

INSTITUT FÜR INFORMATIK
FAKULTÄT FÜR MATHEMATIK, PHYSIK UND INFORMATIK
UNIVERSITÄT BAYREUTH

MASTER THESIS

Task-oriented dialogue system with social capabilities

Aufgabenorientiertes Dialogsystem mit sozialen Fähigkeiten

Vivek Kumar Mahto, B.Sc.



- | | |
|----------------|---------------------------|
| 1. Supervisor: | Prof. Dr. Dominik Henrich |
| 2. Supervisor: | Sascha Sucker, M.Sc. |

October 2, 2023

Abstract

In recent years, there have been notable advancements in task-oriented dialogue systems designed for robotic tasks. Depending on the nature of the task, users can issue instructions to robots, guiding them effectively to accomplish the designated task. However, a fundamental constraint inherent in traditional task-oriented dialogue systems pertains to their limited social awareness. This limitation often results in interactions that are perceived as impersonal and lacking in emotional engagement. This thesis proposes an innovative approach to augmenting task-oriented dialogue systems for robotic tasks with social capabilities, aiming to create more natural, engaging, and empathetic human-machine interactions.

The main objective revolves around designing a framework of a dialogue pipeline for robotic tasks that has the capability of handling social interactions. For this a novel architecture is proposed, designed to seamlessly integrate task-oriented functionalities with non-task-oriented exchanges. This proposed framework relies on the exploration of diverse natural language processing techniques, large language models, and intent and domain recognition methods. The dialogue pipeline, built upon the aforementioned architecture, is capable of generating responses for both task-oriented and social utterances.

In order to facilitate user interaction, a web interface was provided for an intuitive platform for users to engage with the system. Additionally, this interface offers explanations of the generated responses to enhance user understanding and satisfaction. To assess the usability of the system, a user study is conducted, evaluating task effectiveness, efficiency, and user satisfaction. The outcomes of this evaluation reveal that the implemented dialogue system successfully allows to accomplish tasks in an engaging manner, highlighting its potential to facilitate efficient and enjoyable user interactions.

Zusammenfassung

In den letzten Jahren hat es bemerkenswerte Fortschritte bei aufgabenorientierten Dialogsystemen für Roboteraufgaben gegeben. Je nach Art der Aufgabe können die Benutzer den Robotern Anweisungen erteilen und sie so effektiv zur Erfüllung der vorgesehenen Aufgabe anleiten. Eine grundlegende Einschränkung herkömmlicher aufgabenorientierter Dialogsysteme besteht jedoch in ihrem begrenzten sozialen Bewusstsein. Diese Einschränkung führt oft zu Interaktionen, die als unpersönlich und wenig emotional empfunden werden. In dieser Arbeit wird ein innovativer Ansatz zur Erweiterung aufgabenorientierter Dialogsysteme für Roboteraufgaben um soziale Fähigkeiten vorgeschlagen, um natürlichere, engagiertere und einfühlsamere Mensch-Maschine-Interaktionen zu schaffen.

Das Hauptziel besteht darin, einen Rahmen für eine Dialogpipeline für Roboter-aufgaben zu entwerfen, der in der Lage ist, soziale Interaktionen zu verarbeiten. Zu diesem Zweck wird eine neuartige Architektur vorgeschlagen, die eine nahtlose Integration aufgabenorientierter Funktionalitäten mit nicht aufgabenorientiertem Austausch ermöglicht. Der vorgeschlagene Rahmen stützt sich auf die Erforschung verschiedener natürlicher Sprachverarbeitungstechniken, großer Sprachmodelle und Methoden zur Erkennung von Absichten und Bereichen. Die Dialogpipeline, die auf der oben erwähnten Architektur aufbaut, ist in der Lage, Antworten sowohl für aufgabenorientierte als auch für soziale Äußerungen zu generieren.

Zur Erleichterung der Benutzerinteraktion wurde eine Weboberfläche bereitgestellt, die den Benutzern eine intuitive Plattform für den Umgang mit dem System bietet. Zusätzlich bietet diese Schnittstelle Erklärungen zu den generierten Antworten, um das Verständnis und die Zufriedenheit der Benutzer zu verbessern. Um die Benutzer-freundlichkeit des Systems zu bewerten, wurde eine Benutzerstudie durchgeführt, in der die Effektivität der Aufgaben, die Effizienz und die Zufriedenheit der Benutzer untersucht wurden. Die Ergebnisse dieser Evaluierung zeigen, dass das implemen-tierte Dialogsystem es erfolgreich ermöglicht, Aufgaben auf eine ansprechende Art und Weise zu bewältigen, was sein Potenzial unterstreicht, effiziente und angenehme Benutzerinteraktionen zu ermöglichen.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Vision	3
1.3	Task and Objectives	3
1.4	Chapter Overview	5
2	Basics and Literature Review	6
2.1	Basic Terminologies	6
2.2	State of the Art	7
2.3	Related work	9
2.3.1	History	10
2.3.2	Recent development	11
2.4	Research contribution	12
3	Dialogue Pipeline	14
3.1	Domain Classification	16
3.1.1	Existing approaches	16
3.1.2	Rule-based classification approach	17
3.2	Intent Classification	19
3.2.1	Intent in Task-oriented dialogues	20
3.2.2	Intent classification problem	21
3.2.3	Intent Classification Approaches	21
3.3	Slot Filling and Response generation	24
3.3.1	Existing approaches	25
3.3.2	Slot filling for task-oriented instructions	26
3.3.3	Response Generation	33
3.4	Open-domain Conversational AI	40
3.4.1	OpenAI	41
3.4.2	DialogGPT	41

3.4.3	GODEL	42
3.4.4	Blenderbot	43
3.5	Research scope	46
3.6	Chapter Summary	47
4	User Interface and Explainability	48
4.1	User Interface	48
4.2	Response Explainability	50
4.2.1	XAI in Dialogue systems	50
4.2.2	Explainability in VerbBot	52
4.2.3	Expert vs non-expert explainability	53
5	Evaluation	55
5.1	User Study	56
5.1.1	Overall System	56
5.1.2	Procedure of Study	57
5.2	Results	59
5.2.1	Domain Classification	59
5.2.2	Intent Classification and Slot-filling	61
5.2.3	Task Success: Effectiveness	63
5.2.4	Mental Effort: Efficiency	67
5.2.5	User Satisfaction	70
5.3	Discussion	72
5.3.1	Confounding Factors	72
5.3.2	Explainability	73
5.3.3	Advantages and Disadvantages	73
5.4	Chapter Summary	75
6	Conclusion	77
6.1	Summary	77
6.2	Discussion	79
6.3	Outlook	81
R	References	84
A	Additional documents	89

CHAPTER 1

Introduction

The integration of social capabilities into task-oriented dialogue systems presents a distinct advantage over traditional dialogue systems. In the following subchapters, we will delve into the specific merits of this integration, starting with the motivation (Subchapter 1.1). Additionally, we shall explore the envisioned future in which non-experts effectively utilize task-oriented dialogue systems (Subchapter 1.2). Subsequently, Subchapter 1.3 will refine the objectives and goals of this research, narrowing down its focus. Lastly, Subchapter 1.4 will provide a comprehensive overview of the content, and structure of the thesis.

1.1 Motivation

Dialogue systems have garnered widespread adoption across diverse domains, encompassing applications such as customer service chatbots (Figure 1.1), virtual assistants, home automation robots (Figure 1.2), non-playing characters in games, and conversational AI in educational settings. The pervasive presence of dialogue systems is evident in their ubiquity, as they seamlessly integrate into various facets of contemporary human-computer interaction paradigms.

Over the past decade, a notable shift has occurred within the scientific community, emphasizing the transformation of dialogue systems from their traditional task-oriented approach to a more conversational and socially oriented paradigm. This shift reflects a growing interest in augmenting dialogue systems to encompass human-like conversational qualities, steering away from solely task completion and toward more engaging and interactive interactions.

The rationale behind making dialogue systems social stems from several compelling motivations. Firstly, integrating social capabilities into dialogue systems aims to enhance user experience and satisfaction. By emulating human-like conversation, users

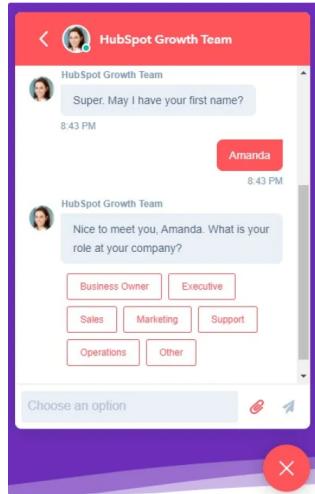


Figure 1.1: Customer service chatbot [Bernazzani22].



Figure 1.2: Home automation robot [Kamps22].

can engage with dialogue systems in a more natural and intuitive manner, fostering a sense of comfort and familiarity in their interactions. Human-like conversation can be described as the quality attribute of a dialogue system [Fellows22]. Secondly, social dialogue systems exhibit improved adaptability to individual user preferences and styles. Users have distinct communication preferences, and incorporating social elements allows the system to tailor responses accordingly, creating a personalized and enjoyable user experience. Third, dialogue systems with social features can better accommodate ambiguous language and handle diverse conversation topics. The ability to engage in small talk and handle informal language contributes to a more seamless and fluid conversation, even beyond the primary task at hand. Fourth, socially oriented dialogue systems are seen as more empathetic and emotionally responsive [Wezel21], potentially catering to users' emotional needs and building rapport. This aspect is particularly relevant in applications such as mental health support, where empathy and understanding are crucial in fostering a sense of support and comfort for users. Overall, the pursuit of making dialogue systems social is driven by the ambition to create more lifelike and dynamic human-machine interactions, therefore this thesis firstly designs the dialogue system to complete the task related to the robotic arm and then augments that with the social touch.

While imbuing a dialogue system with social capabilities offers several advantages, it is equally imperative to ensure that a task-oriented dialogue system remains comprehensible to its users. Traditional dialogue systems often operate as black boxes, rendering it challenging for users to navigate the system effectively or interpret any encountered errors. Explainability improves the understanding and trustworthiness of the dialogue system [You23]. Therefore, in addition to enhancing the social dimension of the dialogue system, there is an additional focus on making it explainable.

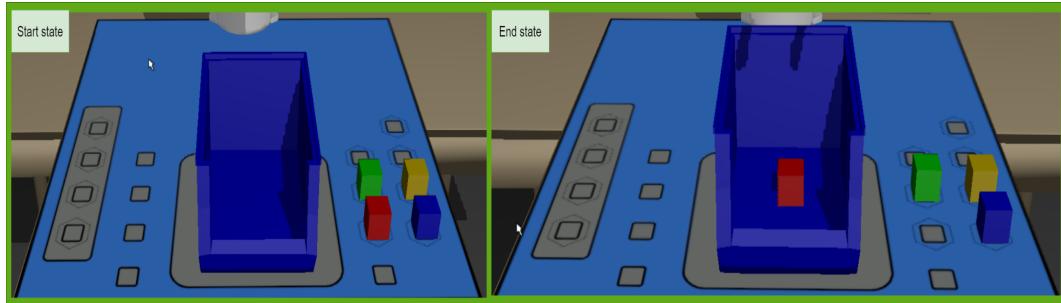


Figure 1.3: Example task-oriented utterance: 'Move the red cube into the container'.

1.2 Vision

Envisioning a future where task-oriented dialogue systems are accessible to non-experts holds significant promise for democratizing technology. Conventional task-oriented systems often demand technical proficiency, limiting their potential user base to individuals with specialized knowledge. However, by augmenting these systems with social capabilities, we can bridge the gap between experts and non-experts.

The vision is to have a free-flow conversation back and forth between the user and the dialogue system where the user can switch back and forth between the task and non-task-oriented utterances. Figure 1.4 shows the switching between such utterances where the user starts with social talk and then switches to task-related conversation.

1.3 Task and Objectives

The idea is to create a dialogue system capable of handling both task-oriented and social utterances effectively. This system should be designed to accommodate future extensibility, allowing users to integrate their own implementations seamlessly. The aim is to imbue the dialogue system with social capabilities to enhance the naturalness of task completion. Achieving this requires a clear distinction between utterances meant for task fulfillment and those intended for social interaction, ensuring precise processing and response for each type of utterance. Therefore the primary tasks of this thesis are as follows:

1. All task-oriented utterances with correct sentence structure are to be dispatched to the robotic arm for execution. The robotic arm will attempt to carry out the instructions provided the environment permits it (Figure 1.3). In the event of execution failure, appropriate feedback will be relayed to the user.
2. All casual utterances are to be handled separately and to be refrained from sending them to the robotic arm. Isolating these informal expressions will prevent any unintended actions triggered by such interactions, ensuring they are treated in isolation.

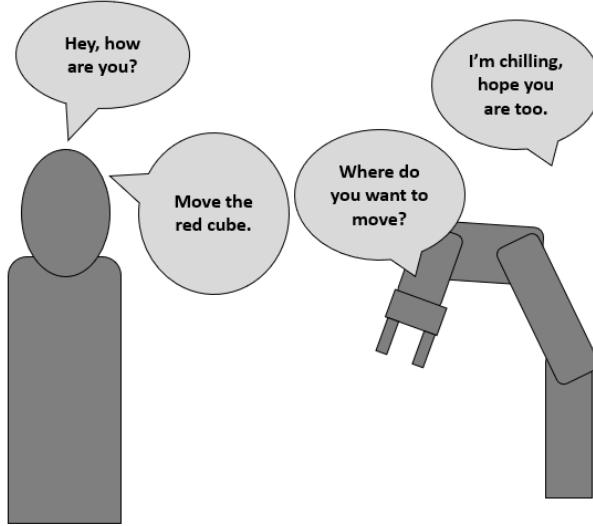


Figure 1.4: Example of task-oriented dialogue system with social capability.

Figure 1.4 shows how the conversation exchange with a socially capable task-oriented dialogue system would look like. The interaction starts with social talk and then switches to an instruction. Any user instruction in order to complete the task falls within the task-oriented dialogue, this instruction contains an intent to complete a goal and also specifies the entities required to complete the goal. To achieve a dialogue system that can complete the instruction, gives appropriate feedback in case of failures, is easily explainable, and is socially engaging, the following research objectives have been defined *Domain Classification*, *Intent Classification*, *Slot Identification*, *Response Generation*, *Explainable AI*, and *Large Language Model Integration*. Next, each of the objectives is elaborated in detail.

- O1 *Domain Classification*: To what extent can social and task-oriented utterances be distinguished?
- O2 *Intent Classification*: How can the intended user goal behind a task-oriented utterance be identified and mapped to the appropriate action?
- O3 *Slot Identification*: How to determine, what information/entities are necessary within the utterance to complete the action and whether that information is present within the user utterance?
- O4 *Response Generation*: How can the response be structured to prompt the user for the missing information necessary to complete the task mentioned in their utterance?
- O5 *Explainable AI (XAI) Impact*: To what extent does providing explanations to system responses improve user satisfaction?

O6 *Large Language Models (LLM) Integration:* Evaluation and comparison of multiple Large Language Models(LLMs), that can be integrated to achieve the desired social capability.

The dialogue system developed in this thesis can be integrated with robot simulation programs, which simulate the operation of robots and execute user instructions. One such program is the VerbBot simulation [Spangenberg17; Wölfel20](robot arm simulation). This integration offers a comprehensive end-to-end evaluation of the created dialogue system.

1.4 Chapter Overview

After the introduction which presents the motivation behind the research, outlines the envisioned dialogue system, and formulates the task and research objectives. Further, in chapter 2 the fundamental principles and terminologies related to dialogue systems are presented. A comprehensive review of the state-of-the-art in dialogue system development and related research is conducted, highlighting the contributions of this work. Chapter 3 discusses the construction and implementation of the dialogue pipeline. The dialogue pipeline is dissected into five components: domain classification, intent classification, slot filling, response generation, and social aspect incorporation. Chapter 4 focuses on the design of the web user interface for a more intuitive user experience. The system's explainability is explored, distinguishing between expert and non-expert explainability. Chapter 5 does the evaluation of the developed dialogue system to measure its performance and effectiveness. Results are presented, and strengths and limitations are analyzed. Chapter 6 concludes by summarizing the key findings and contributions. The discussion reflects on the research implications, and future work is outlined, including areas such as question-answering, intent confirmation, handling multi-intent utterances, and addressing mixed sentences with non-domain and domain utterances.

CHAPTER 2

Basics and Literature Review

This chapter provides an essential foundation for understanding the context and significance of our research. Subchapter 2.1 begins by presenting the basic principles and terminologies that form the building blocks of our study. Subchapter 2.2 delves into the State of the Art, where components of the NLP pipeline and their recent developments are discussed. In Subchapter 2.3, we conducted a thorough review of existing literature and research projects that are closely aligned with our study. This review allows us to identify gaps in the current knowledge and build upon previous contributions, positioning our research within the broader context of the field. Finally in Subchapter 2.4, we outline our contribution to the research.

2.1 Basic Terminologies

This section describes the key terms used throughout the thesis. Starting with the basic terms used very commonly in the areas of natural language processing to specific key terms that are specific to this thesis.

