process of establishing and building mutual understanding–and require a significant amount of initial data to learn how to understand user intent. In this thesis, an incremental processing framework is leveraged through a chatbot interface which updates its understanding state at each inputted word, asks the user to clarify input when the system is unsure and prompts users to give feedback several times during an interaction, all of which are instrumental in establishing conversational grounding between them and enable the system to begin with little or no training data. User interactions can be utilized as labeled data for retraining the model and improving it. We evaluated our model with users on Amazon Mechanical Turk and with each iteration–retraining the model with the labeled data from previous interactions and opening it for new users–this conversational grounding model learns a mapping between the users' words and the actions performed by the system to improve the chatbots natural dialogue. Hence, demonstrating that since dialog processing involves language, it should be seen as a type of joint activity that requires coordination of both participants to establish common ground in order to communicate successfully and systems like these that have provisions for conversational grounding can work with little or no training data. Moreover, our data shows that with each update of the model the user affinity towards the system increased and the users prefer a system that asks for multiple clarifications

over the course of interaction than a system that assumes understanding of utterances without giving any explicit feedback.

TABLE OF CONTENTS

ABSTRACT vii

LIST OF TABLES xi

LIST OF FIGURES xii

LIST OF ABBREVIATIONS xiv

1 Introduction 1

1.1 Grounding Between Humans 1

1.2 Grounding between a Human and an Autonomous System 5

1.3 Thesis Statement 10

2 Background and Related Work 11

3 Our Model 23

3.1 Graphical User Interface 24

3.2 Natural Language Understanding 31

3.3 Dialogue Management 33

3.4 System Description 35

4 Data Collection 39

4.1 Our Experiment 41

4.1.1 Task 41

4.1.2 Performing the Experiment 41
4.1.3 Data Description 42
4.1.4 Evaluation 43

5 Conclusion and Future work 50
5.1 Future Work 51

REFERENCES 53

LIST OF TABLES

4.1 Data collected over four *iterations* 42

LIST OF FIGURES

1.1 Common ground between two humans 2
1.2 Common ground between a human and a robot 6

2.1 Input processed by Incremental and Sequential Processor. Even a much slower Incremental Processor can finish before a Sequential Processor [**?**] 13
2.2 Example of IU network; part-of-speech tags are grounded into words, tags and words have same level links with left IU; *four* is revoked and replaced with *forty* [2] 14
2.3 Expanding-right tree GUI used in amBrOISEa [10] 19
2.4 GUI depicting clarification through the *'?'* symbol. Values it is confused between are denoted by the red dot preceding them [10] 20
2.5 *Low* is selected as the intended value for *price* [10] 20
2.6 Depicting complete intent. Suggestions pane showing suggestions based on intent [10] 21

3.1 Chatbot's response 25
3.2 User utterance for Fig 3.1 25
3.3 Chatbot's response 26
3.4 User utterance for Fig 3.3 26
3.5 Chatbot's response 27
3.6 User utterance for Fig 3.5 27

3.7 A Clarification Request (CR) for *firstname* slot 28
3.8 User responding *no* to a CR for the *firstname* token 29
3.9 User responding *yes* to a CR for the *firstname* token 29
3.10 *Reset* button press response 30
3.11 *Go Back* button press response 31
3.12 A picture of an entire frame, where the red rectangle shows the slots i.e. *firstname*, *lastname*, *email*, *notes* and *web* with the values that user filled them with 32
3.13 Depicting *intent*, *concept* and *properties* 36
3.14 Sample of training examples 36
3.15 System asking for feedback 38

4.1 Overview of data collection setup 40
4.2 Rating each contact information entered 42
4.3 Sum of happy, sad, angry and dissatisfied feedback faces per iteration . 43
4.4 Number of CRs versus the happy faces in each iteration 44
4.5 Number of happy feedback for the amount of training data per iteration 45
4.6 Total number of CRs per iteration 46
4.7 Total number of training examples per iteration 47
4.8 Total number of tokens (slots) filled by the user with total counts at the top 47

LIST OF ABBREVIATIONS

DPAs – Digital Personal Assistants

CRs – Clarification Requests

NLU – Natural Language Understanding

DM – Dialogue Manager

IU – Incremental Unit

SLL – Same Level Links

GRIN – Grounded In Links

SIUM – Simple Incremental Update Model

MTurk – Amazon Mechanical Turk

DS – Dialogue System

SDS – Spoken Dialogue System

MDS – Multimodal Dialogue System

UX – User Experience

GUI – Graphical User Interface

NLG – Natural language Generation

PaaS – Platform as a Service

URL – Uniform Resource Locator

HIT – Human Intelligence Task

CHAPTER 1

INTRODUCTION

The common ground between two humans in a dialog includes mutual beliefs and knowledge about the shared environment–about what they both can see, about what they both have already talked about, etc. Common ground is also a form of self-awareness. For example, two people, Susan and Bill, are aware of certain information they each have. To have common ground, their awareness must be reflexive - it must include that very awareness itself [5]. A significant body of work has investigated common ground in human-human communication, for example, as reported in [3] we see how a shared cultural background, and spatial reasoning capabilities affect human communication.

In the following section, we explain further and give examples of how grounding is accomplished between two humans.

1.1 Grounding Between Humans

In Figure 1.1, we see a father and son enjoying their day out together. Their respective individual knowledge is shown on each side with their common ground (within green boxes), which includes all that they both mutually perceive. There is some information that is only part of their own individual knowledge e.g., only the father knows that the bird is an eastern bluebird, which the son is not aware of

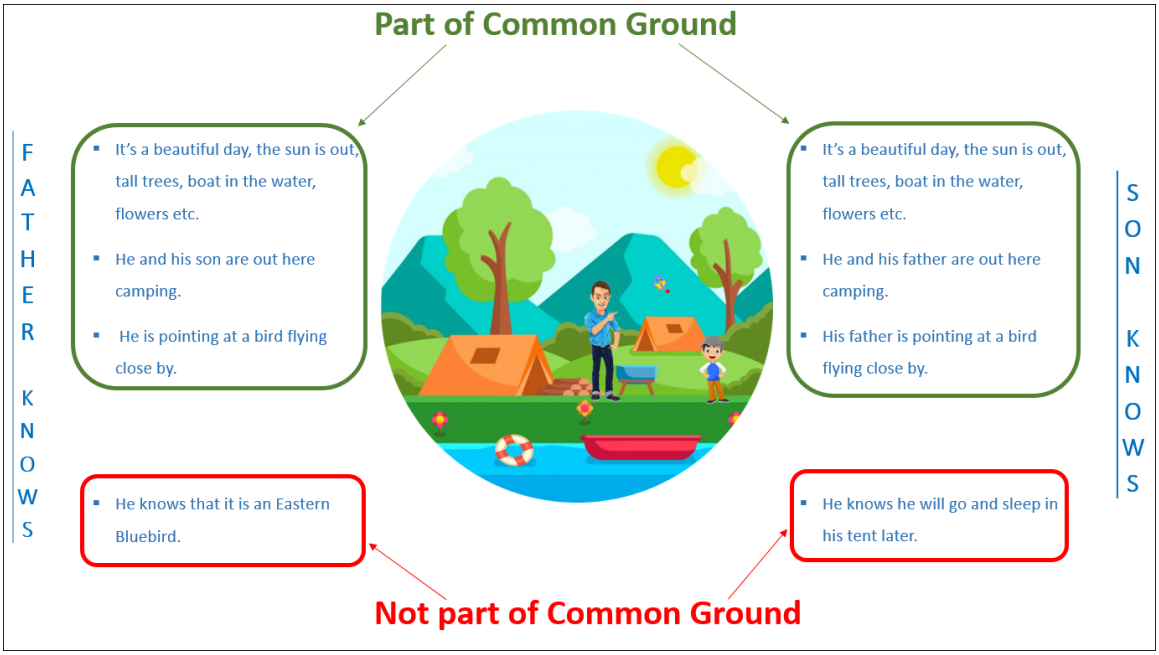


Figure 1.1: Common ground between two humans

since it is not part of the son's knowledge, and therefore not part of the common ground between them. Similarly, the knowledge that the son will go and sleep later is something only he is aware of. It is only when the father and the son start a dialog, that they can begin to unravel each other's intent, share more knowledge in the dialog process, and add some information as part of common ground by reaching a mutual understanding. This is done by taking turns, clarifying and providing feedback for each other to signal understanding. In other words, we can break grounding down into four distinct constituents:

- Knowledge of language
- Displays of understanding
- Displays of misunderstanding

- Memory

For the rest of this section, I will first explain each of these for human-human dialog in detail followed by a comparison of the same in the case of dialog between a human and an autonomous system.

Knowledge of Language Following Pierre Lison's explanation of grounding, [13], we as humans associate words with objects we see around us, knowledge that we read about, emotions that we feel, and abstract ideas. We bring that knowledge of a language with us when one person talks to another person: both dialog participants can assume certain pieces of common ground with each other simply because they are aware that they are talking to another human and they have similar associations between words, objects, emotions, and abstract ideas. Moreover, when two people with little or no background with each other begin interacting, they are also learning from that conversation; building a mapping between the words and the knowledge gained from the interaction. This continual interplay of using words and comprehending words alters the dialog participants' understanding of the meaning of those words. In other words, language understanding must happen during the dialog for the two participants to understand each other, but the dialog itself plays into how the participants understand language.

Displays of Understanding But how does a dialog participant even know that their dialog partner is understanding when they speak? When two people talk to each other, the dialog partners take turns playing the role of *speaker* and *listener*. When a speaker is speaking, the listener gives *feedback* to indicate understanding by different

means, for example nodding intermittently, maintaining eye contact, saying *okay* or *yes* at intervals, or even completing the speaker's sentences. Even if the listener gives no feedback, it is still safe to assume that listener understood the speaker's intent if they both share the same knowledge. For example, two computer science professors talking about efficiency of algorithms will take as common ground the knowledge that both of them are experts of the same field.

Displays of Misunderstanding Unfortunately, natural language is a challenging communication medium even for people which means that two individuals are not likely to completely understand one another. To mitigate misunderstandings, signaling mechanisms are ideal for coordinating what the speakers mean with what their listener understands them to mean [5]. During a conversation when the listener is not sure about their own understanding they stop the speaker to take a turn and clarify by asking the speaker what he meant. These are Clarification Requests (CRs) and as reported in [14], in human-human dialog, 3-6% of *dialog acts*–a functional unit of a dialog used by the speaker to change the context [13]–are CRs. Humans can ask targeted types of clarification depending on where they isolated the misunderstanding, for example phonetic, syntactic, semantic, or referential misunderstandings.

Memory With knowledge of language that two dialog participants can use to effectively communicate, and with the help of feedback to signal understanding, and CRs to signal misunderstanding, the very nature of two people talking to each other builds important information not just about language or the dynamics of dialog, but about the indiviuals themselves. Two individuals may talk about their

hobbies, names of family members, or their occupations. This information goes beyond language understanding and links those attributes to those individuals. Dialog partners learning attributes about each other is an important part of building common ground.

1.2 Grounding between a Human and an Autonomous System

A human dialog participant uses knowledge of language, feedback, CRs, and memory of their dialog partner to establish common ground with that partner. What happens when one of the dialog participants is an autonomous system of some kind (e.g., a robot, personal assistant, virtual agent, etc.)? Can we assume that systems can make use of the four aspects of grounding that I described above?

[3, 6] argues that in human-system dialog, although humans and systems are co-present in a shared environment, they have significantly mismatched capabilities in perceiving the shared environment. In order for humans and systems to communicate with each other successfully using language, it is important for them to mediate such differences and to establish a common ground.

This is further illustrated in the same scenario in Figure 1.1 by replacing the boy in the figure with a robot, as depicted in Figure 1.2. In this case, the robot has highly limited knowledge of language and perceptual information about the scene are severely mismatched (according to [3]). This misalignment of knowledge bases is what makes it harder for a natural dialog to occur in human-robot dialog.

In the following, I explain the implications of this misalignment on the four aspects of grounding.

Figure 1.2: Common ground between a human and a robot

Knowledge of Language Language in automated systems is usually modeled from interaction data. In a more formal sense, knowledge of language in a system amounts to learning a mapping from utterances (i.e., a transcription or typed input into the system from the human "speaker") to an abstract, computable representation that lends itself to some specific task. These data-driven systems require large amounts of annotated data to learn the mappings between the spoken input and the system response. Acquiring this data usually involves a lot of time, effort and money. Sometimes even a minor change to the model or task renders previously obtained data useless when applied to a different model or novel task.

Data is tied to grounding in an important way: if underlying models of a

Digital Personal Assistant (DPA) system learns how to map between user utterance and system response, this is itself a learning of what kinds of actions are expected. This makes the data collection process challenging. Approaches for data collection have included Wizard-of-Oz (e.g., [17]) studies where a human participant performs some kind of task with a system, only (unknown to the human participant), the system is actually being controlled by another human. Though this results in realistic data to fully automate a system to perform the task in question, the "wizard" behind the scenes may introduce complexities into the interaction which could render the data useless. Data obtained this way is helpful for analysis and improving the system but it is manufactured to a great extent. Therefore, whenever a model is retrained, significant amount of data is needed for evaluation at each stage and the initial lack of it becomes a huge roadblock in making marked progress.

Displays of Understanding Current systems can signal understanding, but it is usually in the form of a system turn where the system utters *okay* or *yes*. Current systems do not indicate understanding as utterances unfold like humans do (i.e., with head nods, etc., as explained above). Moreover, even if systems signal understanding by uttering *okay*, it's still not clear to the user if the system did actually understand, which leads to incorrect system actions and frustration on the side of the user.

Displays of Misunderstanding CRs are even more crucial in automated systems than between two humans because systems are more prone to miscommunications. Yet current DPAs don't signal misunderstandings in a natural way: when a system misunderstanding happens, they take a dialog turn and offer a vague request for

clarification which usually requires the user to repeat their entire utterance. This is frustrating to users because, as mentioned above, humans are able to produce targeted CRs, isolating the specific point of miscommunication (e.g., a referrential or phonetic misunderstanding).

Sometimes the user might see an *okay* or a *yes* pop up on the screen when a chatbot tries to signal understanding through feedback. This *okay* or *yes*, however, does not necessarily mean that it understood the user's input; it usually is just the system recognized transcript and there is no guarantee that the request was actually understood or that it will lead to the right system action. The user usually assumes that the system understood a user utterance because it did not ask for any clarifications like we humans do while interacting with other humans. Current systems only display ongoing understanding of an utterance by displaying the state of the transcribed speech to the user, but even perfect transcription does not mean understanding, as mentioned in [9]. What is not displayed is the system's semantic representation of the transcription because it's not easily understandable by users.

Memory Another major shortcoming of current systems is the lack of memory. The following interactions between a known system (S) and a user (U) as reported in [12] illustrates this:

(1)

a. U: Hey S, call my mom.

b. S: I don't know "mom."

c. U: My mom is Martha.

d. S: OK, calling Martha.

(2)

a. U: Hey S, call my mom.

b. S: I don't know "mom."

Following [12], by uttering *OK*, the system makes the user think that system signaled understanding. However, the user was later surprised that the system misunderstood, as evidenced in (2-a). This is a prime example of mismatches of communication as noted in [3]. More specifically in this case, the system had no memory to store and recall facts about users and interactions in order to build mutual understanding. For this kind of grounding to be accomplished, the system should dynamically update with each interaction while signaling understanding to the user.

Discussion Ideally, a system should attempt to establish *common ground*–the process of establishing and building mutual understanding–between itself and users with knowledge of language (i.e., by learning a mapping between aspects of language and a semantic abstraction), signaling understanding to users at appropriate times, signalling misunderstandings when necessary, and by remembering important information about the dialog partner [10].

1.3 Thesis Statement

In this work I hypothesize that conversational grounding in human-system dialog –manifested through incremental processing– enables the system to work with little or no training data.

To address my hypothesis I will leverage a Graphical User Interface (GUI), following my previous work in [10] to display the internal state representation of our Dialog System (DS) in an intuitively readable way to the user and construct a system that can work with little or no training data and have it learn autonomously as it interacts with the user. At every turn of the dialogue, the system will attempt to incrementally establish and maintain common ground through feedbacks and CRs which will make the dialog feel more natural.

The next Chapter talks about the related background work in this domain that is an amalgamation of Computer Science, Data Science and Psychology. Chapter 3 is where I describe the model of our system and provide an in depth understanding of the various modules and their connections that make the incremental processing model work. Then in Chapter 4 I explain our data collection mode and method followed by the experiment I conducted, and evaluation of results. Lastly, In Chapter 5 and 6, I present our conclusion and take a look at the future work that my work in this thesis can potentially serve as a foundation for.

CHAPTER 2

BACKGROUND AND RELATED WORK

A DS is a program that interacts with a human in a natural language, usually through text, speech, or both. Prevalent DSs are commonly used by people from all walks of life to perform simple routine tasks like getting weather updates, setting up a reminder or making a phone call. Generally speaking, one's personal assistant performs such tasks; therefore, it is befitting to collectively refer to them as DPAs e.g. SIRI, Alexa, Cortana, Foxsy, Meekan, just to name a few. Based on the mode of communication, these DPAs can be categorized mainly into (a) Spoken Dialog Systems (SDSs) that use speech to interact with the user, e.g. Cortana (b) Chatbots that use text, e.g. ELIZA and (c) Multimodal Dialog Systems (MDSs) that can process two or more modes of input, e.g. MATCH that utilizes speech, text and haptic input. My thesis work uses a chatbot interface to signal understanding to the user in an intuitive way which helps to incrementally establish common ground with the system. The concept of having common ground is potentially generalizable to other DSs with additional work needed to be done to suit each type.

Irrespective of the communication medium, interaction with all the DPAs happens through language use and takes the form of dialog. By virtue of being humans, we are accustomed to a certain way of engaging in dialogue that revolves

around assuming certain pieces of common ground between participants. Since humans use DPAs to accomplish tasks, delegate small jobs and enable multi-tasking in a hands-free situation, the nature of conversing with them should, to the greatest extent possible, mirror a conversation between two human participants. This will not only reduce human cognitive load but will also foster a dialog that is effortless and less frustrating, and feels more natural.

To accomplish this, I will utilize an incremental processing framework to demonstrate the process of *grounding* i.e. the process of establishing common ground in dialog [4]. In any dialog, participants take turns to communicate successfully. Dialog between two participants is a type of *joint activity* made up of *joint actions* called *dialog acts*, e.g. utterances, clarifications, feedback, etc. A joint action is one performed by an ensemble of people acting in coordination with each other. To complete any joint activity, participants need to perform joint actions to advance and they cannot take joint actions without assuming certain pieces of common ground [5]. Since dialog participants take actions as a whole, they need to have a combined understanding of all the fundamental knowledge that is prerequisite to performing a joint action. This shared understanding is in effect, the sum of their mutual, common, or joint knowledge, beliefs and suppositions [3, 5, 11] called *common ground* and it has three components. First is the knowledge that each participant has before the start of the joint activity; second is the knowledge they gain with each completed step of the joint action; third is the awareness of all the knowledge. Therefore, the common ground is first established and more information is added to it increment by increment. This is how dialog participants include the common knowledge that participants share about the task as it proceeds. In some cases, information is also

removed from the common ground through clarifications and feedbacks to ensure a high degree of accuracy in the estimate of common ground which both participants have. The remarkable thing about common ground is that both participants are aware of both–what they know and what the other participant knows. This common ground is the key to coordination in dialog.

It is known that dialog processing is, by its very nature, *incremental* [15] and even then most dialogue agents process whole dialogues. No dialogue agent (artificial or natural) processes whole dialogs, if only for the simple reason that dialogs are *created* incrementally, by participants taking turns. At the level of taking-turns, most currently implemented dialog systems are incremental i.e. they process user utterances as a whole and produce their response utterances as a whole but it is not so when we consider processing parts of the utterances. [9]

Incremental systems hence are those where 'Each processing component will be triggered into activity by a minimal amount of its characteristic input.' If we assume that the characteristic input of a dialogue system is the utterance, we would expect an incremental system to work on units smaller than utterances. Fig 2.1 illustrates how an incremental processor compares to a sequential processor.

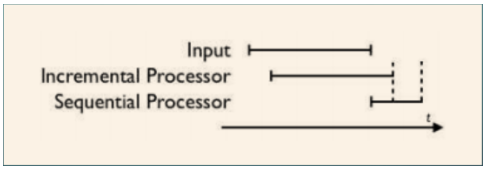


Figure 2.1: Input processed by Incremental and Sequential Processor. Even a much slower Incremental Processor can finish before a Sequential Processor [**?**]

In the InproTK [16] framework we can see that it does incremental processing where incremental systems consist of a network of processing *modules*. A typical module takes input from its *left buffer*, performs some kind of processing on that data, and places the processed result onto its *right buffer*. The data is packaged as the payload of *Incremental Units* (IUs) which are passed between modules. The IUs themselves are also interconnected via so-called *same level links* (SLL) and *grounded-in links* (GRIN), the former allowing the linking of IUs as a growing sequence, the latter allowing that sequence to convey what IUs directly affect it (refer to Fig 2.2 for an example). A complication particular to incremental processing is that modules can "change their mind" about what the best hypothesis is, in light of later information, thus IUs can be *added*, *revoked*, or *committed* to a network of IUs. InproTK determines how a module network is "connected" via an XML-formatted configuration file, which states module instantiations, including the connections between *left buffers* and *right buffers* of the various modules. Also part of the toolkit is a selection of "incremental processing-ready" modules, and so makes it possible to realize responsive speech-based systems.

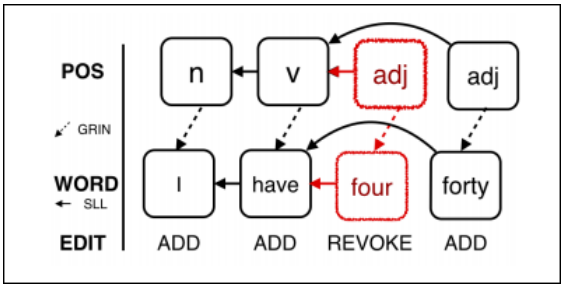


Figure 2.2: Example of IU network; part-of-speech tags are grounded into words, tags and words have same level links with left IU; *four* is revoked and replaced with *forty* [2]

An incremental framework like InproTK is essential because common ground isn't just there, ready to be exploited; people have to establish it with each person that they interact with. The first step in establishing either type of common ground is finding the right shared bases–the right evidence. For example, if a father and a son were at the beach, their joint gaze on a conch shell would be excellent evidence that each of them have that information that there is a conch shell between them but it is poor evidence that they each have the information that the shell is six years old [5]. The father could judge it highly likely that the conch shell is part of our common ground, but unlikely that its age is [3]. Shared bases vary in what H. H. Clark calls quality of evidence and the type of information they give rise to. People are fallible in these judgments and they know it. Fortunately, we have practical strategies in using language for preventing such discrepancies and repairing them when they arise [5]; e.g. asking each other questions to remove certain assumptions we had before the dialogue started, but later on turned out to be false. Therefore, when it comes to coordinating on a joint action, people cannot rely on just any information they have about each other. They must establish just the right piece of common ground, and that depends on them finding a shared basis for that piece. The shared basis is what Schelling called the key to the *coordination problem* and what Lewis called the *coordination device* [3].