Natural Language Processing (NLP): NLP is a subfield of artificial intelligence that focuses on enabling computers to understand, interpret, and generate human language. The two main components of NLP are *Natural Language Understanding(NLU)* and *Natural Language Generation(NLG)*. NLU focuses on the comprehension and interpretation of human language by machines and NLG is dedicated to the generation of human-like language by machines.

Utterance: An utterance can encompass any expression provided by the user. It may range from as brief as a single word to as elaborate as a complex sentence or even an entire paragraph.

instruction: An instruction constitutes a purposeful utterance directed toward accomplishing a specific task. It is a command guiding what task to perform. Simple instructions can be like “Close the door”.

Task-oriented dialogue systems (TODS): Also referred to as *closed-domain dialogue systems (CDDS)*, are specialized conversational agents designed with the primary objective of accomplishing specific tasks within their domains, such as customer service chatbots for food delivery or ticket booking services. While TODS may also have the social capacity to engage in casual conversations, their primary focus remains on task completion rather than social interactions. Throughout this thesis, the terms “task-oriented” and “closed-domain” will be employed interchangeably, as they share the same fundamental meaning and context. To ensure brevity and convenience, the abbreviations TODS and CDDs will be utilized in subsequent discussions.

Non-task-oriented dialogue systems (non-TODS), alternatively known as *open-domain dialogue systems (ODDS)*, are conversational agents that prioritize engaging in informal and casual conversations with users, lacking any specific objectives related to task completion. These systems are designed to foster free-flowing exchanges with users, emphasizing the enjoyment of social interactions rather than the fulfillment of particular tasks or domain-specific functions. Within the context of this thesis, the terms “casual talk”, “social talk”, and “small talk” will be utilized interchangeably to refer to the nature of interactions facilitated by non-TODS. Furthermore, the designations non-TODS and ODDS will also be employed interchangeably throughout this research to denote this category of dialogue systems.

Transformers: Transformers are an advanced neural network architecture that has greatly improved how machines understand and generate human language. They use a clever technique called self-attention [Vaswani23] to process language more efficiently and capture meaningful connections between words. Transformers have played a significant role in tasks like language translation and text analysis, making them an essential tool in the field of Natural Language Processing.

Large Language Models (LLM): Sophisticated artificial intelligence models trained on extensive datasets to understand and generate human language. These models, such as GPT-3 [Brown20] with 175 billion parameters, have shown remarkable capabilities in various language-related tasks, including text generation, translation, and question-answering. Throughout this thesis, the term “Large Language Models (LLMs)” will be used to refer to these advanced AI language models.

2.2 State of the Art

Figure 2.1 depicts the pipeline for the traditional dialogue system. The two most important parts of any dialogue system are: NLU and NLG, these are the basic building blocks of the NLP.

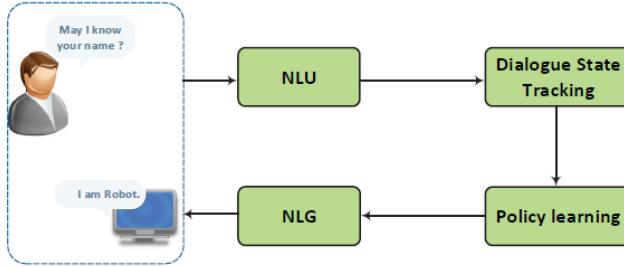


Figure 2.1: NLP Pipeline for Task-oriented Systems [Chen17].

NLU enables machines to understand and analyze natural language by extracting concepts, entities, emotions, and keywords. It plays a crucial role in comprehending user problems reported verbally or in writing. NLU involves various levels of processing 2.2 as listed below, all these processing are focused on understanding a specific component that in overall leads to an understanding of the overall user utterance.

- *Phonology*: Deals with the systematic arrangement of sounds in language.
- *Morphology*: Focuses on the smallest units of meaning known as morphemes.
- *Lexical*: Involves understanding the meaning of individual words, part-of-speech tagging, and word-level analysis.
- *Syntactic*: Analyzes the grammatical structure of sentences and the dependencies between words.
- *Semantic*: Determines the proper meaning of sentences and deals with word disambiguation.
- *Discourse*: Deals with analyzing logical structures and connections between sentences to ensure coherence.
- *Pragmatic*: Focuses on understanding implied meaning based on context and background knowledge.

NLG is the process of generating meaningful phrases, sentences, and paragraphs from internal representations. It involves four phases: goal identification, planning, evaluating the situation, and realizing plans as text ([Khurana23]). NLG works opposite to NLU.

The other two major components are the dialogue state tracker and dialogue policy learning, they have become an integral part of the TODS, as the dialogue state tracker depicts the user goal from the current utterance and previous dialogue history and outputs the current state. The policy learning uses this state and maps it to the appropriate action to be taken. For example, if the current state is “*recommend*” then the “*recommendation*” action will be taken. It is noteworthy that the aforementioned

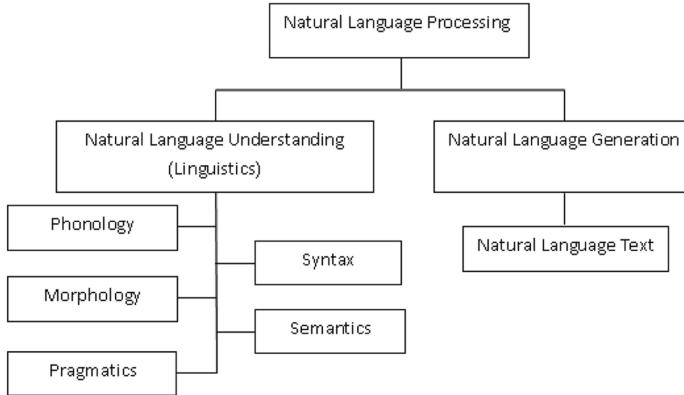


Figure 2.2: NLP components [Khurana23].

conventional components persist, yet there has been discernible progress in recent times concerning the nuanced treatment of each facet encompassing the system’s operational pipeline.

Authors in [Chen17] highlight the recent advancements, like recent advances in NLU including the use of deep learning techniques to learn meaningful feature representations and reduce the amount of hand-crafting required. One example of a deep learning technique used in NLU is the use of a deep feed-forward neural network for context-response matching. NLG is also using deep learning techniques to generate more natural and diverse responses. One example of a deep learning technique used in NLG is the sequence-to-sequence (Seq2Seq) model, which uses a recurrent neural network (RNN) to generate a response sequence given an input sequence. Authors in [Chen17] show that deep learning is being used in every component, it mentions the recent research in the use of a recurrent neural network (RNN) to model the dialogue history and predict the current dialogue state for the dialogue state tracking. For policy tracking the aforementioned paper highlights the use of deep reinforcement learning (RL) techniques to improve the learning process and optimize the policy.

2.3 Related work

This subsection provides an exposition of the collective body of prior research within the realm of TODS and its purview to encompass the broader domain of social conversation systems. It illustrates the recent works to make TODS systems conversational and also the recent developments in dialogue systems that integrate task and non-task-oriented utterances. This amalgamation serves the purpose of endowing the resultant systems with a heightened degree of conversational authenticity and naturalness.

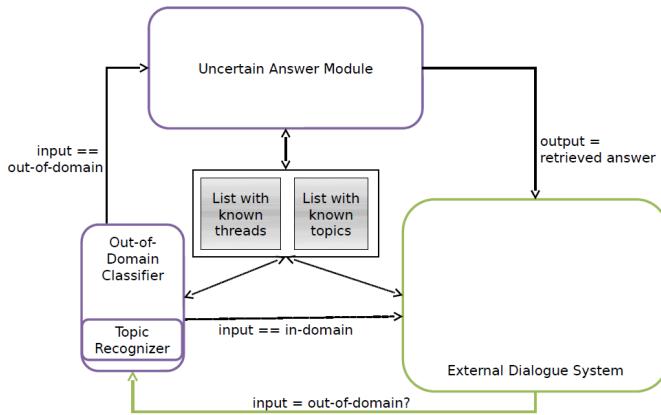


Figure 2.3: Social Component for Out of domain utterances [Klüwer15].

2.3.1 History

The concept of making dialogue bots social or imbuing them with social capabilities has been an ongoing area of research and development in the field of artificial intelligence and human-computer interaction. The idea of creating more socially intelligent agents dates back several decades, with the initial efforts focused on designing conversational agents that could engage users in more natural and interactive ways.

One significant milestone in the quest to make dialogue bots social was the development of ELIZA [Weizenbaum66], it was an early natural language processing program designed to simulate conversation with a Rogerian psychotherapist. It could respond to user's inputs by employing pattern matching and rephrasing techniques, giving the illusion of understanding and engaging in empathetic conversation. While ELIZA was limited in its abilities, it laid the foundation for future advancements in social chatbots. Another important development came in the early 2000s with the rise of chatbots for various online platforms and instant messaging applications. These chatbots were designed to interact with users in a more conversational manner, providing assistance and information on a range of topics. They marked an early step towards creating socially-oriented dialogue systems that aimed to engage users beyond functional interactions.

However, it wasn't until more recent years, with advancements in natural language processing, deep learning, and reinforcement learning techniques, that the idea of making dialogue bots truly social gained significant traction. Around the mid-2010s, research efforts intensified in the direction of endowing chatbots and conversational agents with emotional intelligence, empathy, and the ability to understand and respond to users' emotions, making interactions more engaging and lifelike.

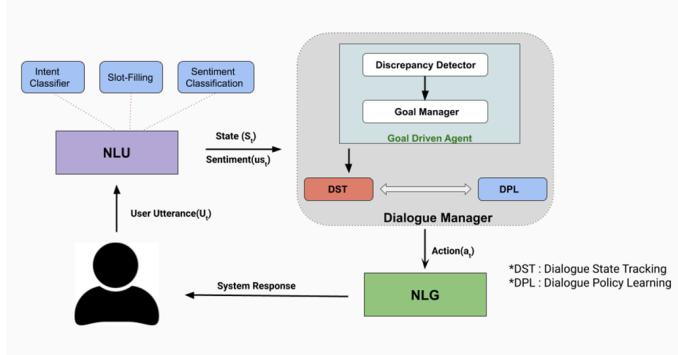


Figure 2.4: Goal-driven TODS [Tiwari21].

2.3.2 Recent development

Authors in [Klüwer15] describe how to handle *out-of-domain* (non-task-oriented) utterances with the TODS system. In the domain classification process, anything that is not within the knowledge base of the TODS system will be handled by the social component(Uncertain Answer Module 2.3).

In [Yildiz21] authors introduce a dialogue management system framework that incorporates social practices for more natural and efficient dialogue systems. And uses the proposed system in a scenario between the doctor and patient roles where the doctor is a medical student and the patient is simulated by the dialogue management system.

Authors in [Tiwari21] tackle the challenge of dynamic goal adaptation in TODS systems. Their approach involves the creation of the “Deviation adapted Virtual Agent (DevVA)” dataset and the development of a “Dynamic Goal Driven Dialogue Agent (DGDVA)” featuring a reinforcement learning-based Dynamic Goal Driven Module (GDM Figure 2.4). By harnessing user sentiment as feedback, the DGDVA adeptly adjusts to users’ evolving goals. Notably, the paper’s experimental findings showcase the DGDVA’s proficiency in adapting to dynamic user goals through reinforcement learning, yielding a remarkable task success rate of 0.88 and enhancing overall conversational ability and user satisfaction.

In the context of task-oriented dialog systems, which excel in tasks aligned with their capabilities, issues arise when user intentions are implicit. To address this, a novel approach is proposed to interleave everyday social conversation with task-related dialogue, ensuring sustained user engagement even if task content falters ([Yu17]). This framework employs reinforcement learning(Response Selection Policy 2.5) to enable seamless transitions between task and non-task segments, enhancing conversation coherence. Experimental results using a movie promotion dialog system demonstrate that this approach improves task success rates and user engagement compared to systems solely focused on tasks.

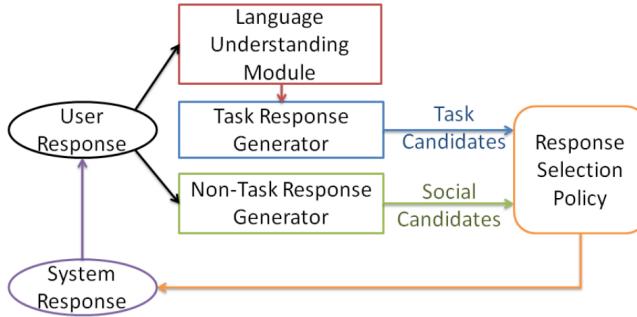


Figure 2.5: Social interleaving with TODS [Yu17].

The Generative Pretrained Transformer (GPT) has emerged as a remarkable advancement in the realm of natural language processing, particularly in the context of social conversation. First introduced by [Radford18], GPT leverages its transformer-based architecture to capture intricate contextual dependencies in textual data. Through unsupervised learning on vast text corpora, GPT acquires a deep understanding of linguistic nuances and patterns, enabling it to engage in social conversations that closely resemble human interactions. Subsequent iterations, such as GPT-2 presented by [Radford19] and GPT-3 elaborated upon by [Brown20], have progressively elevated the model's conversational prowess.

2.4 Research contribution

In the work presented by [Yu17], the authors incorporated social conversation elements when user intention lacked clarity. They introduced a methodology that generated both task-oriented and non-task-oriented responses for each user utterance, subsequently employing reinforcement learning to determine the optimal response. While the paper introduces a pioneering technique involving a knowledge base to generate non-task-oriented responses, our proposal extends this approach by advocating for the utilization of large language models (LLMs) to facilitate the generation of non-task-oriented responses when the user's utterance deviates from the confines of the task domain (3.4).

Within the realm of robotic tasks, we present our approach focusing on distinguishing user utterances into task-oriented and non-task-oriented categories. This method is specifically designed for tasks involving robotic arms, and its applicability can be extended to various other tasks beyond this domain (3.1). We explore user intent detection within these utterances by incorporating established classification techniques. What distinguishes our research is the creative integration of these methods with our approach, resulting in a hybrid intent detection mechanism. This integration enhances the accuracy of identifying user intentions (3.2).

Furthermore, our study addresses a fundamental challenge in task-oriented dialogues: the management of essential information required for task completion. We introduce the concept of representing this vital information in the form of JSON formatted slots, providing a structured framework. These slots encapsulate the necessary details, ensuring a comprehensive understanding of task requirements (3.3). To optimize task execution, we have developed a response structure intricately linked to these slots. This structure enables the generation of contextually appropriate responses in scenarios where crucial information, represented by these slots, is missing (3.3.3).

All our work is done within our constructed dialogue framework. Our offered framework can further be used in the future for the development of dialogue systems tailored for robotic tasks.

CHAPTER 3

Dialogue Pipeline

The previous section explored existing literature, setting the groundwork for our research. Here, we introduce our constructed dialogue pipeline. The diagram referenced as Figure 3.1 illustrates the proposed sequence of steps in the pipeline, which outlines how utterances are processed. The primary building blocks of any Natural Language Processing(NLP) pipeline are the modules responsible for Natural Language Understanding(NLU) and Natural Language Generation(NLG). In the visual representation, sections shaded in blue represent the NLU components, while those in green correspond to the NLG components for our dialogue pipeline

Before delving into the detailed technical aspects of each module, it is important to provide a brief overview of these individual components. This initial overview offers a general understanding of the roles and functions carried out by each integral part within the depicted pipeline architecture.

Domain Classification: The preliminary phase of the pipeline involves the categorization of incoming utterances into closed-domain or open-domain contexts. This initial classification serves as a pivotal determinant in orchestrating subsequent processing trajectories.

Open-Domain Response Generation (Open-Domain Conversation AI): Upon discerning an utterance as non-task-oriented(non-TODS), the system undertakes the generation of socially contextual discourse.

Intent Classification: The intent classification module serves to decipher latent user objectives or goals encapsulated within the enunciated statement. This module engages in deciphering the discernibility of intent within the input and subsequently maps the inferred intent to an actionable maneuver executable by the robotic arm.