It takes two people working together to perform joint activities e.g., play a duet, shake hands, play chess or waltz. To succeed, the two of them have to coordinate both the content and process of what they are doing for example, Alan and Barbara on the piano, must come to play the same Mozart duet [5]. This is *coordination of content*. They must also synchronize their entrances and exits, coordinate how

loudly to play forte and pianissimo, and otherwise adjust to each other's tempo and dynamics. This is *coordination of process*. They cannot even begin to coordinate on content without assuming the existence of common ground. To coordinate on process, they need to update their common ground moment by moment considering all collective actions are built on it.

What does it take to contribute to conversation? For example, suppose Alan utters to Barbara, "Do you and your husband have a car?" In the standard view of the speech acts [1], what Alan has done is ask Barbara whether she and her husband have a car, and, in this way, he has carried the conversation forward. But this isn't quite right. Consider this actual exchange:

Alan: Now, -um, do you and your husband have a j- car

Barbara: - have a car?

Alan: Yeah

Barbara: No -

Even though Alan has uttered "Do you and your husband have a car?", he hasn't managed to ask Barbara whether she and her husband have a car. We know this because Barbara indicated with "-have a car?", that she hasn't understood him (Actually, the word ask is ambiguous between 'utter an interrogative sentence' and 'succeed in getting the addressee to recognize that you want certain information.' You can say, Ken asked Julia, "Are you coming?" but failed to ask her whether she was coming because she could not hear him. We are using ask in the second sense here.) Only after Alan has answered her query (with "yeah") and she is willing to answer the original question ("no-") do the two of them apparently believe he has succeeded. Therefore, asking a question requires more than uttering an interrogative

sentence. It must also be established that the respondent has understood what the questioner meant [5].

We can see that the above observations are not true for human-robot dialog since Chatbots, PAs, SDSs do not share the same world representation as humans. Their representations of the shared world are misaligned with that of human beings because their design limits the number of communication channels to just one-the verbal channel or in some cases text whereas humans can perceive the world and communicate with others by making use of at least five channels (i.e. sense organs) each time they communicate. Hence, extracting information from what we perceive through our senses is very normal for us, but not always an option for current robots or DPAs and SDSs. Therefore, it is all the more important to establish common ground before participating in any type of joint activity so that with each dialog act, both participants come one step closer to the goal of their activity.

The work presented in [3] includes a dialogue system that attempts to mediate a shared perceptual basis between the human and the robot through automatic knowledge acquisition. As conversation proceeds, the robot first matches human descriptions to its internal representation of the shared world. It then automatically acquires and confirms through dialog, common ground knowledge about the shared environment. The acquired knowledge is used to enrich the robot's representation of the shared world. Their results have shown that an extra effort from the robot, to make its human partner aware of its internal representation of the shared world, contributes to better common ground. This was applied to a setting involving a human and the NAO robot. This kind of work is yet to be accomplished in situations where the interaction is between a chatbot or an SDS and a human.

Similar to our work, Julian Hough and David Schlangen presented a simple real-time, real-world grounding framework presented in [8], and a system which implements it in a simple robot, allowing investigation into different grounding strategies. However my approach is different because we are using a GUI to signal understanding while their emphasis was on the grounding effects of non-linguistic task-related actions. They experimented with a trade-off between the fluidity of the grounding mechanism with the *safety* of ensuring task success. They had a simple pick-up-and-place robot with uni-modal communication abilities, which is simply its manipulation behavior of objects. They used basic reinforcement learning to make the robot realize when it had made a mistake by deducting points for each mistake. This ensured that the robot at least asks for a clarification, the next time it is not sure what the next best action would be. Likewise, a positive reinforcement mechanism worked when it did the right thing and promoted it to do the right thing the next time it thought so. While this robot did not have Natural Language Generation (NLG) capabilities, its physical actions are first class citizens of the dialogue so it is capable of dialogic behavior through action.

Furthermore, in our preliminary work, a system called amBrOISEa [10] explored if a GUI helps in establishing common ground between the system and the user by signaling understanding. All the dialog acts were shown using the GUI i.e. Calrification Request (CRs), shown in 2.3 and feedback i.e. when the tree expands to denote confirmation of intent. Selection of a property as shown in Figures 2.3, 2.4, 2.5 and 2.6 was denoted by the red dot. The map on top zoomed into the areas that best match the intent and gave users the options to choose from, shown in Fig 2.6. The user can then pick and add the options from the suggestions pane to the

itinerary pane. Part of the task was to recreate their itinerary when done. Through this system we were only exploring incremental grounding as a preliminary step to see if visual feedback helps users better understand the system while interacting because users can see what they say. It did show us that users prefer incremental grounding over incorrect system action but we did not address the problem of lack of training data and long term memory in this work. Also, this was not the best type of GUI for demonstrating conversation grounding because the onus to clarify the utterance was on the user and not the system. Also, the system had no explicit feedback. We were not trying to solve the problem of lack of training data or memory.

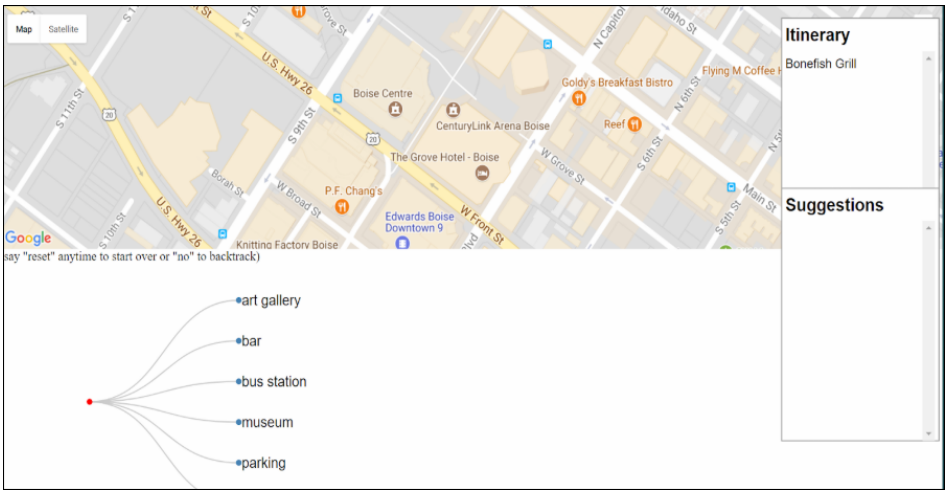


Figure 2.3: Expanding-right tree GUI used in amBrOISEa [10]

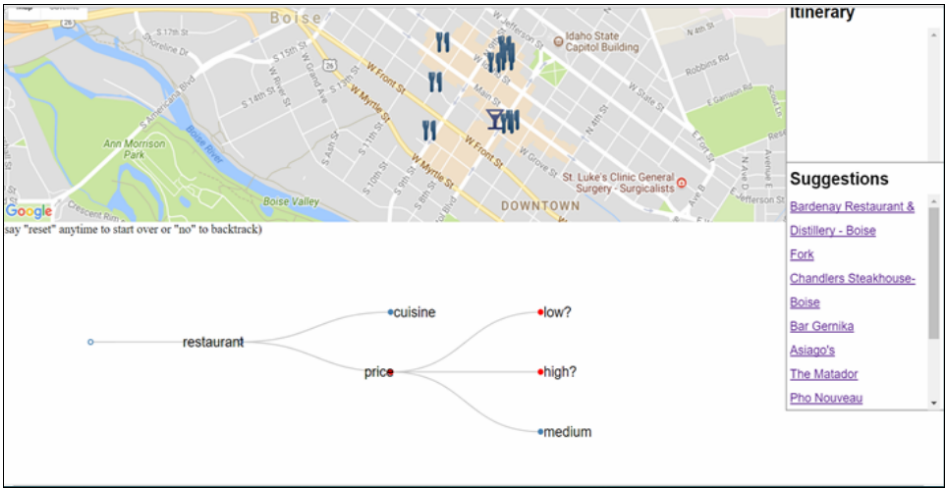


Figure 2.4: GUI depicting clarification through the *'?'* symbol. Values it is confused between are denoted by the red dot preceding them [10]

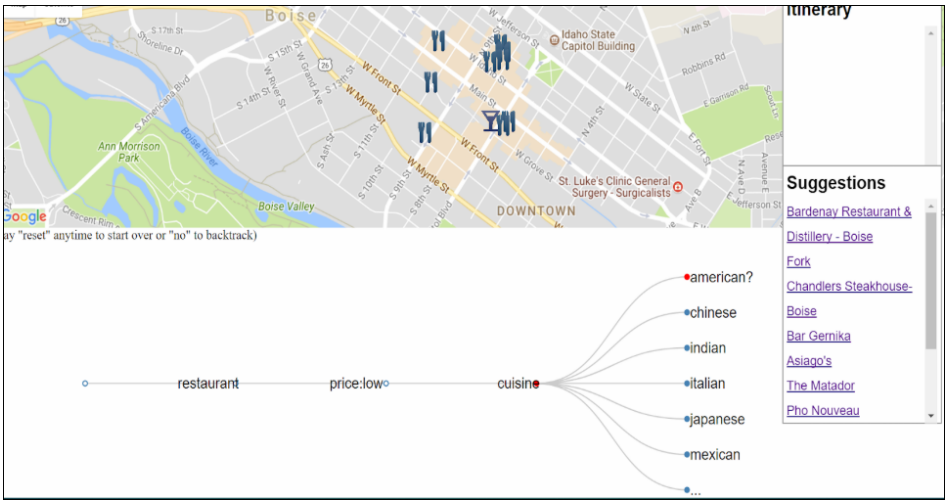


Figure 2.5: *Low* is selected as the intended value for *price* [10]

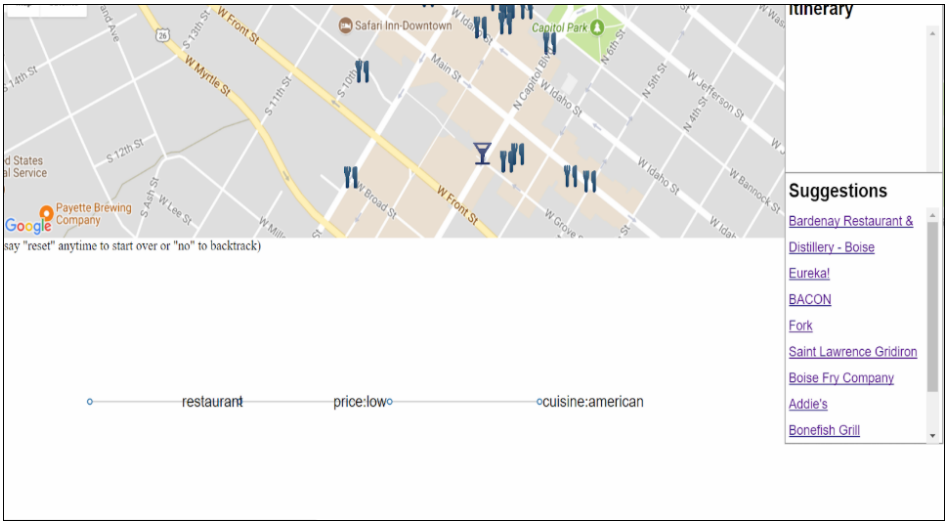


Figure 2.6: Depicting complete intent. Suggestions pane showing suggestions based on intent [10]

My thesis builds off of the amBrOISEa model considering results of [10] were positive overall: the system proved useful and allowed users to fill an itinerary using speech. Users were able to recreate their itineraries with the autonomous system much more accurately than with the baseline system. Minimal grounding indeed took place through the GUI by the tree and map, both of which updated incrementally as the users' utterances unfolded, by properties (i.e. ontology) pre-set in the system, and by improving the mapping between utterances and properties. This will be a starting point for the work here to autonomously improve the Natural Language Understanding (NLU) unit. This will involve a strong influence of grounding which will in turn, help in making the system understanding better. We go beyond that previous work by improving the GUI and by applying the incremental model of natural language understanding to a domain where remembering important facts is the primary purpose of the system. The difficulty lies in mapping from a surface

form utterance (i.e., a sequence of words) to isolate the important bits of information that need to be stored. Though we attempted to build a system that can work with minimal training data in previous work [10], the system was only able to interact with users for a couple of minutes and anything that was learned was then forgotten. In this work I will set up a system that continues to learn and improve over time–a system that not only remembers facts, but learns from past interactions and improves while needing little or no training data to begin with. In the next section we will demonstrate how that was accomplished by describing my model.

CHAPTER 3

OUR MODEL

Our proposed method lends itself to conversational grounding to know if the system and the user are able to understand each other while creating a shared common ground of knowledge to keep adding to, as the conversation unfolds. Applied to any DPA, SDS, or chatbot, we can bootstrap a system to work reasonably well with minimal or no training data, and, using a GUI to make up for the shortcomings of the system, we can signal ongoing understanding in an intuitive, graphical way and allow the user to correct system mistakes explicitly, while only minimally disrupting the interaction between the system and user. In this way, users annotate their input directly though it seems as if it's just an ordinary dialogue.

To accomplish this, our model relies on four modules combined with the incremental processing framework. This setup enables processing installments of the entire utterance where each word is considered an installment. This means that the processing of the entire utterance happens word by word even though the length of the complete utterance varies with each person's information entered by the user. The components of our system interact with each other to get intermediate results as they pass through different modules and do so incrementally every time a word is entered. This is a major advantage over non-incremental systems which makes

the system process faster while having provisions for the system to send clarification requests and other notifications as and when needed. Below we describe the modules and their functioning in greater detail.

3.1 Graphical User Interface

For *grounding* we will bootstrap this system by overcoming the shortcomings of the untrained NLU through buttons and clarification requests in the GUI, allowing the user to make the system understand intent in real-time and be able to make changes to it if required. The goal of the GUI is to intuitively inform the user about the internal state of the system's ongoing understanding with each turn. One motivation behind this is that the user can determine if the system understood that user's intent before providing the user with a response (e.g. a frame with past slots filled, or a clarification request that asks the last entry to the frame be resolved); If any misunderstanding takes place, it happens before the system considers it as part of the common ground in the ongoing dialog and is potentially more easily repaired, in an incremental way.

We hypothesize that the user experiences more natural communication as our incrementality provisions manifested in the interface adhere to the principle of least collaborative effort. People apparently don't like to work any harder than they have to, and in language this truism has been embodied in several principles of least effort. Grice [7] expressed this idea in terms of two maxims: Quantity - make your contribution as informative as required for the current purpose of the exchange, but do not make your contribution more informative than is required - and Manner - Be

brief (avoid unnecessary prolixity). According to this version, speakers are supposed to create what we will call proper utterances, one they believe will be readily and fully understood by their addressees.

Fig 3.1 shows us how the user interaction begins. They are provided an example utterance at the top that can be used as a reference throughout. At the bottom, there is an input box where the user types the utterance and with each white space encountered, the word before that space is processed.

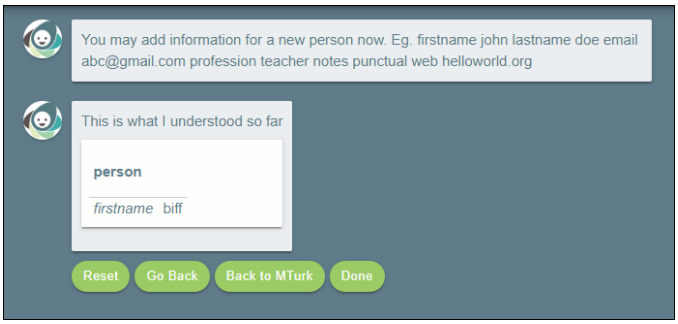


Figure 3.1: Chatbot's response

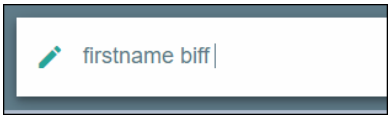


Figure 3.2: User utterance for Fig 3.1

We can see in Fig 3.3 and Fig 3.5 that the user entered values for *lastname* and *notes* respectively, and was shown the internal representation of the system in a user interpretable way i.e. a frame that shows the two slots *firstname* and *lastname* filled out. There are no CRs encountered till now as the system was fairly certain of what to do with words sent to it for processing. Moreover, now since the user entered

a word and that was understood as it is by the system and shown to the user, it is now part of the common ground that exists between them.

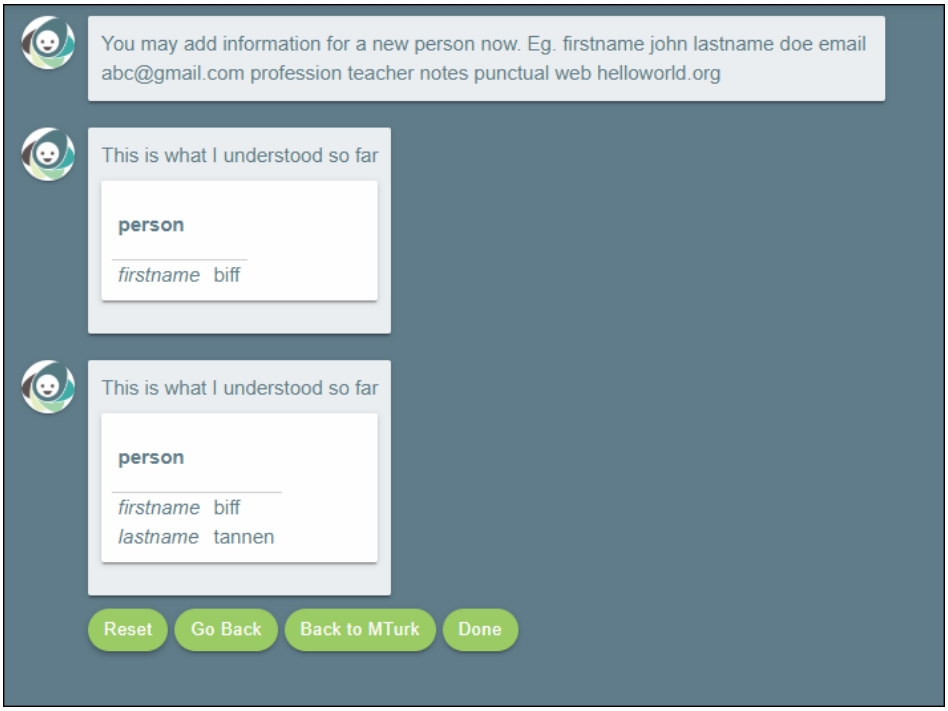


Figure 3.3: Chatbot's response

Figure 3.4: User utterance for Fig 3.3

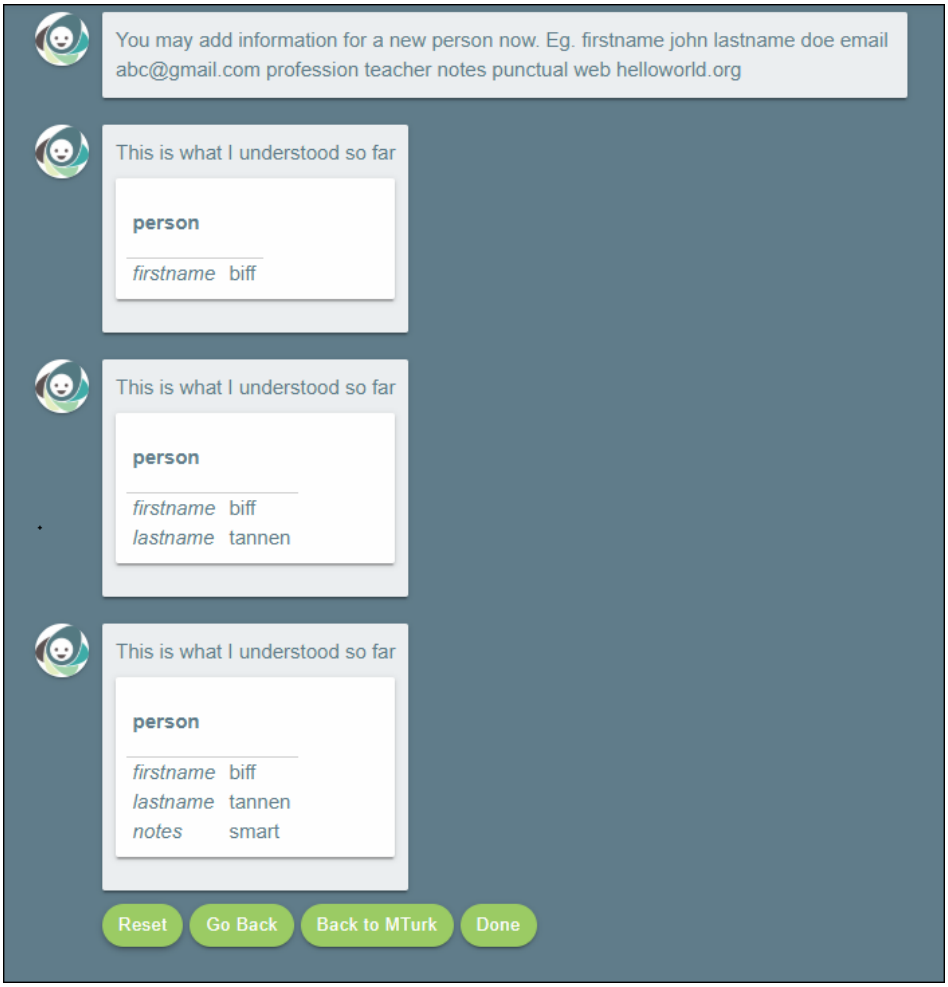


Figure 3.5: Chatbot's response

Figure 3.6: User utterance for Fig 3.5

In another scenario, the user could have entered something that was assigned a lower confidence value by the NLU and it could not reach the *select* threshold. That is when the system sends a CR to the user to get a confirmation on it's decision. At

this point the user can not type any other token value pairs until the CR is dealt with. This is done to avoid keeping CRs' resolution for later and prevent the system from making faulty assumptions that would give us incorrect training examples in the end. In the GUI we force the user to resolve CRs by disabling the input box until the clarification is received by the system, after which they can continue typing their utterance if they want to enter more information. Fig 3.7 illustrates CR

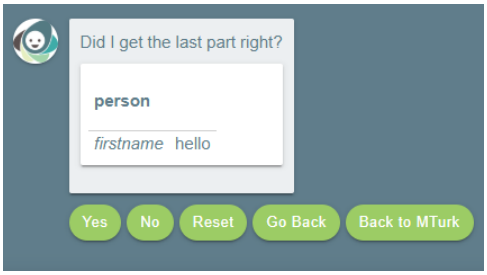


Figure 3.7: A Clarification Request (CR) for *firstname* slot

To respond to a CR the user has two choices- to respond with a *no* and re-enter the intended information for the latest token entered, shown in Fig 3.8 or to respond with a *yes* and have the system confirm the previously added slot value as the intended information indeed, shown in Fig 3.9. Upon getting the clarified entry, that slot's information is added to the common ground. In both these cases we chose to bypass the NLU as the result of clarification is the true intent and need not be processed.