Retrieval-Based Methods: In instances where the input lacks overtly actionable intent, yet remains contextually relevant to the overarching task domain, retrieval-based

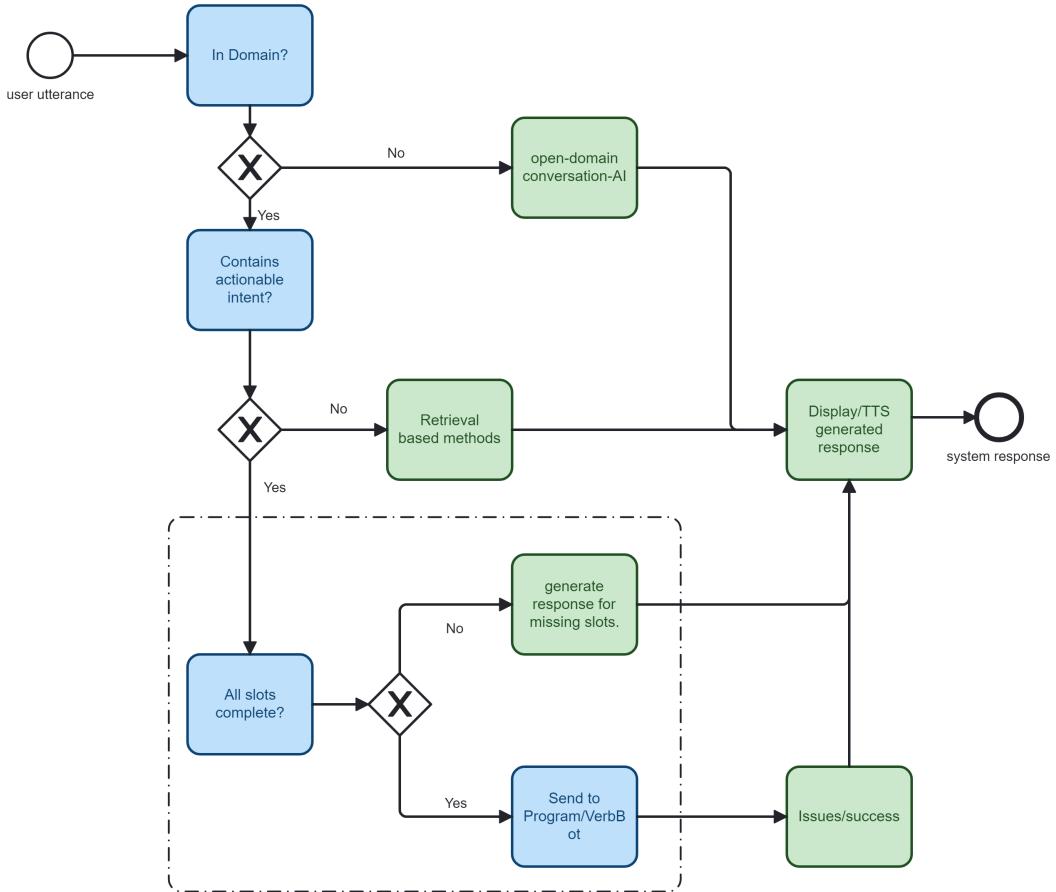


Figure 3.1: Proposed dialogue Pipeline, colored in blue represents NLU components while green represents NLG components

methodologies are activated. This entails generating responses predicated upon a pre-established textual corpus of the domain. Implementation of this is not scoped in this thesis and will be part of the outlook.

Slot Filling: The dashed enclosure in Figure 3.1 denotes the slot-filling component, comprising three interconnected sub-components. When a recognizable actionable intent is identified within the user's utterance, it transitions to this stage. Initially, an evaluation ascertains the presence of essential information requisite for task completion within the structural fabric of the sentence. In the case of structural incompleteness, a tailored response is devised to ask user input to address the informational gap. In the instance of structural integrity, the instruction is relayed to the robotic arm. This iterative slot-filling process persists until all essential slots are populated, or until a user gives a new instruction or switches to open-domain conversation.

Robotic-Arm Feedback: Outcomes ensuing from executed robotic arm tasks trigger corresponding feedback messages, encapsulating instances of successful task completion or encountered failures. These messages are subsequently relayed to the user.

Relaying Response: Generated messages are shared with the user in a way that is easy to understand and coherent,

The dialogue framework outlined in Figure 3.1 is adaptable for future enhancements. Users have the flexibility to develop their unique implementations for each component, allowing for customization and the creation of tailored dialogue systems.

3.1 Domain Classification

Identifying in-domain utterances is vital to our context, given that our domain centers around a robotic arm’s tangible interaction with objects. This section delves into the implementation of our domain classifier.

Prior to that, we will define two essential terms that will be employed throughout this segment, *Open-Domain words* and *Closed-Domain words*. Note it’s crucial to differentiate between *open-domain words* and *open-domain utterances* and the same for *closed-domain words* and *utterances*, they mean completely different things.

Open-Domain Words: These are words characterized by their capacity for easy expansion over time. They often include nouns, verbs, adjectives, and adverbs. Their flexibility enables them to describe a wide array of concepts and ideas. Examples of open-domain words encompass “computer”, “run”, “happy”, and “quickly”.

Closed-Domain Words(or stop-words): These are words identified by analyzing extensive language data. They encompass frequently encountered functional words like prepositions, pronouns, conjunctions, and articles. Instances of closed-domain words comprise “the”, “and”, “in”, and “he”.

Understanding the distinction between open and closed-domain words holds significance as it aids in comprehending the nuances of language for conveying diverse types of information. Open-domain words are inclined to convey specific details and ideas, while closed-domain words structure language and indicate relationships between words and phrases.

3.1.1 Existing approaches

Classification approaches enable the categorization of data into distinct groups or classes based on various features or attributes. One common type of classification is binary classification, where the goal is to categorize data into one of two mutually exclusive classes or outcomes. These binary classification approaches can be used here

to classify the incoming user utterance as in-domain or out-domain. The following are the existing approaches.

Supervised Learning: Supervised learning is a common approach in machine learning where the model is trained on a labeled dataset, meaning that each input data point is associated with its corresponding correct output label. In binary classification, the model learns the relationship between input features and the binary class labels. It then uses this learned information to make predictions on new, unseen data.

Transfer Learning: Transfer learning is a machine learning technique where a model developed for one task is reused as the starting point for a model on a different but related task. In binary classification, transfer learning involves using a pre-trained model, often trained on a large and diverse dataset, as a foundation for a new binary classification task. This can be particularly useful when you have limited labeled data for your specific task, as the pre-trained model can capture general patterns and features that are relevant across tasks.

Rule-Based Methods: Rule-based methods involve creating a set of rules that dictate how to classify new data points based on certain conditions or patterns in the input features. These rules can be explicitly defined by domain experts or learned from the data. For binary classification, rule-based methods involve creating a set of rules that, when applied to the input data, lead to a binary class decision. Rule-based methods can be intuitive and interpretable, but they might struggle with complex relationships in the data.

The 3.1 table provides an overview of the strengths and weaknesses associated with various classification approaches. This forms the basis for the rationale behind our selection of the *Rule-based classification* approach. Our decision is influenced by the unavailability of training data, which precludes the utilization of both supervised and transfer learning methodologies. In the subsequent section, we present a detailed exposition of our chosen approach.

3.1.2 Rule-based classification approach

Here we formalize our classification approach, the classification problem can be represented as a tuple (U, S, V, \mathcal{D}_T)

Where:

- U represents the set of all utterances.
- S represents the set of all stop words.
- V represents the domain vocabulary.
- $\mathcal{D}_T : U \rightarrow \{\text{True}, \text{False}\}$ represents the function that maps utterances to possible outputs.

Approach	Strengths	Weaknesses
Supervised Learning	<ul style="list-style-type: none"> Effective with sufficient labeled data. Accurate predictions on new, unseen data. 	<ul style="list-style-type: none"> Requires a substantial amount of labeled data. May overfit if the model is too complex or the dataset is small.
Transfer Learning	<ul style="list-style-type: none"> Leverages pre-trained models for improved performance. Useful when labeled data is limited. 	<ul style="list-style-type: none"> Relies on the availability of related pre-trained models. Might not capture task-specific nuances.
Rule-Based Methods	<ul style="list-style-type: none"> Provides interpretable decisions based on explicit rules. Suitable when underlying patterns are well-known. 	<ul style="list-style-type: none"> May not capture intricate relationships. Expertise needed for defining accurate rules.

Table 3.1: Strengths and Weaknesses of domain classification approaches, [Bishop06] [Pan10] [Džeroski04]

For an arbitrary input utterance u where $u \in U$, define the following functions:

- $L(u)$ = Set of words in utterance u converted to lowercase and lemmatized (reduced to the base form, e.g. “cubes” is reduced to “cube”).
- $R(u) = L(u) - S$: Words after removing stop words from $L(u)$.
- $P(u) = \frac{|R(u) \cap V|}{|R(u)|}$: Proportion of words in $R(u)$ that are in V .
- $D_T(u) = P(u) \geq T$, where T is the threshold above which the classification is considered True.

The classification outcome is U_d when an utterance is classified as True otherwise \bar{U}_d . $U_d = \{u \in U \mid D_T(u) = \text{True}\}$, where U_d represents set of in-domain utterances. $\bar{U}_d = \{u \in U \mid D_T(u) = \text{False}\}$, where \bar{U}_d represents the set of out-of-domain utterances, and $U = U_d \cup \bar{U}_d$

This formulation encapsulates the problem using a tuple, the related functions, and the classification decision function $\mathcal{D}_T(u)$ with the updated outcome descriptions.

The vocabulary V can be designed by the most prevalent words within the domain of robotic arm operations. This encompassed terms denoting objects the arm interacts with, as well as action-oriented verbs used to convey instructions effectively. And for

the list of stop words, it is readily available (e.g. Spacy Library [Honnibal17]), from which the set S can be constructed

For example, if the threshold is chosen as 0.8 then if 80% of the words in $R(u)$ are found in the vocabulary V , the utterance is categorized as within the domain. A lower threshold can be chosen to make the classification more flexible but keeping it too low can lead to false positives. It also can't be 100% as that would be too rigid and won't accommodate any synonyms of the words present in the vocabulary. Considering this our prototype threshold was chosen to be 0.8.

We proceed to illustrate the application of our approach through classification examples, the examples presented in boxes 3.1 and 3.2 illustrate the preprocessing steps applied to user utterances. Initially, the words in the utterance are converted to lowercase and lemmatized, denoted as $L(u)$. Subsequently, stop words are removed from the lemmatized set, resulting in $R(u)$.

```

u = "Can you move the red cubes into the container?"

L(u) = [can, you, move, the, red, cube, into, the, container]
R(u) = [move, red, cube, container]
```

Box 3.1: Example 1: Preprocessing of user utterance

It is noteworthy that all the words in $R(u)$ are found within the vocabulary V . This observation leads to $P(u) = 1$, thereby rendering $\mathcal{D}_T(u) = \text{True}$. Consequently, u is categorized as belonging to the domain ($u \in U_d$).

```

u = "Are liquids stored in the container?"

L(u) = [be, liquid, store, in, the, container]
R(u) = [liquid, store, container]
```

Box 3.2: Example 2: Preprocessing of user utterance

An interesting aspect emerges from the analysis of $R(u)$, where all words except 'store' are encountered within the vocabulary V . This leads to a $P(u)$ value of $\frac{2}{3} = 0.67$. Consequently, the outcome of $\mathcal{D}_T(u)$ is False, classifying u as falling outside the domain ($u \in \overline{U}_d$).

3.2 Intent Classification

Intent refers to the fundamental objective or purpose that a user seeks to achieve through verbal or textual communication. It encapsulates the underlying motivation

or goal that is conveyed within a given utterance. For instance, consider the following examples:

$$\begin{aligned} u1 &= \text{"Help me book a flight."} \\ u2 &= \text{"Please book a flight."} \end{aligned}$$

For utterance $u1$, the user's intent pertains to seeking assistance and relevant information about the process of booking a flight. In utterance $u2$, the user's intent is centered around the actual act of booking a flight, indicating their desire for practical engagement in the reservation process. The identification and understanding of intent within user utterances are pivotal as it allows for tailored and contextually appropriate responses or actions to be generated in response to user inputs.

3.2.1 Intent in Task-oriented dialogues

In the context of the robotic arm domain, which involves tasks of robotic manipulation, a variety of user queries can arise, ranging from explicit instructions to queries seeking information about the system. While the primary focus lies in dealing with instructions, it's important to acknowledge that not all queries are instructional in nature. Some queries are more informative, seeking to gather information about the state of the system or its components.

For instance, questions like "What is the size of your arm?", "Can you move the robot gripper?", and "Is there liquid in the container?" fall within the robotic arm domain but do not inherently contain actionable instructions. Given that our research scope concentrates on facilitating instructional interactions (TODS), we consider the implementation of retrieval-based methods to address such queries as a future outlook. However, in cases where the intent lacks a clear actionable nature, a fallback mechanism directs these queries to an open-domain conversational AI module (ODCAI), which is equipped to handle broader conversational contexts.

Conversely, we focus on examples that exemplify instructional intent specific to the Task-Oriented Dialogue System (TODS) framework. These examples include instructions explicitly given to the robot arm:

- "Place the red cube into the box."
- "Move the red cube 5 cm to the right."
- "Could you kindly shake the bottle?"
- "Please pour the cylinder into the container."

Notably, in the context of these instructional instances, the root verbs serve as descriptive cues that encapsulate the primary intention behind the instruction. This

emphasis on the root verbs aligns with the practice of identifying the intended action in the instructional context of the TODS domain.

3.2.2 Intent classification problem

Now we formalize the problem. For $u \in U_d$, where U_d represents the collection of in-domain utterances, we define a tuple (U_d, I, δ, A) as follows:

- $U_d \subseteq U$ denotes the subset of in-domain utterances (See section 3.1).
- I signifies the set encompassing all possible intents.
- $\delta : U_d \rightarrow A$ stands as a transition function mapping in-domain utterances U_d to corresponding actions A .
- A represents the set of all possible output actions that the robotic arm can execute.

The transition function δ can be subdivided into two functions δ_1 and δ_2 . Function $\delta_1 : U_d \rightarrow I$, that map in-domain utterances U_d to their respective intents I , δ_1 also refers to the intent identification approaches, which will be discussed in the next section. Then function $\delta_2 : I \rightarrow A$, maps intent to actions. The transition function uses both these functions to get action from u .

This formalized presentation establishes the core constituents governing the classification of utterances into intents and the consequent mapping of intents to actions.
∴ For input $u \in U_d$, it results in output $a \in A$.

3.2.3 Intent Classification Approaches

Before heading on to describing the approaches, first, we will define the list of actions that were associated with our domain.

$$A = [\text{move, shake, pour, clean, close, open}]$$

As a simplification of the below classification approaches we considered intent and actions to be the same (i.e. $I = A$), so no mapping is needed here.

Zero-shot classification

Zero-shot classification involves training a model to classify instances into classes that it has not encountered during training. In other words, the model is expected to generalize its understanding from seen classes to unseen classes based on some auxiliary information. This auxiliary information could be in the form of textual descriptions, attributes, or other relevant metadata associated with the classes.

The zero-shot classification is built on Natural Language Inference (NLI), also known as recognizing textual entailment, which is a fundamental task in natural language processing. It involves determining the logical relationship between two given sentences: a premise and a hypothesis. The goal is to ascertain whether the hypothesis can be logically inferred from the information presented in the premise. NLI models are trained to classify this relationship into three categories: “entailment” (the hypothesis can be inferred from the premise), “contradiction” (the hypothesis contradicts the premise), or “neutral” (there is no clear logical relationship between the two).

Premise: The cat is sitting on a windowsill.

Hypothesis: The feline is perched near a window.

In this example, the relationship between the premise and hypothesis is “entailment” since the information in the hypothesis can be logically inferred from the premise.

The approach proposed by [Yin19] uses the approach of NLI in zero-shot classification. They utilize pre-trained NLI models as effective zero-shot sequence classifiers. To achieve this, the sequence to be classified is treated as the premise in an NLI context, and a hypothesis is formulated for each potential class label. So in our scenario, the user utterance can be considered as the premise and the actions A as the hypothesis, here each element in A would be a potential class label. NLI Models to do this are readily available like *facebook/bart-large-mnli* that uses bart-large model [Lewis19] after being trained on the MultiNLI [Williams18] dataset. Below is an example

Premise : Place the red cube into the box.

Hypothesis : [move, shake, pour, clean, close, open]

The pre-trained NLI model is then used to evaluate the relationship between the premise and hypothesis, yielding probabilities for entailment and contradiction. These probabilities are subsequently repurposed as label probabilities for zero-shot classification. If the entailment probability is high for ‘move’, it indicates that the text is most likely related to the ‘move’ class, even if the model has never been explicitly trained on such examples.

In essence, by utilizing the NLI framework, the model leverages its ability to comprehend semantic relationships between texts to make predictions for unseen classes. This integration of NLI principles with zero-shot classification offers a powerful mechanism for extending a model’s classification capabilities to categories absent from its training data.

Word Embedding

Word embeddings provide a way to transform words into numerical vectors that uncover their semantic relationships. Words can be represented as points in a multi-dimensional space, where proximity reflects similarity. For instance, consider ‘king’,

'queen', 'man', and 'woman'. By manipulating their vectors, such as subtracting 'man' and adding 'woman' to 'king', the resulting point is remarkably close to 'queen'. This showcases how word embeddings capture analogies and relationships. Prominent techniques like Word2Vec [Mikolov13] and GloVe [Pennington14] drive this process, enabling NLP models to grasp nuances and connections within language.

As mentioned before, the root verb of the sentence in task-oriented utterances is the intent. With this approach, one can extract the root verb from the sentence and generate its vector. Subsequently, each action in set A is individually matched with the vector of the root verb, where vectors for each action have also been computed. The action with the highest similarity value is then selected and executed. Box 3.3 gives an example of intent classification using the word embedding approach.

```

Utterance: "Place the red cube into the box"
=> Root-Verb: Place
Action Probabilities: [move: 0.7326, shake: 0.048, pour: 0.0702,
                      clean: 0.085, close: 0.038, open: 0.0262]
=> intent(action): move

```

Box 3.3: intent classification with word embedding

Approach	Strengths	Weaknesses
Zero-shot Classification	<ul style="list-style-type: none"> • No training required • Outperforms word embeddings 	<ul style="list-style-type: none"> • Not enough auxiliary information available for our labels • Doesn't work in some cases (e.g., "swipe the whiteboard" \Rightarrow Intent: shake)
Word Embeddings	<ul style="list-style-type: none"> • No training required 	<ul style="list-style-type: none"> • Doesn't give accurate results in some cases (e.g., "Place the cube in box." \Rightarrow Intent: shake) • Performs worse than zero-shot classification

Table 3.2: Comparison of Zero-shot Classification([Socher13]) and Word Embeddings

Table 3.2 presents an assessment of the strengths and limitations associated with the aforementioned approaches. Upon employing these approaches across numerous example utterances, it was observed that these methods do not consistently yield accurate outcomes, as elucidated in the weaknesses section of the table. Consequently, in light of these observations, a hybrid approach was deemed more suitable.

Hybrid approach

Building on top of the previous mathematical notation, we formalize our approach here. Given tuple (U_d, I, δ, A) , that we formalized in section 3.2.2. We define

- Synonym sets: For $a_i \in A$, there exists a synonym set S_i such that $\forall i \neq j$, $S_i \cap S_j = \emptyset$. The synonyms are chosen from various example sentences, not from the synonyms list in the english dictionary.
- The function $\text{match}(a, i)$ that returns true if $\text{intent}(\text{root verb}) i \in I$ is present in the set of synonyms $S(a)$ for action a :

$$\text{match}(a_k, i) = ((i) \cap S_k \neq \emptyset)$$

This function returns true if i exists in the set of synonyms $S(a)$ for action a .

In a Hybrid approach, for every user utterance (u) in the set U_d , a function denoted as δ_1 (see section 3.2.2) extracts the root verb(i) from the utterance u . Subsequently, the match function is applied to evaluate compatibility with each action, this is done by function δ_2 .

$$\delta_2(i) = \{a \mid a \in A \text{ and } \text{match}(a, i)\}$$

Since all the actions have disjoint synonym sets, the above algorithm will lead to a single output action or it can be empty. As mentioned in section 3.2.2, the δ_1 defines the intent classification approaches. In the previous approaches, δ and δ_1 were the same ($I = A$), here δ_1 and δ_2 are used within δ function as proposed in 3.2.2 ($I \neq A$)

$$\therefore \delta(u) = \begin{cases} \delta_2(u), & \text{if } \delta_2(u) \neq \emptyset \\ \text{ZSC}(u), & \text{if } \delta_2(u) = \emptyset \text{ (Zero-Shot Classification)} \end{cases}$$

Below is an example, here the intent i.e. the root verb of the utterance is 'place' which is present in the move action synonym set, therefore the action to be taken is 'move'.

$$\begin{aligned} A &= [\text{move, pour, shake, clean, open, close}] \\ \text{MOVE} &= [\text{move, position, lay, push, set, place}] \\ L(u) &= \text{"place the red cube into the container"} \\ \text{intent} &= \text{'place'} \end{aligned}$$

3.3 Slot Filling and Response generation

Slot filling is an information extraction task, it involves identifying and extracting specific pieces of information, known as "slots," from a user's input or query. These slots typically correspond to specific types of information that the system needs to

gather to fulfill a user's request or perform a task [Jurafsky]. In the context of a dialog system or chatbot, a slot can represent a particular piece of information that the system needs to collect from the user to provide a relevant and accurate response. For example, if a user is booking a flight, the slots might include departure city, destination city, departure date, and return date. The system's goal is to prompt the user for these specific pieces of information and then use them to complete the desired task.

```
User: I want to book a flight.  
Chatbot: Great! I can help you with that. Could you please provide  
me with your departure city, destination city, and  
departure date?  
User: Departure city is Munich, destination is Delhi, and I want to  
leave next Friday.
```

Box 3.4: Example of slot-filling in conversation

In a conversation, the slot-filling process might look like Box 3.4. In this instance, the chatbot has identified essential slots such as departure city, destination city, and departure date for the "booking" process. The chatbot prompts the user to provide this information and subsequently extracts the necessary details from the user's response. This extracted information enables the chatbot to progress seamlessly with the flight booking process.

Response generation is closely tied to the concept of slots. In the example provided earlier, the response generated by the chatbot involved posing questions to the user, aiming to obtain the necessary information to complete the missing slots.

The extraction of slots from the utterance is contingent upon the intent expressed within that utterance. In this section, we will be using the terms *intent* and *action* interchangeably. It's crucial not to conflate this with the terminology employed in the preceding section, where *action* holds robot-arm domain-specific meaning while *intent* is the more common usage within the NLP community. As demonstrated in the aforementioned example, the intent was to *book flight*, thus necessitating the extraction of slots such as departure and destination cities, along with the date.

3.3.1 Existing approaches

Considerable effort has been dedicated to the field of *intent and slot filling*, resulting in the development of various models. It's worth noting that these models have predominantly been crafted using specific datasets.

- *ATIS Dataset* [MCTC]: The ATIS (Airline Travel Information Systems) dataset contains queries related to airline travel. Each query is annotated with the intent and slot labels. Intent labels in the ATIS dataset correspond to actions

or queries related to airline travel. Common intent categories include flight booking, flight status inquiries, airport information, and travel schedule queries. Whereas slot labels in ATIS denote specific details relevant to airline travel. These include departure and arrival cities, travel dates, flight numbers, seating preferences, and other travel-related information.

- *SNIPS Dataset* [Sonos]: The SNIPS dataset consists of common spoken language commands or queries. Similar to ATIS, each query is annotated here with intent and slot labels as well. In the SNIPS dataset, intent labels cover a wide range of user intentions across different domains. Examples of intent categories include setting alarms, playing music, weather inquiries, restaurant reservations, and more. Slot labels in SNIPS represent specific pieces of information within the user’s query. These could include date and time information, location names, quantities, and other details depending on the intent.

Many models have been developed using the above datasets, including [Wang18], [Wu20], and [Qin19], etc. These datasets serve as foundational tools for constructing models capable of efficiently identifying user intent and extracting pertinent slot information. The existing models crafted from these datasets can either be directly applied or subjected to fine-tuning with additional data, catering to specific use-case requirements. However, given the scarcity of available data for fine-tuning, we developed a distinct approach for slot filling, as detailed in the upcoming section.

3.3.2 Slot filling for task-oriented instructions

Let A be the set of intents. For each intent $a \in A$, let S_a be the set of sub-intents associated with a , denoted as $S_a = \{s_1, s_2, \dots\}$. For each sub-intent $s \in S_a$, let T_s be the set of slots needed for s , represented as $T_s = \{slot_1, slot_2, \dots\}$. To improve the comprehensibility of the notion related to sub-intents, let’s begin by exploring the structure of an instruction. After that, we will outline the step-by-step process for the algorithm associated with slot filling. We will use the example of intent “move”, which can be generalized to other intents as well.

Instruction structure

Figure 3.2, referred to as a dependency graph, provides a visual representation of the relationships between words within a sentence. A dependency graph illustrates the connections and dependencies between individual words, elucidating the syntactical structure of the sentence. Positioned beneath each word in the graph is its corresponding Part of Speech (POS) tag, denoting its grammatical classification, such as verb, noun, and so forth. Additionally, each word is associated with an incoming dependency, also known as a DEP, such as *dobj* (direct object) and *pobj* (preposition object), offering further insights into the grammatical and semantic relationships

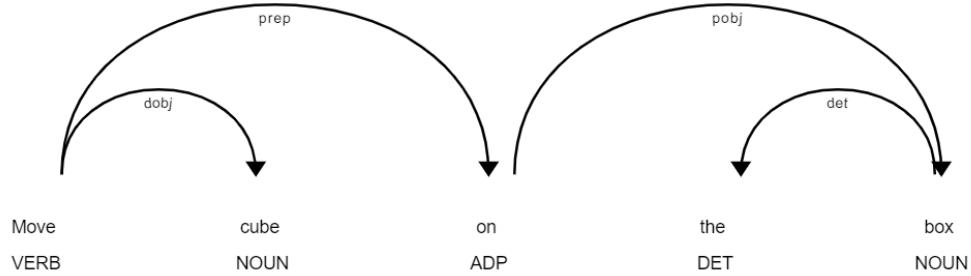


Figure 3.2: Basic Move instruction structure (created using [Honnibal17])

within the sentence ([De Marneffe21]). This visual and syntactical analysis aids in the comprehensive understanding of sentence structure and linguistic nuances.

It is crucial to emphasize that the provided examples serve a vital purpose: they underscore the significance of sub-intents and slots within the dialogue system. The examples specifically focus on the “move” intent, offering specific dependency graphs. It’s important to note that a single instruction can be articulated in various ways, each potentially resulting in a different dependency graph. By demonstrating this process in specific scenarios, we illustrate its adaptability to other intents and diverse dependency graphs. This showcases the system’s inherent flexibility, paving the way for seamless extensibility in future applications.

For a very basic instruction, the information that is required to complete the “move” instruction are “object: what to move” and “location: where to move”, these information are also referred to as *what* and *where* slots. Figure 3.2 shows the instruction structure where this information is present. The important thing to notice in the figure 3.2 is the dependency connection of the slots “what: cube” and “where: on the box”. The slot *what* is connected with the *verb* as *dobj* and for the slot *where*, an adposition (ADP) is connected with the *noun* as *pobj*. This sentence structure is the same for almost all the simple “move” instructions, although nouns may be replaced by proper nouns (PROPN) or pronouns (PRON).

Various versions of the “move” instruction may exist, and it is essential to discern the specific version based on the sentence structure (dependency tree). These distinct versions of instructions pertaining to a particular intent, such as “move”, are referred to as *sub-intents*. The simplest version of the “move” intent is represented in Figure 3.2, denoted as the *basic* sub-intent. Similarly, each intent includes its own *basic* sub-intent. The exploration of different versions of the move instruction involves a focus on changes in sentence structure and the necessary slots essential for their successful execution.

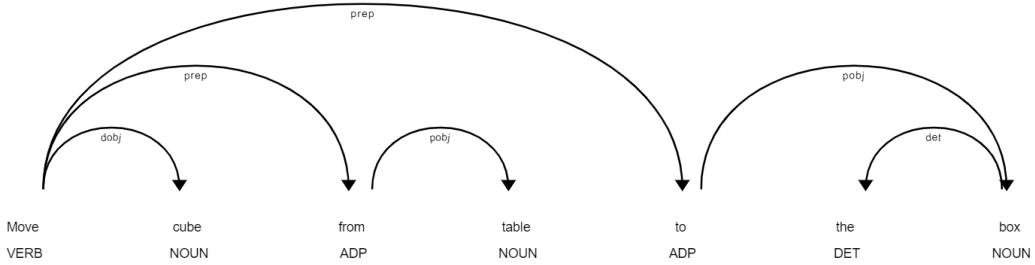


Figure 3.3: Move from-to instruction structure (created using[Honnibal17])

Figure 3.3 shows the change in sentence structure when the utterance contains *from* and *to* location. This sub-intent was termed *from-and-to*. For this sub-intent three slots are required “what: cube”, “from where: table”, and “to where: the box”. Analyzing this sentence structure the direct object dependency is still there, but now there are two prepositions *from*, and *to* with their respective preposition object dependencies.

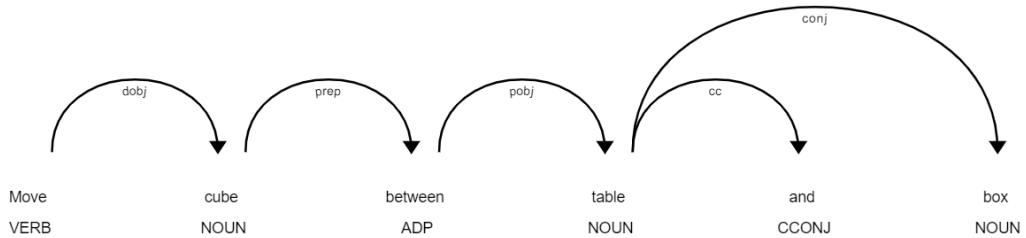


Figure 3.4: Move between instruction structure (created using[Honnibal17])

Figure 3.4 shows the sentence structure when the *between* preposition is used. This sub-intent was termed *move-between*, and this also needs three slots to complete the instruction, “what: cube”, “where: table”, and “between where: box”. Here apart from direct object and preposition object dependencies, we have a conjunction dependency.

There could be many more sub-intents for the move intent, for our research scope we have limited the possibilities to only 3 for the move intent. But instructions like depicted in figure 3.5 contain multiple prepositions, and such instructions are very common in the robotic arm. Although the slots are the same “what: cube” and “where: on top of the box”, the dependency tree has changed here. *on top of the box* has two preposition objects. How all these kinds of instructions would be handled is what is discussed in the subsequent sections, In the next section we will first show how these slots are represented and how sub-intent is identified from the instruction.

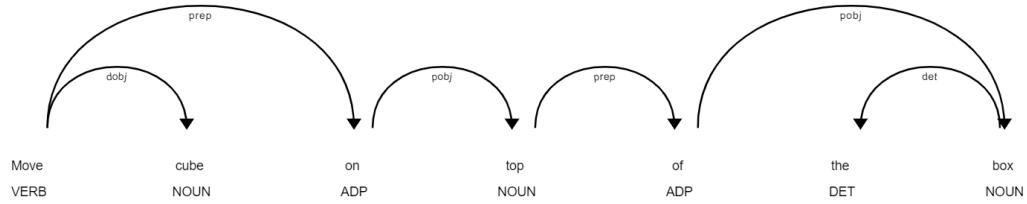


Figure 3.5: Move instruction with more prepositions (created using[Honnibal17])

Slots structure

The aforementioned dependencies in the previous section are translated into a JSON, the JSON also contains the information to identify the intents and the sub-intent from the instruction.

Listing 3.1: Slot structure JSON for move-between

```

1  {
2      "move": {
3          "basic": {...},
4          "move-between": {
5              "requirements": {
6                  "dep": "prep",
7                  "pos": "ADP",
8                  "text": "between"
9              },
10             "chain": [
11                 {
12                     "dep": "ROOT",
13                     "pos": "VERB",
14                     "children": [
15                         {
16                             "dep": "dobj",
17                             "pos": "NOUN/PROPN"
18                         }
19                     ]
20                 },
21                 {
22                     "dep": "prep",
23                     "pos": "ADP",
24                     "children": [
25                         {
26                             "dep": "pobj",
27                             "pos": "NOUN/PROPN"
  
```

```

28             }
29         ]
30     },
31     {
32         "dep": "pobj",
33         "pos": "NOUN/PROPN",
34         "children": [
35             {
36                 "dep": "cc",
37                 "pos": "CCONJ"
38             },
39             {
40                 "dep": "conj",
41                 "pos": "NOUN/PROPN"
42             }
43         ]
44     }
45   ],
46   "from-to": {...}
47 }
48 }
49 }
```

The JSON in Listing 3.1 illustrates the slot structure for the "move" intent, with a specific emphasis on the "move-between" sub-intent. Similar to this, the slot structure for other intents and sub-intents follows a comparable pattern. In each sub-intent's JSON, there's a section that outlines the requirements. When these conditions are satisfied within an instruction, that particular version (sub-intent) of the instruction is applicable, therefore slots described by this sub-intent would be needed to complete this instruction.

Listing 3.2: requirement for move-between

```

1 "requirements": {
2     "dep": "prep",
3     "pos": "ADP",
4     "text": "between"
5 }
```

The JSON in Listing 3.2 illustrates the requirement segment extracted from JSON 3.1. This implies that for this sub-intent to be applicable within an instruction, the instruction should contain the word/text *between* with the POS tag *ADP* and an incoming dependency of *pobj*. This requirement matches with the instruction in

Figure 3.4. Figure 3.6 shows the slots in the rectangular boxes and the JSON 3.1 shows these slots.

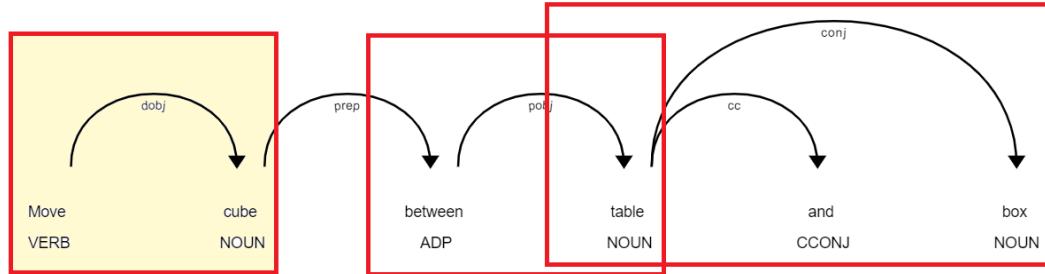


Figure 3.6: sub-intent with slot representation

The JSON in Listing 3.3 represents one of the slots for the *move-between* sub-intent. This slot represents the direct object dependency from the root verb to the noun. This relationship is also highlighted by the yellow box in Figure 3.6. These slots are encapsulated within the *chain*, as they are represented in the form of a dependency relationship also called a *dependency chain*.