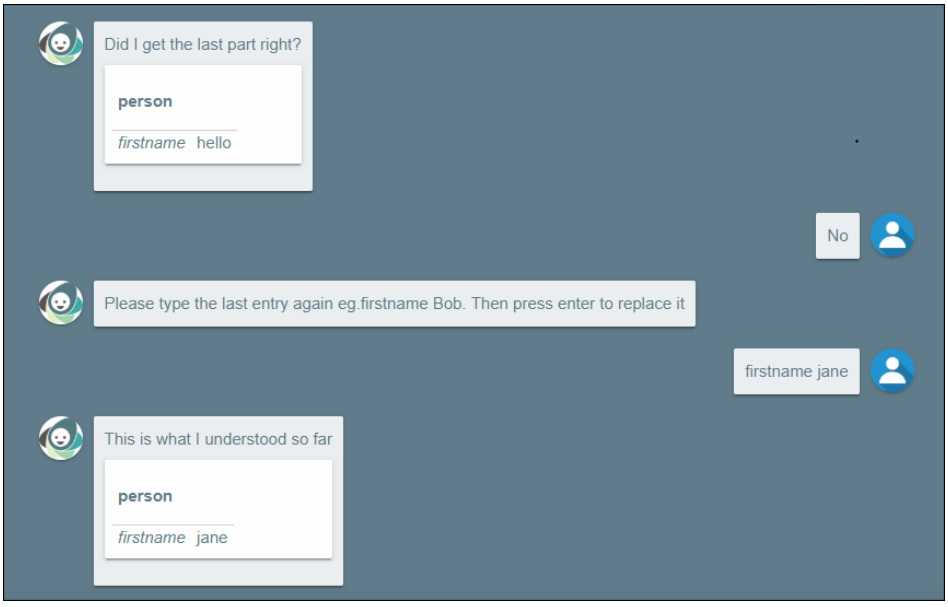


Figure 3.8: User responding *no* to a CR for the *firstname* token

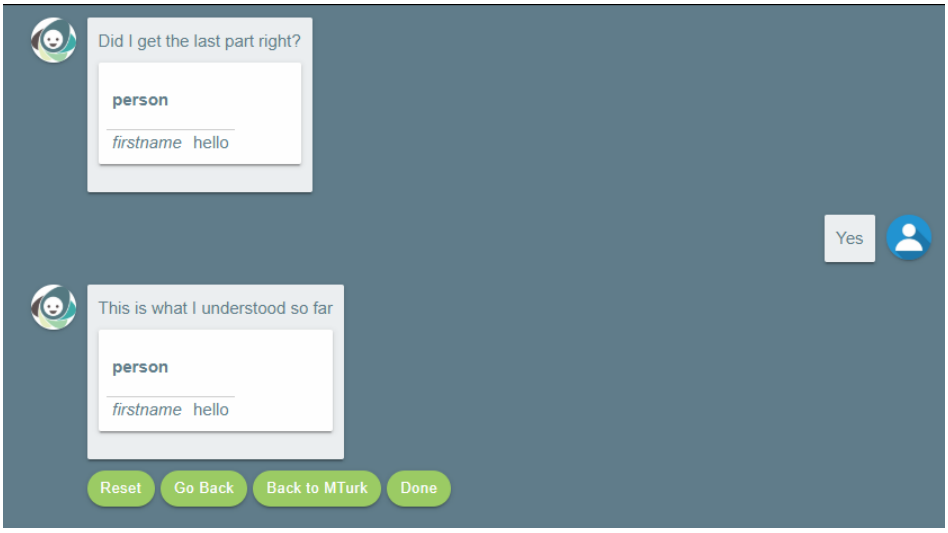


Figure 3.9: User responding *yes* to a CR for the *firstname* token

Irrespective of the response, the user is shown the system's understanding to denote what is part of the common ground after each CR. That's because there is

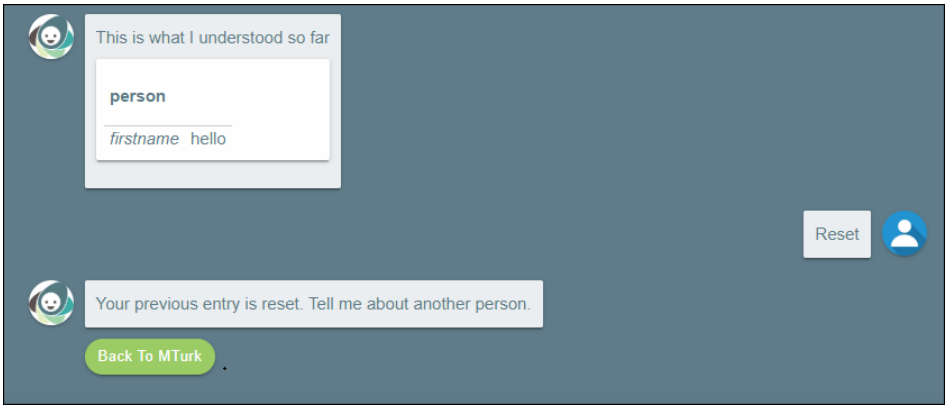


Figure 3.10: *Reset* button press response

a chance that the previous entry may not remain the same post-resolution, we show the filled frame till that point with the latest entry changed. This manifestation of the GUI, i.e. having *yes* and *no* button presses to respond is unique to CRs and in case of *Reset* or *Go Back* button presses, users are shown a message that tells them what the button press did and the action they need to perform next. *Reset*, removes the entire frame while *Go Back* removes the latest slot from the frame. The users can press *Go Back* multiple times to clear more than one slot if they wanted, shown in Fig 3.10 and Fig 3.11.

When the user presses the *Done* button, it marks the end of the utterance and the information entered for that person is considered complete. Now the user can begin to enter information for a new person.

The *Back To MTurk* button was created to manage Amazon Mechanical Turk (MTurk) submissions and it's working is explained in the data collection section.

The upcoming sections contain a detailed explanation of the NLU and DM, which will help in better understanding GUI behaviour within an incremental

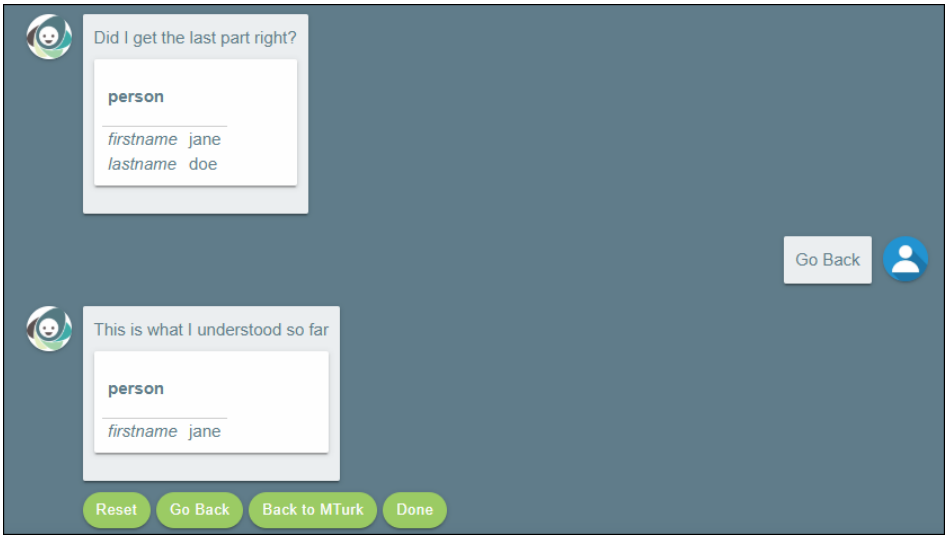


Figure 3.11: *Go Back* button press response

processing framework.

3.2 Natural Language Understanding

In amBrOISEa [10], we approached the task of NLU as a slot-filling task [9], where an intent is complete when all slots of a frame are filled was shown by a tree structure. Similarly with our chatbot, we see a frame like shown in Fig 4.8.

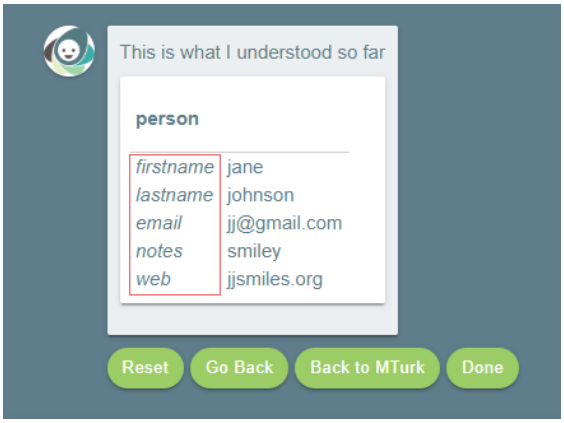


Figure 3.12: A picture of an entire frame, where the red rectangle shows the slots i.e. *firstname*, *lastname*, *email*, *notes* and *web* with the values that user filled them with

The main driver of the NLU will be the Simple Incremental Update Model (SIUM) which has been used in several systems that reported substantial results in various domains, languages, and tasks. Following Kennington and Schlangen (2016) [9], though originally a model of reference resolution, it was always intended to be used for general NLU. The model is formalized as follows:

$$P(I|U) = \frac{1}{P(U)} * P(I) \sum_{r \epsilon R} P(U|R = r) P(R = r|I) \tag{3.1}$$

That is, $P(I|U)$ is the probability of the intent I (i.e., a frame slot) behind the speaker's (ongoing) utterance U. This is recovered using the mediating variable R, a set of properties which map between aspects of U and aspects of I. We used this model in [10] to get us started with no training data. Also, SIUM assigns probability mass to words that are similar to properties, so it can work with little or no taining data. Based on the usual contact list information usually found in address books,

we opted for six pieces of information (e.g. The frame for person can be filled by concepts such as firstname with a value say, biff where the concept has properties like firstname, first, name, fname etc. Properties are pre-defined by a system designer and can match words that might be uttered to describe the concept in question. For $P(R|I)$, probability is distributed uniformly over all properties that a given intent is specified to have. If other information is available, more informative priors could be used as well.) The mapping between properties and aspects of U can be learned from data. During application, R is marginalized over, resulting in a distribution over possible concepts. This occurs at each word increment, where the distribution from the previous increment is combined via P(I), keeping track of the distribution over time. For the purpose of the equation, concept is equivalent to intent.