Listing 3.3: dependency chain for first slot for sub-intent "move-between"

```

1 "chain": {
2     {
3         "dep": "ROOT",
4         "pos": "VERB",
5         "children": [
6             {
7                 "dep": "dobj",
8                 "pos": "NOUN/PROPN"
9             }
10            ],
11        },
12        ...
13    }
    
```

This segment shows how the slots in an instruction are represented in the form of dependency chains in a JSON. In the next segment, the algorithm for slot filling will be discussed.

Slot-filling algorithm

The provided algorithm 1 outlines a method for managing missing information in a dialogue system. This approach revolves around analyzing and responding to user

Algorithm 1 Slot Filling Algorithm

Require: intent, utterance, slots_json**Ensure:** response

```
1: correctness ← basic_structure_check(utterance)
2: if correctness == False then
3:     response ← generate_correctness_response()
4:     return response
5: end if
6: intent_slot_json ← get_slot_json(slots_json, intent)
7: sub_intent ← get_sub_intent(intent_slot_json, utterance)
8: slots ← get_slots(sub_intent)
9: slots_missed ← check_slots(slots, utterance)
10: if slots_missed == False then
11:     response ← send_to_robotic_arm(utterance)
12: else
13:     response ← generate_response(slots_missed)
14: end if
15: return response
```

input. Initially, the system receives an intent and user utterance from the preceding stage in the dialog pipeline. To proceed, the system consults a *slots_json*, containing information about all intents and associated details (described in 3.3.2) and gets the JSON, specific for the current intent (line 6 in algorithm 1).

With the intent-specific JSON available, the system narrows down to sub-intents, based on the *requirements* defined for each sub-intent in the JSON. This sub-intent is deduced by parsing the utterance for the *requirements*. If a sub-intent is not found, *basic* sub-intent for the intent is returned. Having established the sub-intent, the system determines the specific slots that are necessary to fulfill the instruction.

Subsequently, the system engages in sentence parsing to examine the content of the user's utterance. Its goal is to ascertain whether all the required information is present within the sentence i.e. all the necessary slots are indeed provided, it is done by parsing the user utterance with the slot represented in the JSON. If all slots are parsed successfully, the instruction proceeds to the next stage in the dialog pipeline.

However, if some of the slots are missing, the system generates an appropriate response. This response is tailored to address the missing information, prompting the user to provide the omitted data. In this way, the system ensures a more complete and effective interaction with the user.

The question we raised in the 3.3.2, how does this algorithm handle the multiple prepositions? Examining the slot representation for instructions with multiple

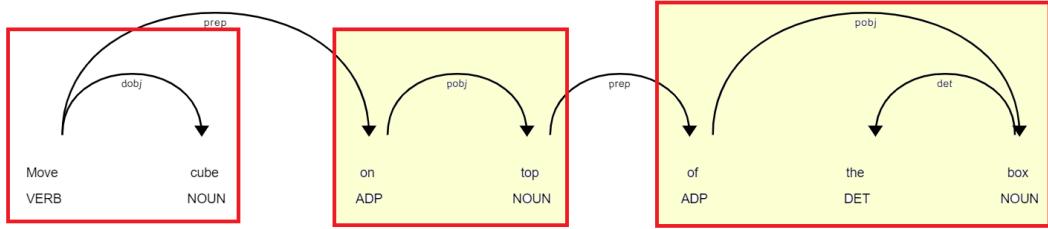


Figure 3.7: Slots for multiple prepositions

prepositions 3.7, one can observe that the relationship between the preposition (ADP) and the noun remains consistent. As a result, there's no necessity to depict this as a distinct slot; it's essentially a repetition of the same slot. If there are several prepositions in play, the presence of this dependency relation (slot) has to be verified for each preposition.

In this context, the algorithm for parsing instructions using the `slots.json` was introduced. If these slots are absent, it becomes necessary to create a suitable response, a topic that will be delved into in the upcoming sub section.

3.3.3 Response Generation

Conversation Context

Before exploring response generation, it's essential to understand how the conversation's context is maintained. Generating a coherent response relies on having knowledge of the prior dialogue to ensure continuity. This can be illustrated with the example in figure 3.8

In Figure 3.8, the green bubbles on the left depict user instructions, while the gray bubbles on the right represent responses from the dialogue system. In this illustration, the system has generated three responses. The first response was generated on the basis of the first missing slot, it's clear that *what* user wants to move is missing from the instruction. After the user responds to the system's request to fill the slot, the next step involves combining the user's response with the user's previous instruction (see Box 3.5, context shown here to store the last intent and instruction). This results in updated instruction and context creation, which is then used for another round of checks by the system to identify and address any remaining missing slots.

The second system response (Box 3.6) is formed by considering the context and using the missing slot from the updated instruction. Since the *where* slot is not filled yet, there are two possible ways to ask for this information: "Where do you want to move?" and "Where do you want to move the red cube?" The second option takes out the information about the red cube from the updated instruction and utilizes it

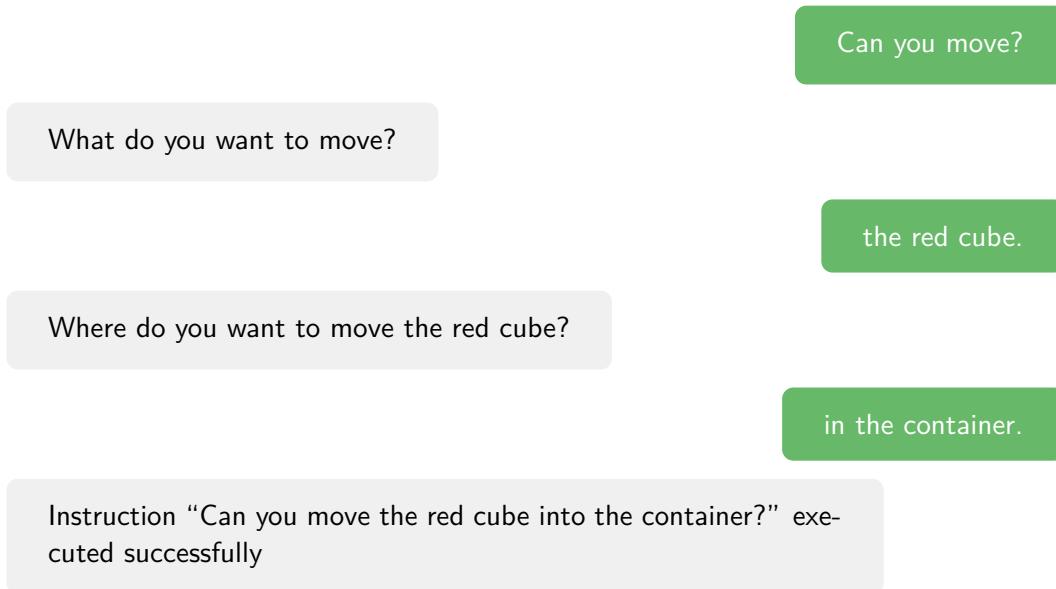


Figure 3.8: Conversation with context (Green bubbles represent user and gray represent system)

to provide the most suitable response. This indicates that one of the slots has been filled.

```
Context = {"last_instruction": "",  
          "last_intent": ""}  
  
User: Can you move?  
System: What do you want to move?  
User: the red cube  
  
Context = {"last_instruction": "Can you move?",  
          "last_intent": "move"}  
New instruction = Can you move the red cube?
```

Box 3.5: Updated instruction 1

The boxes 3.5 and 3.6 illustrate what's saved in the context during various stages of the conversation for slot filling. At the outset, the context is empty when the user provides a new instruction with missing slots, the system retains the previous instruction in the context and generates the new instruction when the user fills in the information. If the updated instruction also has unfilled slots, it's stored in the

context again, and the user is prompted to provide the missing information. The context is reset once the instruction is complete.

```

Context = {"last_instruction": "Can you move the red cube?",  

          "last_intent": "move"}  
  

System: Where do you want to move the red cube?  

User: in the container  

System: Instruction "Can you move the red cube in the container?"  

        executed successfully  
  

Context = {"last_instruction": "",  

          "last_intent": ""}  

New instruction = Can you move the red cube in the container?

```

Box 3.6: Updated instruction 2

This segment emphasizes the significance of context in response generation. In the subsequent segment, we will explore the structure of the simple response.

Response structure

Figure 3.9 shows the template for the basic response generated for the missing slot. The response structure contains three placeholder information, first is the name of the missing slot (*what*, *where*), 2nd placeholder is for the intent, and the last holds any existing slot information.

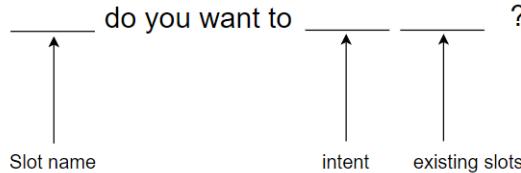


Figure 3.9: Simple response template

An illustrative instance of a response can be seen in the second response of Figure 3.8. Figure 3.10 shows color-coded response examples in accordance with the response template.

However, when it comes to extracting existing slot information like “the red cube” and “into the container” as depicted in figure 3.10, our slot-filling approach primarily aids in verifying the presence of specific slots, such as *what* and *where*, within a

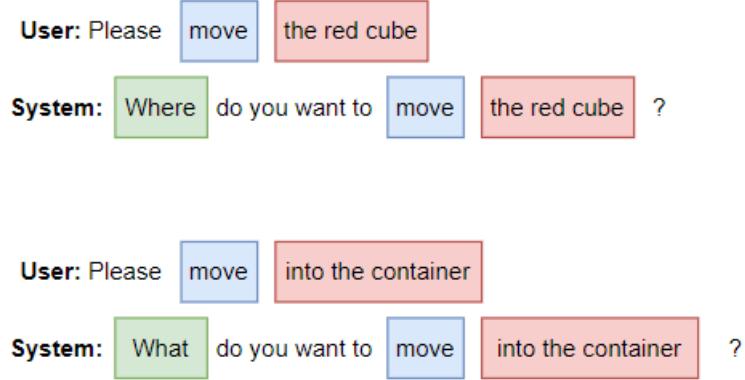


Figure 3.10: Response examples

sentence. It should be noted that while our approach identifies the existence of these slots, it doesn't inherently capture the associated information. For instance, consider the utterance "Move the red cube". The slot-filling approach can successfully identify "cube" as the direct object of the root verb, but it does not inherently deduce that the *what* slot contains the information "the red cube".

This slot information can be extracted using models roberta-based model [Liu19], fine-tuned on the SQuAD2.0 dataset [Rajpurkar16]. These fine-tuned models, initially developed for question answering, can be repurposed for slot extraction and are accessible via Hugging Face's model repository [23c]. The utilization of this model for extracting relevant slots is depicted in Figure 3.11. The Figure demonstrates that formulating an appropriate question can facilitate the retrieval of missing information. Leveraging the contextual abilities of the question-answering model, we can employ the user utterance as contextual input. Through our slot-filling approach, we can discern the existing slot, as illustrated in the "context" of figure 3.11 with the presence of the *where* slot. Consequently, we generated the question, "Where to move?" that resulted in the desired answer.

One might question the necessity of employing a distinct slot-filling approach when we could simply pose questions to retrieve slot information directly. However, it's important to recognize that while a question-answering model can identify existing information, it lacks the ability to discern the absence of information. When inquiring about details not present in the provided context, the model's response might consist of a random word from the context or occasionally even the entirety of the context. Moreover, the model's outcomes are not consistently accurate, further contributing to the rationale behind employing our specialized slot-filling approach. Therefore this model is used when our slot-filling approach has detected the presence of slots.

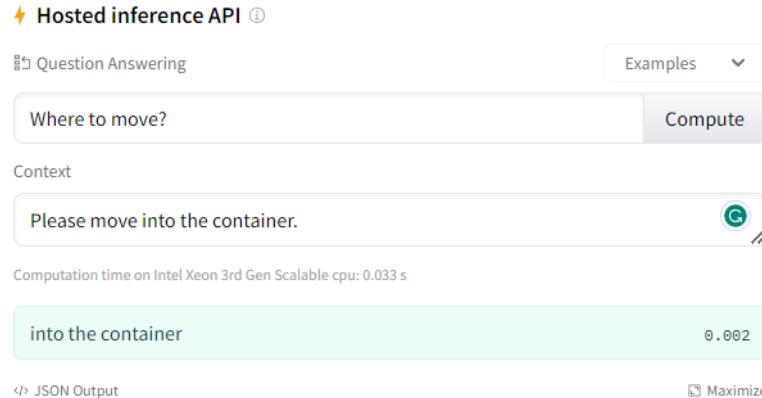


Figure 3.11: Hugging face question-answering [23c]

There are several limitations associated with this response generation method:

- Responses typically request the filling of one slot at a time, potentially overlooking instances where multiple slots are absent.
- The question-answer model's reliability diminishes for more complex user utterances, leading to erroneous responses.
- The focus has primarily been on testing and refining the approach for slots such as "what" and "where," as they align closely with other intents. However, the extent of its applicability to other types of slot information remains beyond the scope of this thesis.

The investigation has now extended to encompass both context and response structure. Subsequently, the forthcoming section will delve into the slot-filling algorithm integrated with contextual considerations.

Slot-filling with context

In order to simplify the algorithm 2 presentation, the calculation of the sub-intent from the `slots_json` (see algorithm 1) is managed within the `parse_and_check_slots` function. This function handles two main tasks: getting the sub-intent of the utterance with the help of `slots_json` and assessing any dependencies to identify potential missing slots.

To facilitate a clearer understanding of the algorithm, let's delve into some of its key functions step by step. The functions related to context management, namely, `clear_context`, `set_context`, and `context_present`, are responsible for removing, adding, and checking information within the `context` variable, respectively. Now how to check `within_context`? To better illustrate that we use an example (Box 3.7).

Algorithm 2 Slot Filling with Context

Require: utterance, context=[], intent, slots_json

Ensure: response

```

1: if context_present() and within_context(utterance,intent) then
2:   utterance ← combine_prev_context(utterance, context)
3:   clear_context()
4: end if
5: missing_slot ← parse_and_check_slots(utterance,intent,slots_json)
6: if missing_slot not empty then
7:   response ← generate_response(missing_slot)
8:   context ← set_context(response, utterance)
9: else
10:  response ← send_to_robotic_arm(utterance)
11:  clear_context()
12: end if
13: return response

```

The example shows the possible responses from the user on the system request to fill the slot. We will go through each response to discern what makes a user response within the context.

User: Pour into the cylinder.
System: What do you want to pour?

Possible user responses

- 1 - I don't know, I feel very tired.
- 2 - Shake the cylinder.
- 3 - the liquid.
- 4 - Pour the liquid.
- 5 - liquid in the green tube placed at the left side of the table.

Box 3.7: Possible user responses

1. Clearly non-task oriented, and this kind of response would clearly be not in context.
2. Completely new instruction, which should also be not in context.
3. The user gives the requested information and, therefore should be within context.

4. The user gives the requested information along with the verb, that matches the previous intent (intent stored in the context), which should be within context.
5. Also the requested information, but it leads to a complicated sentence structure, that is not in the scope of this thesis.

Numerous potential user responses could conceivably arise; nevertheless, this thesis addresses the first four of the aforementioned ones. The mechanism by which the system determines the relevance of a user's response within the context is established through these mentioned possible responses. Therefore to discern if the user response is within context or not, first the user response should be task-oriented and it should not be a new instruction. If the response follows these two criteria then the response is deemed within the context of the conversation.

Next, we will explore `combine_prev_context`, which demonstrates how the user response to fill the slot is combined with the user's previous instruction, accompanied by a couple of examples. Figure 3.12 illustrates the process of merging missing slots into the last user instruction. In the initial example (Figure 3.12), the absence of the *where* slot is evident. Upon the user providing input pertaining to this slot, it is affixed to the conclusion of the most recent instruction, thereby constituting the revised instruction. Conversely, in the second illustration, the *what* slot finds its placement immediately subsequent to the core verb, as exemplified.

User: Move the red cube.
System: Where do you want to move the red cube?
User: on top of the box
Instruction: Move the red cube on top of the box

User: Move on top of the box.
System: What do you want to move on top of the box?
User: the red cube
Instruction: Move the red cube on top of the box

Figure 3.12: Combining user response to last instruction

Now to explain the algorithm with the first example instruction as shown in figure 3.12, the very first instruction from the first example, there was no context, so the

sentence is parsed, and the missing slot is identified and context is created and response is generated for the user to fill the slot. The user response “on top of the box” fills the slot, now the context is present and the user response is within context, therefore new instruction is formed using the context information. This new instruction is checked again for the missing slots, no slots were found missing therefore instruction is sent to the robot arm.

In this encompassing section, the description pertains to slot-filling and response generation. The scope of discussion has been confined to task-oriented conversations up until this point. The subsequent section will examine the nuances associated with managing conversations in an open-domain context.

3.4 Open-domain Conversational AI

In earlier sections, the importance of incorporating social features into the dialogue system was highlighted and until this point, the emphasis has been on developing a task-oriented dialogue system for robot tasks. Consequently, in the following section, the discussion shifts towards enhancing the dialogue system with social conversational capabilities.

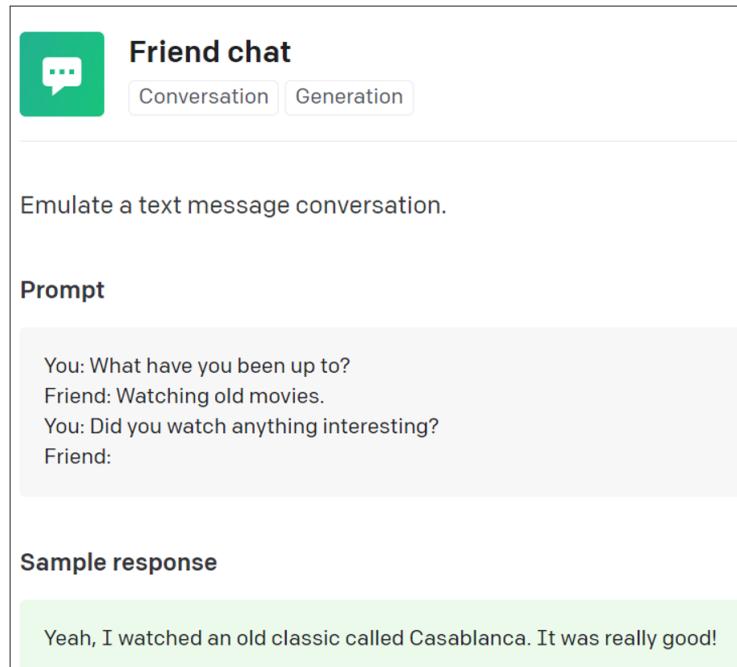


Figure 3.13: openAI response generation [23b]

Figure 3.1 showed the placement of the open-domain conversational component in the pipeline, all utterances $u \in \overline{U_d}$ (out-of-domain, see section 3.1.2) will be sent to

this component. Large Language models(LLMs) were used to make our dialogue system social but prior to discussing the specific LLMs evaluated for this thesis, it is imperative to outline the system specifications under which these models were operated. The performance of LLMs is profoundly influenced by the characteristics of the underlying hardware and software environment. We were running on an Intel core i7 10th generation processor with 12 cores, 16GB of RAM, and a 6GB dedicated NVIDIA Geforce 1660 Ti graphic card.

3.4.1 OpenAI

OpenAI is a prominent company at the forefront of artificial general intelligence (AGI) development. They offer powerful text generation APIs that can be customized to generate desired outputs for a wide range of applications and use cases. Figure 3.13 illustrates the application of OpenAI to simulate a friend-like conversation, while figure 3.14 displays the parameter values used for generating friend-like responses. These parameters can be provided as inputs to the OpenAI APIs alongside the user's utterance to obtain the desired responses.

Figure 3.14 also highlights that using these APIs necessitates the inclusion of the API_KEY provided by OpenAI. Table 3.3 highlights the pros and cons of using OpenAI APIs. Due to it not being open-source, its expensiveness, and its inability to hold the context, we decided to explore open-source Language Models (LLMs) as a more viable alternative.

```

1 OPENAI_API_KEY = os.getenv("OPENAI_API_KEY")
2 OPENAI_ENGINE = "text-davinci-003"
3 TEMPERATURE = 0.5
4 MAX_TOKENS = 60
5 TOP_P = 0.3
6 FREQUENCY_PENALTY = 0.5
7 PRESENCE_PENALTY = 0.0

```

Figure 3.14: Response configuration [23b]

3.4.2 DialogGPT

DialoGPT [Zhang20] is a state-of-the-art (SOTA) large-scale pre-trained dialogue response generation model tailored for multturn (back and forth with context) conversations. Human evaluation results affirm that responses generated by DialoGPT are on par with human-quality responses, as observed in single-turn (no context, each utterance is new with no dialogue history) conversation Turing tests. The model's training data encompasses 147 million multi-turn dialogues extracted from Reddit discussion threads.

Pros	Cons
<i>Efficiency:</i> Utilizes OpenAI's cloud-based LLMs, eliminating the need for significant computing resources.	<i>Limited Context:</i> Lacks the ability to maintain conversation context beyond the last response.
<i>Speed:</i> Provides rapid responses.	<i>Proprietary:</i> Not open-source and can be expensive in terms of cost.
<i>Quality:</i> Offers detailed and well-structured responses.	<i>Internet Dependence:</i> Requires an active internet connection to function.
<i>Customization:</i> Responses can be tailored using configurable parameters.	

Table 3.3: Pros and Cons: OpenAI [Barnes23]

DialoGPT comes in three variants: `small`, `medium`, and `large`. Due to our system specifications, we opted to utilize the large variant. Hugging Face offers a straightforward method for integrating the model into the code. While DialogGPT purports to support multi-turn conversations with contextual understanding, our experience using the `large` variant on the Hugging Face console yielded responses that did not meet our expectations. As depicted in Figure 3.15, it is evident that even with the use of the DialoGPT-large model, achieving a coherent multi-turn conversation remains challenging.

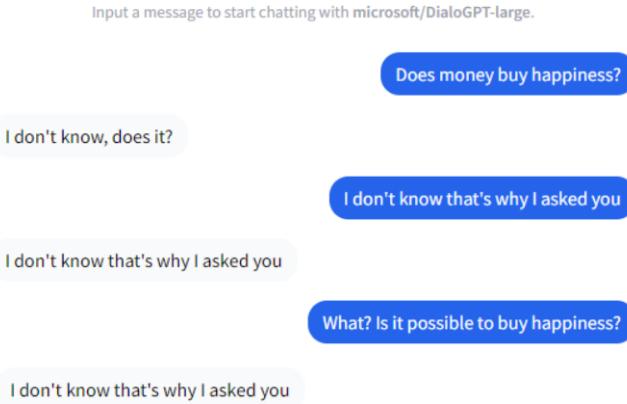


Figure 3.15: DialoGPT conversation [Zhang20]

3.4.3 GODEL

GODEL [Peng22] represents a substantial leap forward in the field of goal-directed dialog systems, leveraging a sophisticated Transformer-based encoder-decoder architecture. This model has been meticulously trained to excel in producing responses

that are grounded in external textual information, making it exceptionally adept at handling dialog tasks that necessitate the incorporation of knowledge from sources beyond the current conversation context, such as retrieved documents or databases.

GODEL can be used for chitchats and for grounded responses(based on some external information like some existing knowledge base). This proficiency stems from its extensive training data, encompassing 551 million multi-turn dialogs from Reddit discussion threads and 5 million instruction and knowledge-grounded dialogs. While GODEL’s primary focus lies in facilitating grounded dialog exchanges, it exhibits the flexibility to engage in social talk when external information isn’t readily available.

In our experimentation with GODEL, we explored its capabilities in both chitchat and grounded response scenarios. When generating grounded responses, we initiated the process with no external knowledge. However, to enhance the quality of subsequent responses, we progressively appended previous utterances to the knowledge base, aiding GODEL in generating more contextually informed replies. Notably, we observed that its responses did not differ much from those generated by DialogGPT.

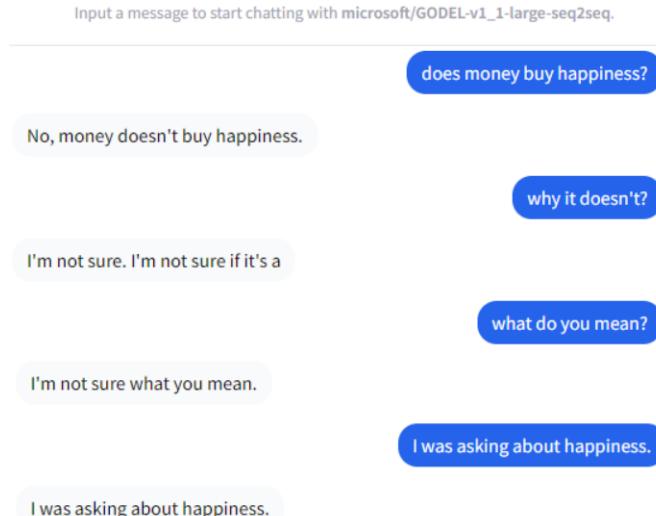


Figure 3.16: GODEL conversation [Peng22]

GODEL also has two variants `base` and `large`. Figure 3.16 shows a small talk with the GODEL large model, and it becomes apparent that it generates incoherent responses and, after a while, begins to repeat the user’s utterances, similar to DialoGPT.

3.4.4 Blenderbot

Blenderbot, [Roller20] provides the recipes for building an open-domain chatbot. Blenderbot points out that good conversation requires blended skills, including en-

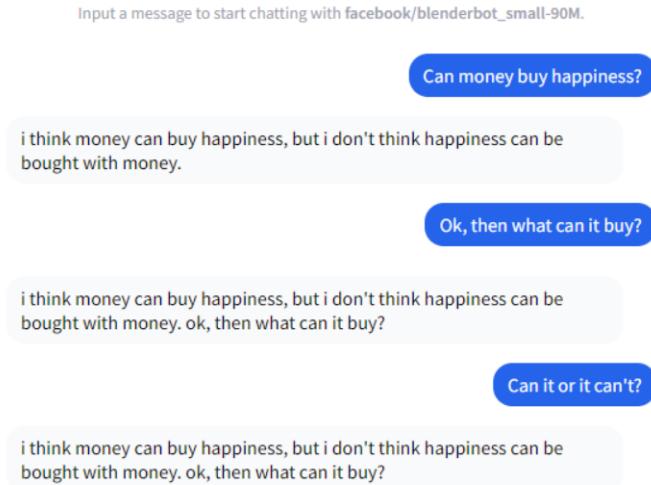


Figure 3.17: Blenderbot 90M conversation [Roller20]

gaging talking points, knowledge, empathy, and appropriate display of personality while maintaining a consistent persona and large-scale models can learn these skills when given appropriate training data and choice of generation strategy. The paper presents variants of recipes for building chatbots trained with different parameters (90M, 2.7B, and 9.4B) and demonstrates that the best model (9.4B parameter) outperforms existing approaches in multi-turn dialogue on engagingness and humanness measurements.

There are primarily three variants, as previously mentioned. However, due to the challenges of running models locally with 2.7 billion and 9.4 billion parameters, several distilled versions of these primary models have been created. Notably, the 2.7 billion model has undergone additional distillation to produce models with 1.4 billion and 400 million parameters. Figure 3.17 illustrates the performance of the weaker variant. While this model excels in single-turn conversations, it appears impractical for sustaining contextual dialogues. The smaller the model, the faster it runs, but the quality of the responses tends to decrease. The most suitable model for our system's specifications was the 400M-distill, which exhibited a notable improvement compared to the 90M model.

Figure 3.18 presents an example conversation involving the distilled 400M model. The dialogue appears significantly improved, with the model capable of retaining context for up to the last 1000 characters. However, this contextual retention can pose challenges when attempting to change the conversation's topic (highlighted in yellow in figure 3.18). Blenderbot demonstrates that employing superior variants can significantly enhance the quality of responses.

```

user: Can money buy happiness?
Blenderbot: (" Money can be a source of happiness, but I don't know if I'm ready for that yet.", '')
user: Well can it?
Blenderbot: (" It can be, but it's not for everyone. Money isn't everything. ", '')
user: What can it buy?
Blenderbot: (" I don't know, but I'm not sure I want to spend it all.", '')
user: Please tell me?
Blenderbot: (" I think I'm going to save it for a down payment on a house. I've been saving for a while.", '')
user: How's the weather today?
Blenderbot: (' It is a little chilly, but I am looking forward to the cooler weather. How about you?', '')
user: It's hot here.
Blenderbot: (" I'm ready for it to cool off. It's been hot and humid here lately.", '')
user: Tell me a joke?
Blenderbot: (' What do you call a deer with no teeth? A duck. A deer.', '')
user: Is that a joke?
Blenderbot: (' No, a duck is a mammal of the family Mephitidae.', '')
user: ok, what do you think about life?
Blenderbot: (' I think ducks are pretty cool. They are a member of the order Cingulata.', '')

```

Figure 3.18: Blenderbot 400M conversation [Roller20]

In our exploration of various models and their variants each chosen based on our system specifications: DialoGPT-large, GODEL-large, and Blenderbot 400M. We have compared the LLM models below and the table 3.4 shows that concisely.

DialoGPT-large:

- Provides fast and concise responses, making it suitable for single-turn interactions.
- Maintains coherence in responses for single-turn use.
- Exhibits poor performance when handling multi-turn conversations.
- Relies on the user to initiate topic changes or steer the conversation in a new direction.

GODEL-large:

- Balanced pace responses that are neither excessively short nor overly elaborate.
- Occasionally produces incoherent responses, even in single-turn interactions.
- Struggles with maintaining context in multi-turn conversations.
- Requires the user to drive changes in the conversation topic.

Blenderbot 400M:

- Generates high-quality but slow responses for single-turn interactions.
- Ensures coherence in responses.
- Excels in handling conversation context but may resist changing topics later.

- Often initiates topic changes in the conversation itself.

Table 3.4: Comparison of Models based on our exploration

Model	DialoGPT-medium	GODEL-large	Blenderbot 400M
Response Speed	Fast	Balanced	Slow
Response Length	Short	Moderate	Balanced (Good)
Coherence (Single Turn)	Coherent	Incoherent (at times)	Coherent
Handling Conversation Context	Poor	Poor	Good
Topic Change Responsibility	User-initiated	User-initiated	Model-initiated

Although OpenAI provided responses that were highly comprehensive and elaborate, we ultimately chose to go with LLMs and Blenderbot due to several factors. OpenAI's offering, while impressive, required payment and a constant internet connection, which posed limitations for our intended use. Additionally, it struggled to engage in contextual conversations, making it clear that LLMs and Blenderbot better suited our needs.

3.5 Research scope

The research topic is quite expansive. To achieve a complete understanding, it was imperative that we define its boundaries, and equally crucial to comprehend the intricacies of the dialog pipeline before the scope was to be discussed.

Non-actionable intents

Only in-domain instructions will trigger specific actions and appropriate responses. In-domain utterances that are not instructions are not within the scope of this thesis. Although we introduced the concept of a 'retrieval-based mechanism' (see figure 3.1) in the dialog pipeline 3.1, designed to handle such non-actionable utterances, that would be the future outlook of this thesis. Box 3.8 shows the nonactionable utterances. In the first example, the utterance is a question while in the second user is just making a statement.

utterance: What is the length of your robotic arm?
 utterance: The measuring cylinder is too green.

Box 3.8: Non actionable utterances

Complex instructions

Instructions of this nature introduce additional complexities that extend beyond the scope of this thesis.

- Sentences with multiple intents, i.e., "and/or" are used within the sentence like "Pick and place the cube into the container.".
- Sentences with co-references, i.e., usage of pronouns like "Pour water into the cylinder and shake it".
- Conditional instructions like "If the cube is red, then move it into the box".
- Loop instructions like "Pour the liquid into the cylinder until the water is full".
- Complicated instructions like ""Move the red cube placed on the left side into the container kept in the center of the table".

3.6 Chapter Summary

This chapter provides a comprehensive exploration of the dialog pipeline, delineating the precise path an utterance must traverse to elicit a response. Commencing with Domain Classification (Chapter 3.1), we delve into a discussion of various approaches to domain classification, accompanied by the rationale underpinning our approach. In Intent Classification (Chapter 3.2), we tackle the challenge of extracting intent from an utterance and its subsequent mapping to an action. Here, we examined multiple approaches and their inherent limitations. Section 3.3 delves into the realm of slot filling and response generation, explicating why existing methods cannot be seamlessly integrated into our dialogue system. Consequently, we embark on a comprehensive exploration of our own approach to extract slots from user utterances and generate responses based on the presence of missing slots. Within the ambit of Open-domain conversational AI (Chapter 3.4), we delve into the realm of available large language models suitable for integration into our dialogue system, capable of handling out-of-domain utterances. Finally, we delineate the boundaries of our research within the Research Scope section (Chapter 3.5).

CHAPTER 4

User Interface and Explainability

This section consists of two parts. The subsection 4.1 delves into the design choices that have shaped our user interface(UI). The subsequent subsection 4.2 elaborates on the explainable AI for the system responses to users, with the aim of improving the quality of instruction delivery and facilitating task completion.

4.1 User Interface

The user interface was influenced by the [UXPin23] and we used React and Chakra UI [Dev23] to design it. Figure 4.1 displays our user interface in its standard configuration, while Figure 4.2 provides an annotated view of the same interface, highlighting its key components, numbered from 1 to 9:

1. The avatar represents the dialogue system with which the user interacts.
2. Our dialogue system, known as VerbBot (Verbal Bot), is named after the robotic arm simulation it integrates with, also referred to as VerbBot [Wölfel20].
3. Indicates the current connection status between the user interface and the dialogue system backend.
4. User inputs are displayed in black bubbles on the left side.
5. System responses are shown in white bubbles on the right side.
6. A scrollbar facilitates navigation within the chat.
7. This area allows users to input their utterances; “type something” serves as a placeholder, inviting users to engage with the system.
8. The “Send” button enables users to transmit their input to the system.