When the current word gets a low confidence value assigned to it, the system sends out a CR that the user sees and has to respond to first. This new entry bypasses the NLU and is selected as the intent because it is now clarified.

3.3 Dialogue Management

The DM plays a crucial role in the DS [9]: as well as determining how to act, the DM is called upon to decide when to act, effectively giving the DM the control over timing of actions. The DM policy is based on a confidence score derived from the NLU (in this case we used the distribution's argmax value) using thresholds for the actions set by hand (i.e. trial and error) deemed suitable with our experience during testing the application with test data. At each word and resulting distribution from NLU, the DM needs to choose one of the following:

- *wait* - wait for more information in order to make sense (i.e. for the next word.) Confidence value lower than 0.4.
- *select* - as the NLU is confident enough, fill the slot with the argmax from NLU. Confidence value higher than 0.6.
- *request* - signal a CR on the current slot and the proposed value. Confidence value between 0.4-0.6.

The system uses OpenDial here as an IU-model to perform the task of the DM and gives us the opportunity to adjust the values through reinforcement learning. The DM processes make a decision for each slot, with the assumption that only one slot out of all that are processed will result in a non-*wait* action (though this is not enforced).

The system in [10] improves upon previous work by leveraging the GUI to learn as it interacts. We accomplished this by collecting the words of a completed utterance and corresponding filled slots then informing the NLU that the utterance led to the filled slots-effectively providing an additional positive training example for the NLU. The NLU can then improve its probabilistic mapping between words and slot values; i.e., through updating the sub-model $P(U|R)$ by retraining the classifier with new information. This is a useful addition because the system designer could not possibly know beforehand all the possible utterances and corresponding intents for all users; this effectively allows the system to begin from scratch with little or no training data. In the system we propose here, improving NLU and the DM will allow us to train the model from the data annotated by the user and also help in adjusting the thresholds on which the DM acts, which in combination with grounding that adds to the long-term memory component.

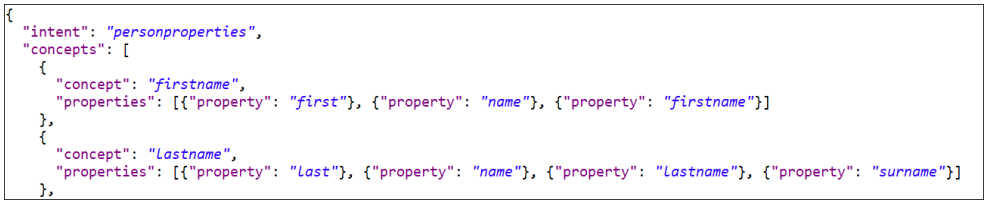
3.4 System Description

For our work, we utilized the InproTK toolkit, written in Java that does this for us. It is like the oil that keeps the systems running whenever dropped. At the time the user finishes entering a word–denoted by entering a space after a sequence of letters–everything before that space is processed. We chose a word as our incremental unit because of the kind of task we are trying to accomplish i.e. for each person enter values for at most six tokens of information. Once we have the incremental unit, the processing starts with sending that unit to the next module i.e. NLU.

Our system has a Java server (named conv-server1 on heroku) where the above explained components, along with their interactions are present. This is the biggest piece of the project. The next big piece is the front-end (names conv-augi on heroku) written in Javascript and served through NodeJS. The front-end and back-end are connected to each other via web sockets which facilitates full-duplex communication over a single connection.

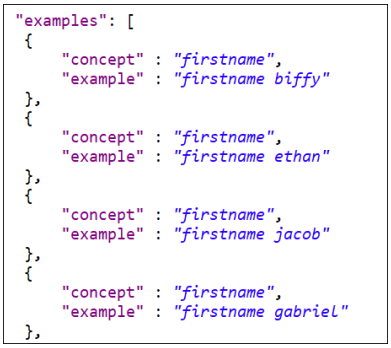
In conv-server1, we provide the structure –*a frame*– that will contain each person's information. This frame is setup as a slot-filling task for the user where each slot is a token of information for that person that is filled by it's value. This is part of our ontology which has one *concept* that belongs to *intent* for which information is to be entered by the user. This intent is named `personproperties`, refer to Fig 3.13. Also here, we define the *concepts* which represent the basic information that can be provided by the user for each person such as: first name (given name), last name (family name), email (email ID), profession (their occupation), web (their website) and notes (an adjective that helps you remember them by). Each concept has its own

set of at least three *properties* which are synonyms of the concepts. This is done to ensure that even if the user enters a different word that denotes the same concept eg. *occupation* instead of *profession*, the system will still be able to correctly map input to the *concept* and not treat it as invalid and wait. Lastly, we added sixteen *examples* for each concept to train the model the first time. These examples are in the JSON format as shown in Fig 3.14 and this is where we add more examples of each concept when we get data after each user interaction.



```
{
  "intent": "personproperties",
  "concepts": [
    {
      "concept": "firstname",
      "properties": [{"property": "first"}, {"property": "name"}, {"property": "firstname"}]
    },
    {
      "concept": "lastname",
      "properties": [{"property": "last"}, {"property": "name"}, {"property": "lastname"}, {"property": "surname"}]
    },
```

Figure 3.13: Depicting *intent*, *concept* and *properties*



```
"examples": [
 {
     "concept" : "firstname",
     "example" : "firstname biffy"
 },
 {
     "concept" : "firstname",
     "example" : "firstname ethan"
 },
 {
     "concept" : "firstname",
     "example" : "firstname jacob"
 },
 {
     "concept" : "firstname",
     "example" : "firstname gabriel"
 },
```

Figure 3.14: Sample of training examples

As soon as the web sockets connection is established, a new instance for the user is created and the session information is stored in a hashmap. Now, the user can start sending and receiving information from the server without waiting for a long time.

There are two channels for processing the input, one is incremental- utilized for each word and the other is- sequential that is utilized at the end of the utterance to process the last IU of the utterance to submit and completely process the entire utterance before sending anything back to the client. Each time the user enters a concept with the information associated with it, it is stored in a `linkedHashmap` where the concept is the key and the value associated with it is stored inside a `FrameForDisplay` object along with a flag for stating if it is in the clarify state or not. Each time there is a CR, that entry is marked until resolved, in case of a 'yes' the flag value is changed to false and no other change happens while in case of 'no' the value inside `FrameForDisplay` is changed to the new one that came from 'no' and the flag is changed to false denoting that the update was made. When the user clicks 'Done,' the contents of the linkedHashmap are displayed as a frame that's now considered part of their common ground.

At the end of every person's information, the user is prompted for feedback. Fig 3.15 shows us the feedback frame, where they can click on any of the smiley faces which best describes their experience. We predict that the number of happy faces will increase with each successive iteration.

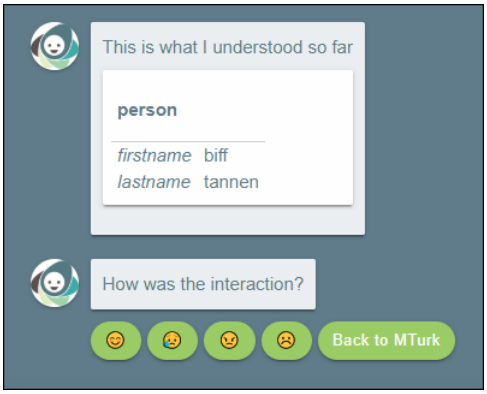


Figure 3.15: System asking for feedback

The next section describes our data collection process, experiment performed and the evaluation.

CHAPTER 4

DATA COLLECTION

The entire system was hosted on Heroku which enables online, concurrent user access for multiple turkers at the same time. Fig 4.1 depicts the data collection infrastructure. The entire system was hosted as a combination of three separate apps - *conv-consent* for taking the turkers' consent, *conv-augi* for hosting the interface and *conv-server* for hosting the server. The database was a Heroku add on–a feature that comes as part of Heroku apps–utilized to collect the turkers' data. After successfully deploying the system, we used the Heroku-generated Uniform Resource Locator (URL) to publish Human Intelligence Tasks (HITs) on Amazon Mechanical Turk (Mturk).

MTurk It is a crowd sourcing marketplace that lets *turkers* (MTurk workers) perform data collection tasks for *requesters* through external HITs. For each HIT the requester is required to set duration and expiration of the task, maximum assignments and payment for each task among other settings. When the turkers see our HIT listed on Mturk, they consent to our terms and instructions first, then start interacting with our chatbot and submit their HIT *assignment* when done.

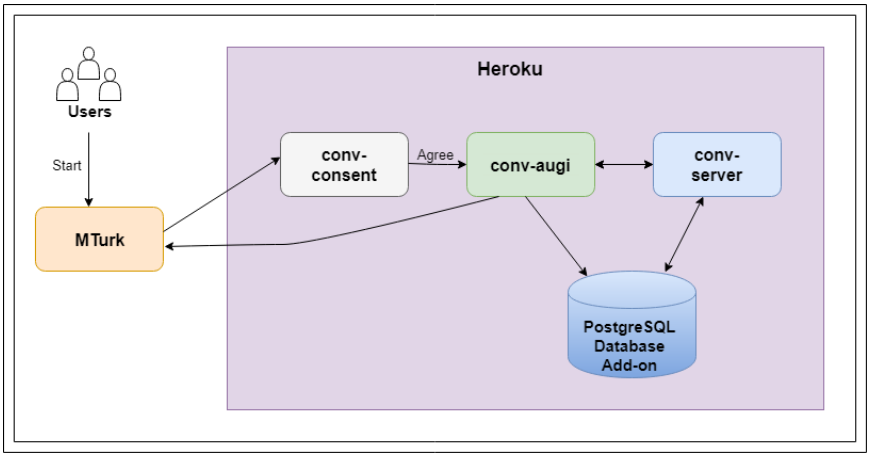


Figure 4.1: Overview of data collection setup

With the collected data we retrained our model after each of four iterations. Each *iteration* is defined as the entire exercise where we publish HITs for turkers to interact with while closely monitoring the data. Once the HIT times out, query all the data from the database, run scripts to convert it to JSON format, add these training to our system's ontology, retrain the NLU model and push the changes back to *conv-server*. Thus being able to publish another batch of HITs for the next iteration which contains a retrained model.

Our data analysis is best seen through these four iterations where each iteration contains all the interaction data from three novel turkers. In the next section we discuss the experiment followed by a brief description of the data collected followed by evaluation of data in the final section.

4.1 Our Experiment

4.1.1 Task

Each user was to imagine that they were the career center head of a renowned university attending a gala to network with people so that they can invite those people to attend their own university wide Career Fair at a later date. While networking at the gala, they should use Conv-augi (our system) to maintain a contact list. The information they enter for each person can be categorized under these six *tokens* (NLU slots) in any order or combination: *firstname* for First name, *lastname* for Last name, *email* for Email, *web* for their Website, *profession* for Work and *notes* for Notes. Valid input would contain token along with the corresponding information for each.

4.1.2 Performing the Experiment

In the entire duration of each turker's interaction with Conv-augi they were disallowed to enter details of people they knew existed e.g. celebrities. An example of their utterance would be, *firstname* biff *lastname* tannen *profession* teacher *email* asteacher@gmail.com *web* aster.org *notes* funny. They can even enter information in any order or combination for example for the same utterance turkers could enter it this way: *lastname* tannen *firstname* biff *email* asteacher@gmail.com *profession* teacher *web* aster.org *notes* funny.

Each turker was to enter no less than twenty-five people's information and at least three token-value pairs for each. Since MTurk allows users to exit the HIT

without submitting data, some turkers entered lesser data resulting in inconsistent number of values for some of the training data. The data we received is summarized in Table 4.1 and a detailed description follows in the next section.

Iteration	Happy	Sad	Angry	dissatisfied	number of CRs	Training Data Instances	Cumulative Training Data
1	75	1	3	3	113	78	94
2	71	1	2	9	255	93	187
3	70	2	0	6	96	78	265
4	122	3	4	5	207	138	403

Table 4.1: Data collected over four *iterations*

4.1.3 Data Description

In each iteration, turkers entered contact information for twenty-five people. After submitting each person's information to the system, the user was asked to rate their experience of entering that person's information. For example in Fig 4.2 we can see this person's information consists of firstname and lastname after which the system asks them to rate that experience which they can do by clicking one of the four smileys - happy, dissatisfied, angry and sad.

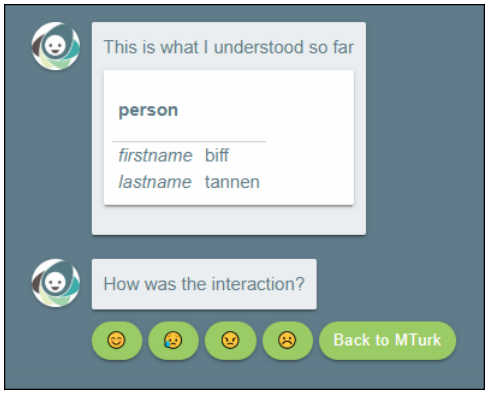


Figure 4.2: Rating each contact information entered

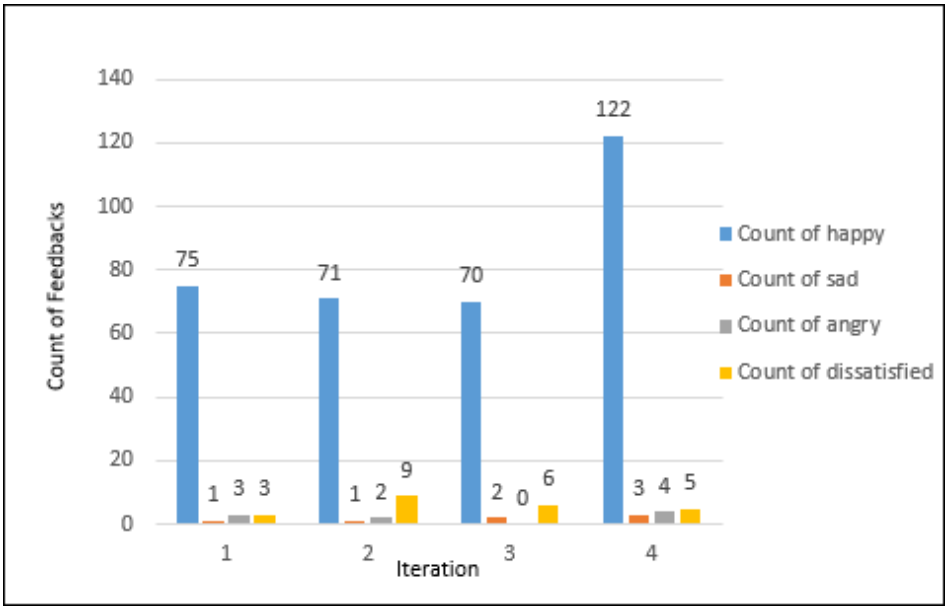


Figure 4.3: Sum of happy, sad, angry and dissatisfied feedback faces per iteration

The CRs are shown in 3.7 where the user has the option to respond in *yes* or *no* in case clarification is required. The number of training data instances is the highest sum of values out of all the tokens i.e. if in an iteration, firstname has the highest number of values entered by the turker, then that is the value of the training data instance. This was selected as the criteria to ensure no CRs are left out of the evaluation. Cumulative training data is the sum of training data examples before and after an iteration. The next section focuses on the evaluation of this data.

4.1.4 Evaluation

We were focused on finding out whether - (1) the system and user were able to establish common ground, (2) is the system learning and getting better with each iteration and (3) do number of CRs decrease with each iteration. Ideally, the system should have improved its ability to interact with users by familiarizing itself with various new examples obtained as a result of establishing common ground.

Fig 4.4 shows the number of each smiley rating per iteration. Another related graph is shown in Fig 4.5 where we compare the number of training examples added with every iteration to the number of happy faces received in the rating. With the help of these two graphs we can conclude that user affinity with the system increases with an increase in training data and the happiness of users increases. It also shows us that the system is grounding well with users because their interactions result in many more happy smiley ratings compared to the other smileys in each iteration.

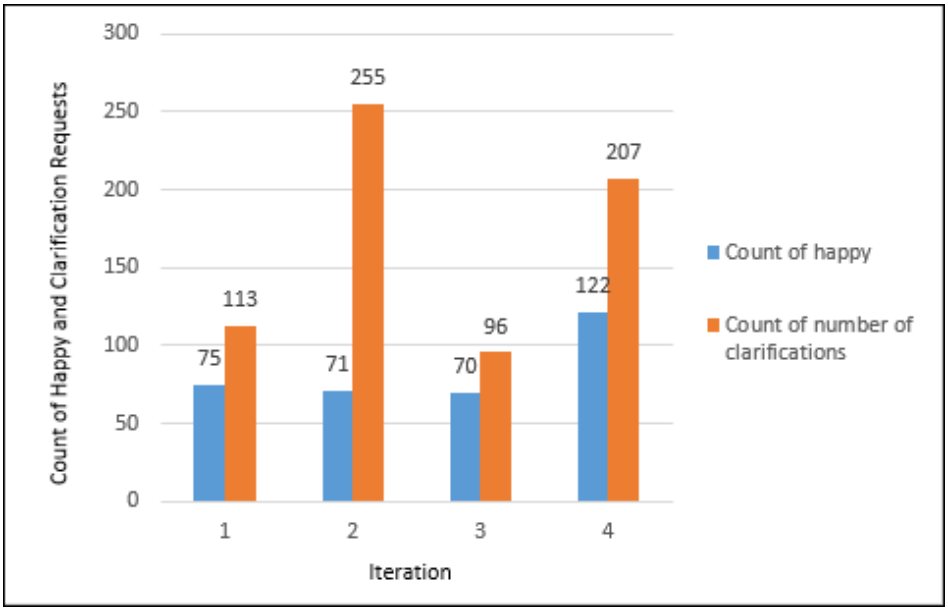


Figure 4.4: Number of CRs versus the happy faces in each iteration

We compared the number of CRs over the iterations and the results we got are shown in Fig 4.6. We did not expect a dip in the number of CRs in between iterations, but more of a gradual decrease over the four iterations. Now, when comparing the total number of CRs with the number of happy faces, shown in Fig 4.5, we can say that the users like the system in general. The correlation coefficient

between training data and count of happy is 0.957882944. This is another indication of common ground being established because with more training data the users are happy. That means the system is able to show its understanding to their users and is learning with added data. Fig 4.4 shows that there even if there are CRs, keeping in mind they are all new users we can say that there is learning due to the fact that there is common ground which allows feedback and CRs but even then they are happy as the system does the right system action.

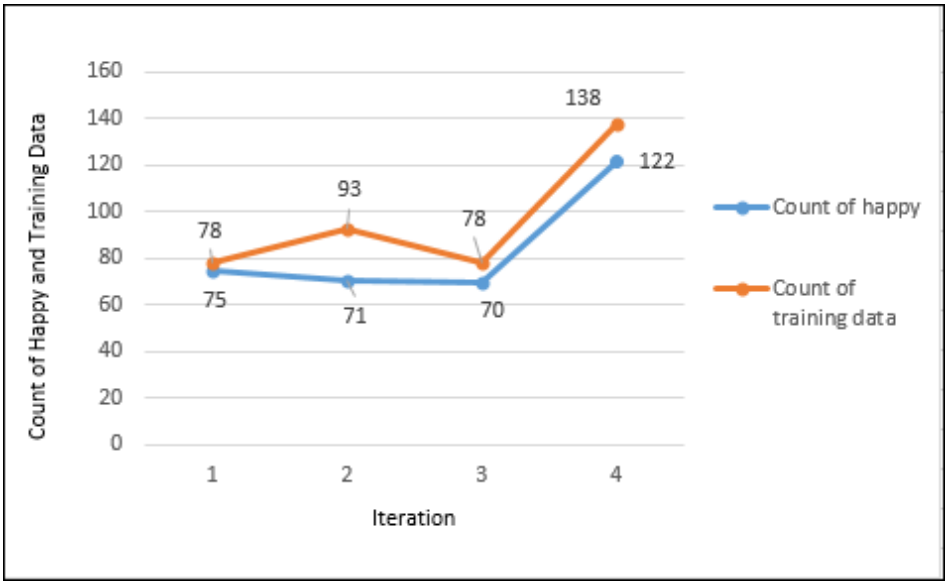


Figure 4.5: Number of happy feedback for the amount of training data per iteration

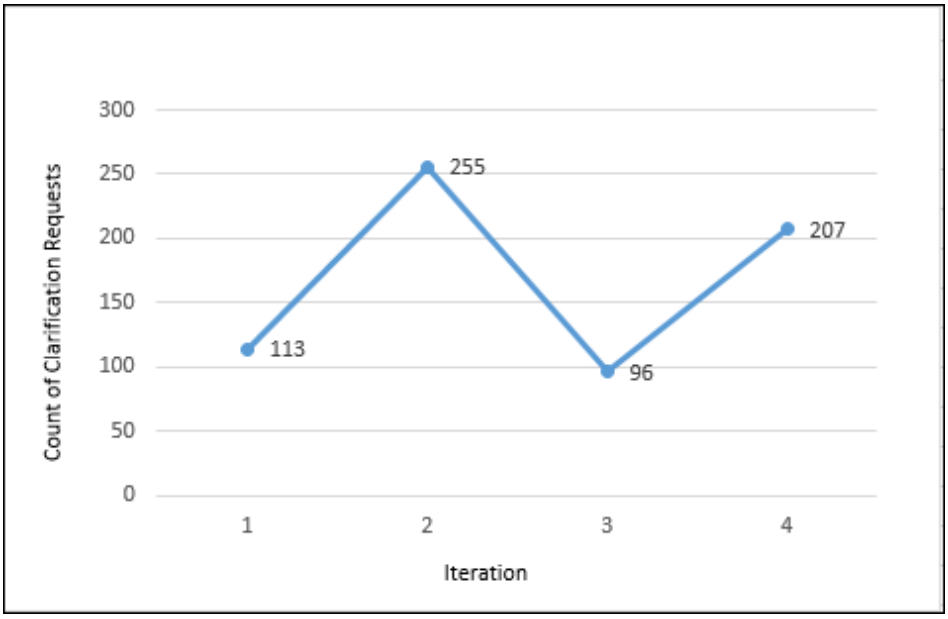


Figure 4.6: Total number of CRs per iteration

In Fig 4.7 we see the total number of training examples with each iteration. There is a high increase in the number of training examples with every iteration. That means we are adding a significant number of training examples for each successive iteration that the system can learn from. We had anticipated that this ratio of happiness versus number of interactions (i.e., collected data) would show us a point where increasing the number of interactions is no longer improving the system. That point would indicate the amount of data needed to train the model that would work well for any subsequent new user as well.