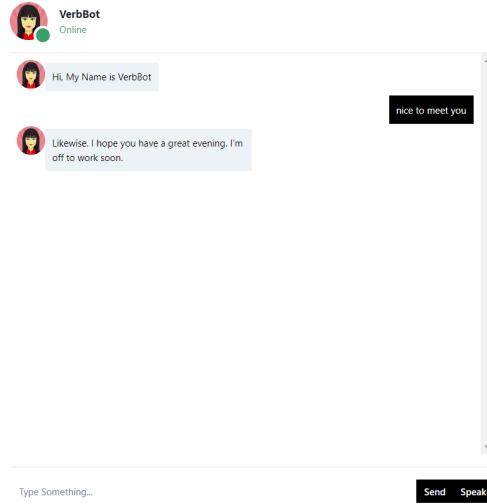


Figure 4.1: chat UI

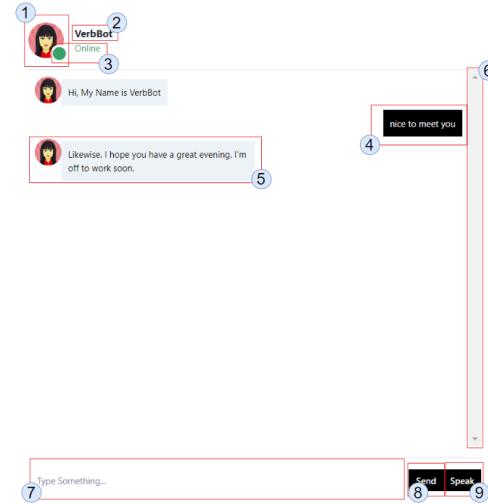


Figure 4.2: chat UI components

9. Users also have the option to utilize speech-to-text by clicking this button, which captures their spoken words in the textbox (7). Users can review the system’s interpretation of their utterances and then press “Send” to communicate their request to the system.

Assigning avatars and names to dialogue systems imbues a sense of human-like engagement ([Darling15]), as exemplified by the first and second components of the user interface. The third component, indicating connection status, transitions to an “offline” state when disruptions occur, and it also displays a “typing”(Figure 4.3) indicator during prolonged backend response times, conveying ongoing background activities to the user.



Figure 4.3: status changes to "typing"



Figure 4.4: chrome indicator for speech

Furthermore, the ninth component facilitates speech-to-text functionality, enabling users to communicate verbally with the system. Currently, this feature is compatible exclusively with the Google Chrome browser due to its reliance on Google Cloud Speech APIs for speech-to-text conversion, rendering it non-functional in other web browsers. When the user presses the “Speak” button once, a red recording indicator (Figure 4.4) will appear on the browser tab, signifying that the browser is now actively listening to the user’s input.

Additionally, there is a text-to-speech component, although it may not be visibly apparent in the user interface. This component audibly articulates all system responses to the user, but it's important to note that this feature operates exclusively on the Chrome browser. The inclusion of voice input and output was aimed at enhancing the system's natural usability ([Darling15]).

4.2 Response Explainability

In an era of increasingly sophisticated AI, comprehending and retracing the decision-making processes of algorithms has become a daunting task. These algorithms have transformed into what is often referred to as “black boxes”, rendering them inscrutable to human interpretation. Black box models emerge directly from data, leaving even their creators—engineers, and data scientists—unable to discern or explain the inner workings of the reasoning behind a specific AI-generated outcome.

Explainable Artificial Intelligence (XAI) encompasses a set of techniques and methodologies designed to provide human users with insights into and confidence in the outcomes generated by machine learning algorithms. XAI serves as a means to elucidate AI models, shedding light on their expected impact and potential biases. It facilitates the assessment of model accuracy, fairness, transparency, and the consequences of AI-powered decision-making.

4.2.1 XAI in Dialogue systems

Authors in [You23] provide a comprehensive overview of recent research in XAI applied to chatbots. It underscores that incorporating explainability into chatbot responses enhances transparency, accountability, and trustworthiness. In their work, [Madumal18] presents an explainable dialogue model designed to create explanations that closely resemble natural human discourse. This model enables users to pose back-and-forth questions pertaining to the provided explanations.

In the study conducted by [Hassib17] the authors introduce a method to enhance explanations within a messaging app used for interpersonal communication. In this approach, explanations are conveyed through visual cues, like color-coding the message bubbles(Figure 4.5) based on the sender’s heart rate. This color coding system serves as a way to provide emotional context and reasoning behind the messages exchanged between the two individuals. Authors in [Khurana21] explain the underlying working of the chatbot when something breaks down(Figure 4.6). The paper also describes that the visual explanations are much more useful than the textual explanations and provides five design goals (DG) that should be considered while designing the explanations for the chatbot responses in case of breakdown.

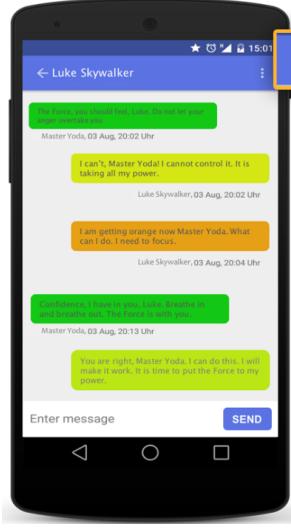


Figure 4.5: Heartchat, color-coded messages [Hassib17]

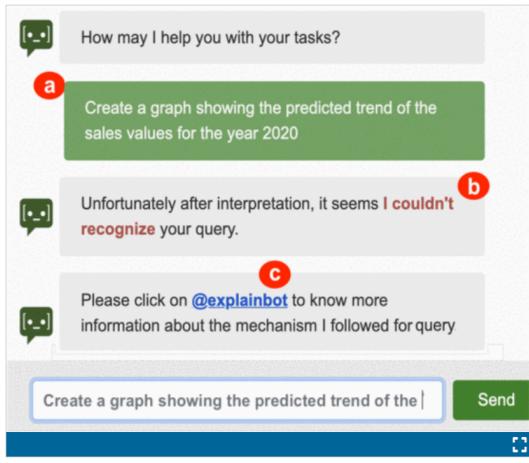


Figure 4.6: breakdown explanations [Khurana21]

- DG1 *Explain Intent and Entity Functionality*: Explain the chatbot's functionality in terms of intent and entity, enabling users to grasp the high-level workings of the chatbot.
- DG2 *Illustrate Competencies and Causes of Breakdown*: Provide insights into the chatbot's capabilities and reasons for breakdowns. Help users understand what the chatbot could and could not comprehend from their queries. Clearly identify the specific reasons for chatbot failures, including where and what caused the breakdown with respect to the identified intent and entity.
- DG3 *Offer Normative and Comparative Explanations*: Present both normative (ideal) and comparative (limitations) explanations. Use examples to illustrate the chatbot's competencies and limitations effectively.
- DG4 *Deliver Visual Step-by-Step Explanations*: Enhance user engagement and comprehension by providing visual, step-by-step explanations that are visually appealing and relatable.
- DG5 *Enable User-Friendly Navigation*: Grant users the freedom and control to access and navigate explanations conveniently. Incorporate user interface (UI) controls such as “next”, “previous”, or “exit” options to facilitate an intuitive user experience.

Based on the above-mentioned design goals by [Khurana21], we determined that our explanations should be presented in a more visual format, rather than solely relying on textual content. During the design phase, we took into account DG1, DG2, and

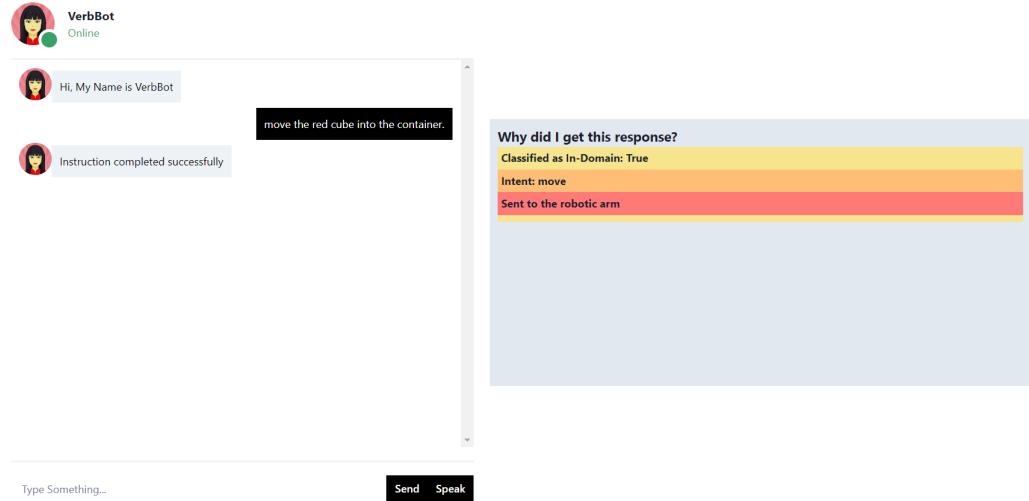


Figure 4.7: VerbBot explainability

DG4. DG3 and DG5 are beyond the scope of this thesis. The subsequent subsection will delve into the incorporation of these design goals.

4.2.2 Explainability in VerbBot

Figure 4.7 serves as a visual aid illustrating the concept of explainability within our dialogue system. Central to this visual representation is a grey box with the heading “Why did I get this response?” Notably, when the user hasn’t started a conversation or provided any input, this grey box remains empty. However, its purpose becomes clear once the user initiates a conversation with the system, as it begins to provide explanations for the system’s responses.

We are providing a step-by-step explanation adhering to our design goals of the process that occurs within our dialog pipeline, even in cases where there may be a disruption in the dialogue. It is important to note that while this explanation lacks visual elements, it utilizes a color-coded system to organize the pipeline stages in a visual manner, in line with our design goal, DG1. This approach not only guides the explanation through the pipeline but also highlights the user’s intent and identified entities (slots), as per DG1. In instances where certain slots are absent, the explanation provides insights into why the instruction failed and specifies which slot is missing from the user’s utterance, consistent with DG2.

Figure 4.7 also illustrates a successful scenario within our system. In this context, the explanation accompanying the scenario demonstrates that when a user’s instruction falls within the system’s domain, contains a clear intent, and doesn’t lack any slots, the system successfully processes the instruction. Consequently, the robotic

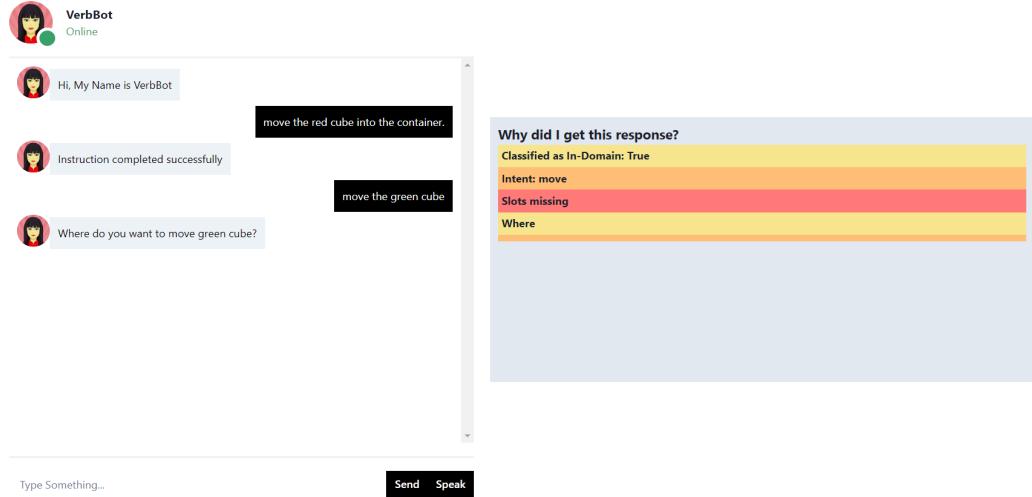


Figure 4.8: Explanation when slots missing

arm receives and successfully executes the task as intended. Figure 4.8 shows the explanation when the slots are not filled completely.

4.2.3 Expert vs non-expert explainability

Authors in [Bertrand23] highlight the differences between expert and non-expert explainability in the XAI and present the previous work done in this regard. Where expert-level explainability is designed for individuals with in-depth knowledge of the underlying system. It provides detailed and technical insights into how systems make decisions, including information about model architecture, feature importance, and mathematical details. It is essential for model development, debugging, and fine-tuning. Whereas non-expert explainability in XAI is tailored for users who lack technical expertise in the system. It simplifies explanations using user-friendly visuals, natural language, or intuitive graphics to help users understand the system's decisions.

While all the explanations offered for our dialogue system are presented in a manner accessible to non-experts, i.e. users with slight experience of using chat systems would be able to use it. In certain scenarios, the system's response may not align with the user's intent, despite the correctness of the user's input from their perspective. Such discrepancies often arise from variations in sentence structure. In such instances, individuals with technical expertise may find it helpful to consult the dependency tree(3.3.2) of the instruction to diagnose the source of the issue. This can be achieved by clicking on the user utterance, represented by black bubbles in the chat user interface, which will trigger the display of the dependency tree within the user interface for reference.

As previously discussed in the reference [You23], the incorporation of explainability into response generation serves to augment transparency, accountability, and trustworthiness within the dialog system. To quantify the extent to which these benefits are realized, we will conduct an evaluation in the subsequent section, wherein we thoroughly assess the performance and impact of our dialog system.

CHAPTER 5

Evaluation

In this chapter, we thoroughly evaluate the dialogue system we conceptualized in previous chapters with our implemented prototype. A user study was conducted to assess its performance and usability, discussed in the section 5.1. Subsequently, the results of the user study are presented in the subsequent section 5.2. Section 5.3 discusses the confounding factors and presents the advantages and disadvantages of the dialogue system based on user feedback. Finally, a summary of the evaluation is provided in the last section.

We conducted an assessment to evaluate both the usability of the system and the impact of explainability on user interactions. ISO 9241-11([23a]) provides guidelines for evaluating usability, and we considered the following aspects:

- *Effectiveness*: This dimension measures the success rate of user instructions in accomplishing their intended tasks. The evaluation of effectiveness is contingent on the performance of internal system components, such as domain and intent classification, and slot-filling. We will delve into the performance of these components in the results section.
- *Efficiency*: We assessed efficiency using the Scale to Measure Subjective Experienced Effort (SSEE), developed by [Zijlstra85]. This scale gauges the mental effort exerted by users to complete their assigned tasks.
- *User Satisfaction*: To gauge user satisfaction with the system, we employed the System Usability Scale (SUS) ([Brooke96]). Additionally, we conducted a qualitative evaluation to assess the naturalness of the system's interactions.

By examining these dimensions, we aimed to comprehensively evaluate the usability of the system and explore how explainability influences user interactions.

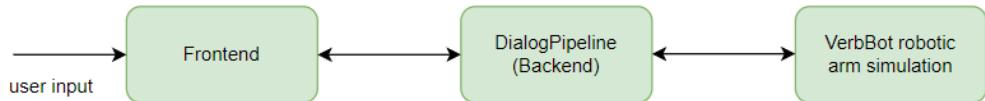


Figure 5.3: Overall System

5.1 User Study

A total of 14 participants took part in the user study, representing diverse fields and varying levels of prior experience with dialogue systems. The study was conducted within the Department of Informatics, and the majority of participants came from the field of informatics. The inclusion of participants from diverse fields aimed to capture perspectives from individuals outside the informatics domain.

5.1.1 Overall System

Figure 5.1 displays the distribution of study participants. We also inquired whether participants had prior experience using ChatGPT to assess if this prior experience influenced their perception of the dialogue system. There were only 3 non-ChatGPT participants and they also belonged to the non-Informatik group.

Figure 5.3 illustrates the user's interaction with the system through the frontend(chapter 4). User utterances undergo processing within the dialogue pipeline, and upon successful processing, the instructions are transmitted to the robot arm simulation program, as elaborated in Chapter 3.

The participants in the study were assigned a set of tasks, each of which consisted of an initial state and a desired final state. The primary goal for participants was to navigate from the initial state to the final state while engaging in social conversation. Further details regarding the specific tasks will be elaborated upon in the following subsection.