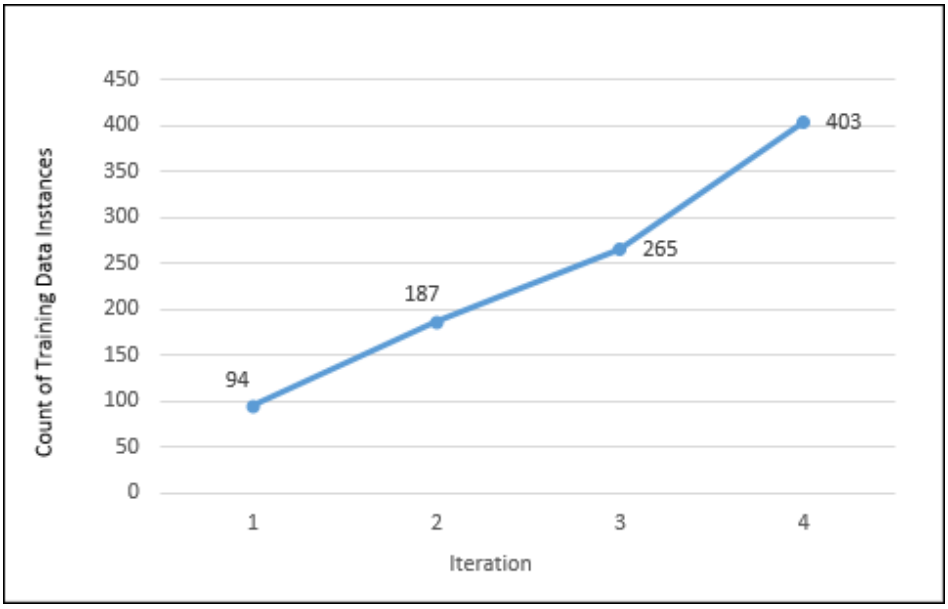


Figure 4.7: Total number of training examples per iteration

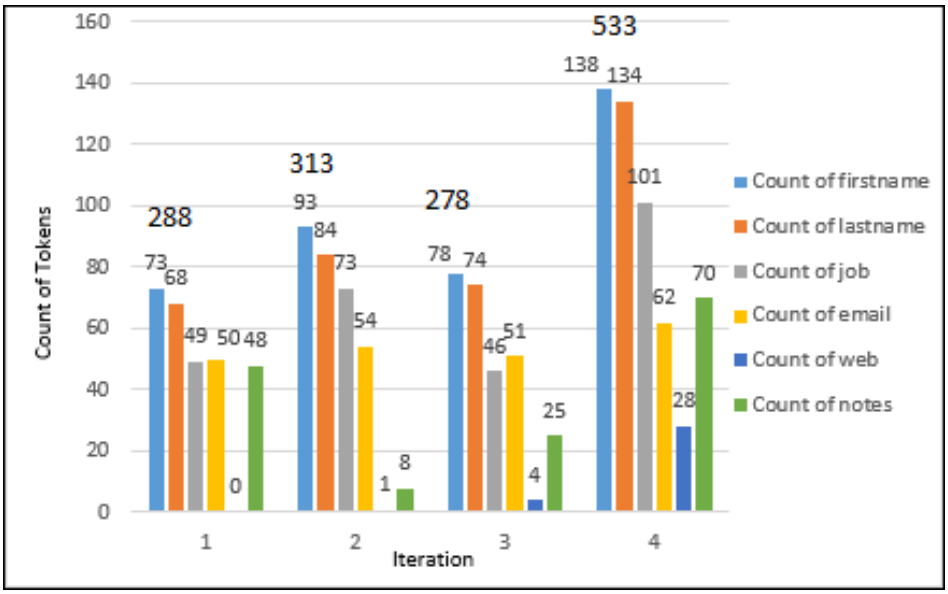


Figure 4.8: Total number of tokens (slots) filled by the user with total counts at the top

Finally Fig 4.8 shows the number of tokens (slots) turkers entered in each iteration. This increased for the first two iterations and the fourth iteration. What we can derive is that since turkers were able to add more information with each successive iteration, in the same amount of allotted time to do the task, the system is therefore learning a mapping from the training examples we are giving it and improving over each iteration.

Discussion Taken together, the results tell an informative story: the feedback (i.e., second aspect of grounding) and clarification (i.e., third aspect of grounding) mechanisms of our system enabled users to effectively signal that the system's understanding was indeed correct for that particular interaction. This resulted in labeled training data that we used to improve the knowledge of language (i.e., the first aspect of grounding). Importantly, the incremental nature of the feedback and CR mechanisms allowed us to assume that, at the end of the user-typed utterance, the understanding was complete and correct. A non-incremental version would have required an explicit and unnatural system turn asking "is this information correct?" which we wanted to avoid.

Though we cannot state with certainty the intent behind the user when they select one of the emojis, we can say that, overall, the fact that the happy emjois correlates highly with the amount of training data tells us that the increased knowledge of language resulted in increased "overall" user satisfaction with the system. We can see that this happened even though each iteration had novel users who had never before interacted with the system, so we can safely assume that they were not just getting used to the interface and task.

In the next section, we discuss the conclusion of our work and shed some light on possible future work that this work will be helpful in.

CHAPTER 5

CONCLUSION AND FUTURE WORK

The evaluation of our results shows that our initial assumption that if the turker is not making any changes to their input before submitting the utterance to the system, the system provides incrementally shown feedback and has provision for CRs showing that the user and system interactions are correct. Since they are correct and we trained our model using those interactions as training examples, and each iteration has novel turkers using our app, we can say that the system is getting better with each user interaction.

We can also say that the users like a system that asks CRs for an ongoing utterance even if the CRs come up multiple times in an utterance and they are not annoyed by it since they rate the system happy irrespective of the number of CRs received.

This work also shows that our DPA with a chatbot interface successfully establishes common ground during human-robot dialog. Consequently, it can ask for feedback and clarifications thereby letting the user correctly annotate all the interactions while having a sense of control during the entire interaction with our interface. On the other hand, the DPA is able to learn and improve over time by

utilizing those correct system interactions as training examples.

Incremental processing is the key concept for facilitating the creation and maintenance of this shared basis between dialog participants.

From the user perspective, they like interacting with the interface in general, we can see that with increase in the training data, there is also an increase in user satisfaction with the system. There is a constant increase in the number of happy faces with each iteration that indicates a higher level of performance of the system with each iteration because more data is being added with each subsequent iteration.

CRs were considered a good mechanism by users when they had entered incorrect information without themselves realizing it while, it was considered annoying when the CR was raised for a correct slot value. Therefore, users dislike false positives.

To conclude, we can say for sure that incremental processing is the answer to solving the grounding problem, which in turn solves the training data problem by providing user with feedback and asking for CRs that gives us correct training examples, that are annotated to serve long term memory by querying what the user wants to recall.

5.1 Future Work

This research is an ongoing process and there is scope for improvement. One of them is to do an A/B test between our system i.e. an incremental one with a non-incremental one. There can also be a questionnaire that the users can fill out

as part of their study giving us more data on how natural was their experience. In general, we can improve the task do a more detailed user study that gives us additional notes about the GUI, interaction etc.

Research on whether the DM thresholds that are, as of now, manually set should be changed in value to accommodate more *selects* or should we, based on the number of CRs sent or the happy rating received, let the thresholds set dynamically which would require a more detailed user study. Another research arena would be to model more than one concept i.e. intent at one time. Right now users are only allowed to add information for person, that can be modified to add more concepts like company, restaurants, schools etc. along with it.

We can also try and find out if the same concept when applied to users who are kids instead of adults results in greater, equal or lesser likability towards the interface and if common ground is easily established because it hard to know children's intent from an utterance that doesn't map to the intent very well. Another point here is to look into this system's utility with kids and adults that have communication or learning disorders.

Since DS and conversational grounding is a rapidly advancing area of Computer Science, Psychology and Data Science there will be a lot of scope for future work with any new advancements made in these fields.

REFERENCES

[1] Kent Bach. Meaning, Speech Acts, and Communication. *Basic Topics in the Philosophy of Language*, pages 1–23, 1994.

[2] David Schlangen Casey Kennington, Spyros Kousidis. InproTKs: A Toolkit for Incremental Situated Processing. June 2014.

[3] Joyce Y Chai, Lanbo She, Rui Fang, Spencer Ottarson, Cody Littley, Changsong Liu, and Kenneth Hanson. Collaborative effort towards common ground in situated human-robot dialogue. In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*, pages 33–40, Bielefeld, Germany, 2014.

[4] Brennan Susan E. Clark, Herbert H. *Grounding in communication.* American Psychological Association, 1991.

[5] Herbert H Clark. *Using Language*. Cambridge University Press, 1996.

[6] Coventy. *Spatial Dialogue between Partners with mismatched abilities*. Oxford University Press, april 2009.

[7] Paul Grice. *Studies in the Way of Words*. Harvard University Press, 1989.

[8] Julian Hough and David Schlangen. Investigating Fluidity for Human-Robot Interaction with Real-time , Real-world Grounding Strategies. (September):288–298, 2016.

[9] Casey Kennington and David Schlangen. Supporting Spoken Assistant Systems with a Graphical User Interface that Signals Incremental Understanding and Prediction State. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 242–251, Los Angeles, sep 2016. Association for Computational Linguistics.

[10] Casey Kennington and Aprajita Shukla. A Graphical Digital Personal Assistant that Grounds and Learns Autonomously. In *proceedings of Human-Agent Interaction*, Bielefeld, Germany, Oct 2017. Association for Computing Machinery.

[11] Kenny R. Coventy, Thora Tenbrink and John Bateman. Definite reference and mutual knowledge. *Elements of discourse understanding*, 2009.

[12] Dan Kondratyuk and Casey Kennington. Towards a Dialogue System with Long-term, Episodic Memory. In *Proceedings of SEMantics and Dialogue*, Saarbrücken, Germany, 2017.

[13] Pierre Lison. Structured probabilistic modelling for dialogue management. 2014.

[14] Kepa Rodriguez and David Schlangen. Form, intonation and function of clarification requests in German task oriented spoken dialogues. In *Proceedings of Catalog*, volume 4, pages 101–108. Citeseer, 2004.

[15] David Schlangen and Gabriel Skantze. A General, Abstract Model of Incremental Dialogue Processing. In *Dialogue & Discourse*, volume 2, pages 83–111, 2011.

[16] David Schlangen Timo Bauman. The INPROTK 2012 release: A toolkit for incremental spoken dialogue processing. Braunschweig, Germany, Jan 2012. VDE e.V. (German: Verband der Elektrotechnik, Elektronik und Informationstechnik).

[17] Sina ZarrieB, Julian Hough, Casey Kennington, Ramesh Manuvinakurike, David DeVault, Raquel Fernndez, and David Schlangen. PentoRef: A Corpus of Spoken References in Task-oriented Dialogues. In *10th edition of the Language Resources and Evaluation Conference*, Portoro, Slovenia, May 2016. ELRA.