Figure 5.2 illustrates the user interface of the simulated robotic arm. In addition to displaying the robotic arm itself, it also portrays the designated task area, which

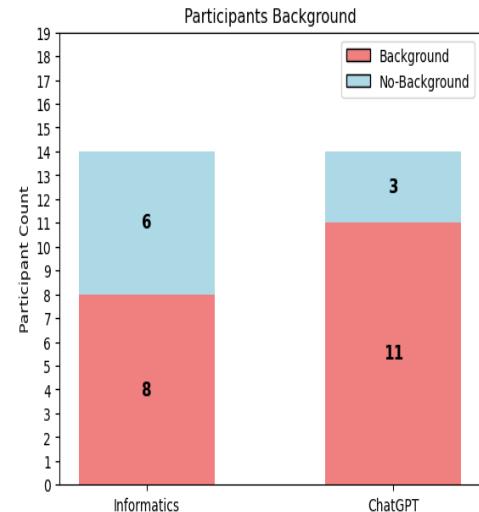


Figure 5.1: Participants distribution

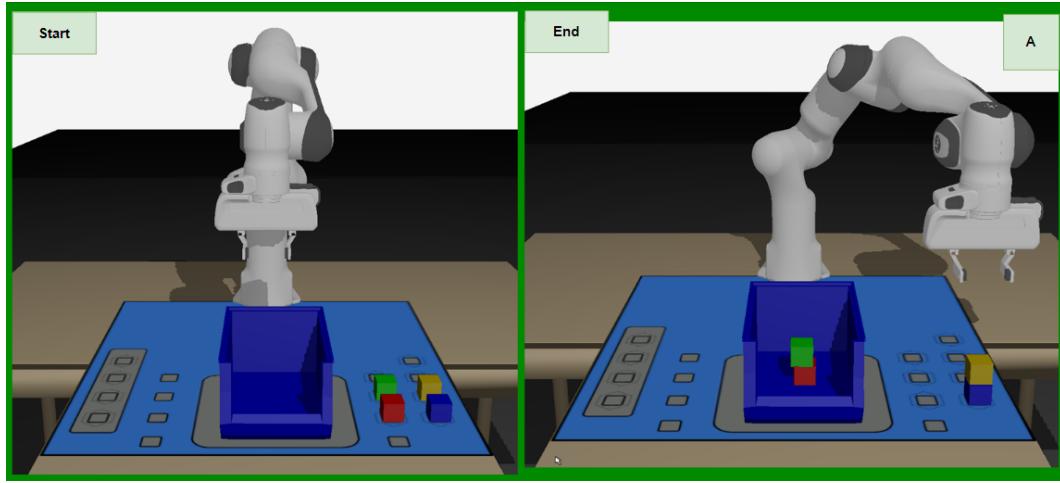


Figure 5.4: Task A User study: start state (left) and target state (right) to be reached by verbal instruction

serves as the location for task-related objects. It is important to note that all movements of the robotic arm occur exclusively within this task area. During the course of the simulation program, participants can observe the robotic arm's actions when specific programmatic actions are executed.

5.1.2 Procedure of Study

Before describing the procedure for the user study, it is important to explain the nature of the tasks that were assigned to the participants. The tasks were designed based on the supported intents, resulting in a total of three tasks:

- A Simple task related to the “move” intent.
- B Task involving the “move” intent with repetitive actions.
- C Task encompassing the “pour” and “shake” intents.

This subsection provides a detailed elaboration of the first task (task A, figure 5.4), while additional information about the other tasks can be found in the appendix.



Figure 5.2: Simulated robotic arm

In Figure 5.4, we present the initial state and the desired target state of the first task. In this task, participants are required to provide written instructions to guide a system from the starting state to the target state. Notably, our system possesses the capability to engage in social discourse. As a result, participants have the flexibility to seamlessly transition between casual social conversations and task-related discussions while accomplishing the assigned task. The figure shows the task area consisting of containers and cubes in the task area. In the desired end-state, one can observe the presence of red and green cubes situated within the container, while a yellow cube is positioned atop the blue cube. The user's objective is to attain this configuration by providing a set of instructions.

Now that we have described the tasks and elaborated on the first task, we can proceed to describe the steps in the user study procedure.

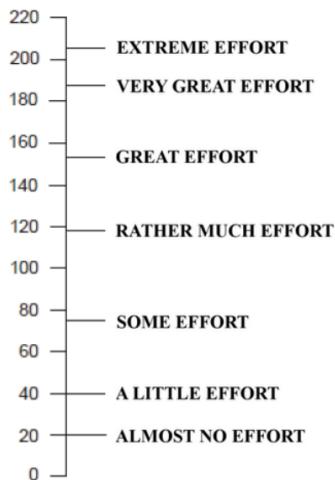


Figure 5.5: SSEE mental effort scale [Zijlstra85].

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1. I think that I would like to use this system frequently.	<input type="checkbox"/>				
2. I found this system unnecessarily complex.	<input type="checkbox"/>				
3. I thought this system was easy to use.	<input type="checkbox"/>				
4. I think that I would need assistance to be able to use this system.	<input type="checkbox"/>				
5. I found the various functions in this system were well integrated.	<input type="checkbox"/>				
6. I thought there was too much inconsistency in this system.	<input type="checkbox"/>				
7. I would imagine that most people would learn to use this system very quickly.	<input type="checkbox"/>				
8. I found this system very cumbersome/awkward to use.	<input type="checkbox"/>				
9. I felt very confident using this system.	<input type="checkbox"/>				
10. I needed to learn a lot of things before I could get going with this system.	<input type="checkbox"/>				

Figure 5.6: SUS questionnaire [Brooke96].

- Participants filled out a consent form regarding the collection of user data. The participants were divided into two groups, one group was shown explainability UI(see chapter 4) for the responses while the other group received no explanation for the responses.
- Participants were informed about the tasks they were required to perform, were provided with information about the dialogue system, and were told the research scope(3.5) to facilitate smooth conversations with the system.
- Participants were informed that they could engage in social talk at any point during the study. They were not coerced into social conversations to maintain the naturalness of interactions.

4. Participants were given the three aforementioned tasks, and at the end of each task, they were asked to rate their mental effort using the SSEE([Zijlstra85], Figure 5.5) scale.
5. At the end of the study, participants were asked to complete SUS ([Brooke96], Figure 5.6) questionnaires and provide feedback on the naturalness of their interactions with the system.

5.2 Results

In this section, the results of the user study will be analyzed. First, an examination of effectiveness will be conducted, focusing on the performance of the internal components of the dialogue system, as this directly impacts system performance and user satisfaction. Subsequently, an analysis of user feedback will be carried out to assess efficiency and user satisfaction.

5.2.1 Domain Classification

In total, there were 478 utterances, out of which 97 utterances were classified as out-of-domain. Figure 5.7 shows the confusion matrix for the domain classification, only one out-of-domain utterance was classified as in-domain, and 46 in-domain utterances were classified as out-of-domain. The confusion matrix displays the correct classification label (based on the user expectations and research scope) on the vertical axis. The horizontal axis indicates the classifier's actual outcome. Although the precision in classification is high but recall is low due to incorrect non-task-oriented(non-TODS) classification.

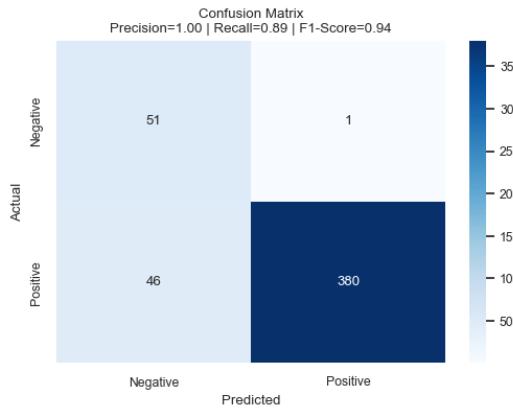


Figure 5.7: Confusion matrix for domain classification

A comprehensive examination was performed on statements that had initially received an “out-of-domain(ood)” classification but were, in fact, meant to fall within the “in-

domain" category. These misclassifications occurred predominantly due to the absence of specific terms in the vocabulary. Subsequent analysis focused on elucidating the characteristics of these user utterances. There were 9 more utterances although they were correctly classified as *ood* due to *typos*, these utterances were mostly instructions to complete the tasks. Therefore out of 51 instructions that were correctly classified as *ood*, 9 were instructions with typos so there were only 42 utterances that were actually meant to be social interaction.

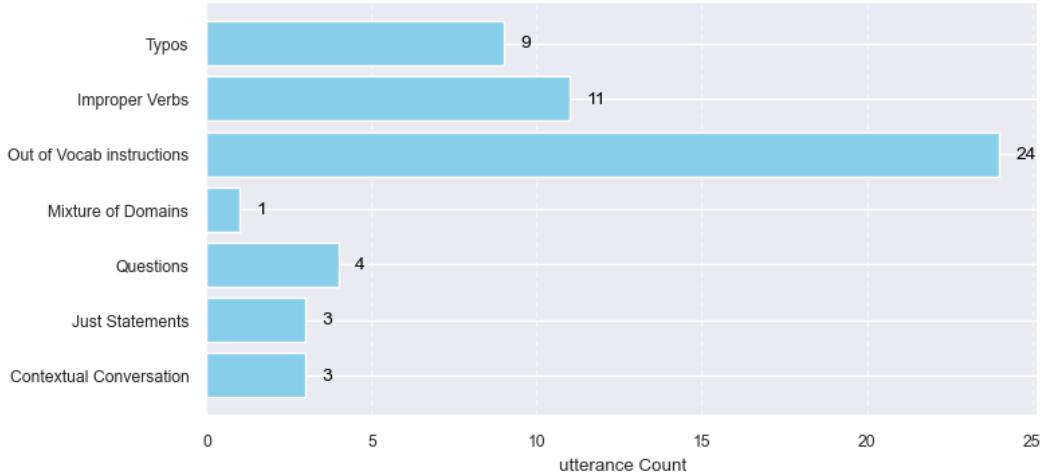


Figure 5.8: utterances count that were intended as in-domain

Figure 5.8 illustrates the count of the utterances and their categories that were not intended as "in-domain" but classified as *ood*. Here we will describe all these types of utterances in detail.

- *Typos*: This category pertains to utterances containing spelling errors in the words used, for example "move readcube into the Blue Box".
- *Improper Verbs*: This category encompasses sentences in which verbs employed are not found within the domain vocabulary, for example "Stack the green cube on top of the red". Here verb "stack" is not present in domain vocabulary.
- *Out of Vocabulary instructions*: In this category, instruction features words that are not included in the domain vocabulary, for example "put rest red cube into container" (where "rest" is not present in the vocabulary).
- *Mixture of Domains*: Sentences categorized here blend task-related information with non-task-related conversational elements, for example "Hi Robot, could you please touch the yellow cube?".
- *Questions*: This category encompasses instances where users pose queries related to the task at hand, for example "Which objects exist in the scene?".

- *Just Statements*: Users within this category make statements that lack direct relevance to the task, for example “I will give some instructions to a robot simulation soon”.
- *Contextual Conversation*: Users in this category attempt to reiterate or restart the same instructions for example “Try again”.

Figure 5.8 shows 55 instructions in total that were intended as in-domain from users but classified as *ood*, 9 out of 55 were correctly classified as *ood* due to typos, although some participants expressed that it should have also been classified as in-domain since many chat systems do autocorrect but since it was not in the scope of the thesis, they were correctly classified as *ood*. The other 46 should have been classified in-domain but due to inadequate domain vocabulary, it failed to do so. Based on the correct classifications from the confusion matrix (5.7) the ratio of true positives to total utterances comes out to be 0.9017 therefore accuracy is 90.17%.

In summary, the classifier exhibited strong performance in classification, although there is still potential for improvement to minimize erroneous classifications. Given the dynamic nature of scenes and tasks, it’s not feasible to include all possible synonyms within the domain vocabulary. A more effective approach, as suggested by participants, could involve leveraging the Language Model (LLM) for domain classification. This would entail providing the LLM with scene information by converting it into text and exploring the feasibility of this approach is a topic to be explored in the thesis outlook (6.3).

5.2.2 Intent Classification and Slot-filling

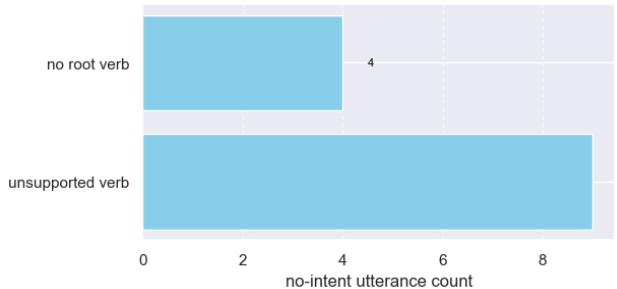


Figure 5.9: No-Intent Utterance Types

Figure 5.10 presents the confusion matrix for intent classification, with a total of four classes (as the simulated arm could only support three actions and a fourth class for the “no-intent”). Out of the 381 utterances that reached this stage, 31 of them were classified as having no intent, even though in reality, there were only 13 such utterances. It’s worth noting that all unsupported verbs/intents should also be categorized as having no intent.

5.2 Results

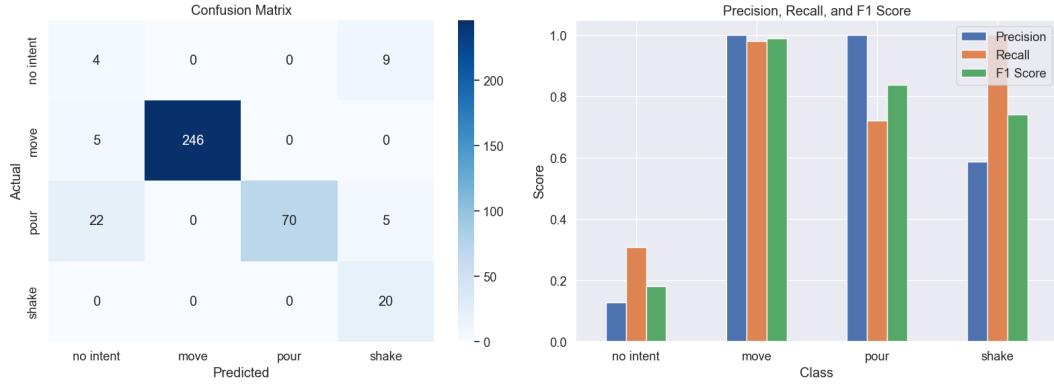


Figure 5.10: intent classification: confusion matrix

Among the 13 non-intent utterances (Figure 5.9), users did not employ any root verb in 4 of them. For the remaining nine, users attempted to execute unsupported intents like “remove”, but all were inaccurately categorized as “shake”. Under our current intent classification approach (Chapter 3.2), unsupported intents are getting classified as intents falling within the supported intents. How to address these unsupported intents will be a subject of future research outlined in this thesis (6.3).

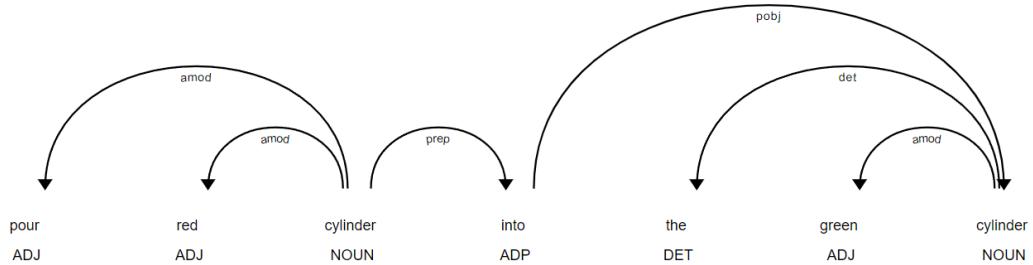


Figure 5.11: Non-detection of root verb

As Figure 5.10 shows 27 utterances that should have been classified as either “move” or “pour” intent were instead categorized as “no-intent”. This misclassification occurred due to the dependency tree of the instruction failing to identify the root verb. In Figure 5.11, one can observe that the supposed verb “pour” was labeled as ADJ (adjective) within the dependency tree. This labeling led to the absence of a recognized verb in the sentence, resulting in the classification of the utterance as ‘no-intent’. Investigating the identification of verbs in cases where they are missing from the dependency tree presents an intriguing research avenue and this exploration could potentially serve as an area for future study. Total correct classification accuracy measured from the confusion matrix (Figure 5.10) comes out to be 89.24%. But individually as can be seen from the respective graph for the F1 scores, it is the

maximum for the “move” intent but for other classes, it is really low, therefore the classification for other classes is not as good as for the “move” intent.

In slot-filling, a total of 14 utterances in which slots were missed. Among these, only one instance was due to user omission, while the remaining instances were attributed to the generation of distinct dependency trees that did not conform to the instruction structures(3.3.2) we supported in our implementation, for example, "Move five green cubes where the crate is empty", this leads to different dependency tree compared to what we proposed.

5.2.3 Task Success: Effectiveness

In the preceding subsections, we assessed the internal components and their accuracies, as these components significantly contribute to task success. Accurate domain and intent classification, for instance, is a crucial factor that can enhance task success. Here, we analyze the effectiveness.

Total Utterances	478
Classified as In-Domain	381
Utterances with Intent Found	350
Utterances where Slots Were Missing	14
Instructions Sent to VerbBot	327

Table 5.1: Recap of Utterances

Table 5.1 provides a summary of the previous subsections. Out of the 350 utterances that followed successful intent classification, 14 of them had missing slots, and 327 were sent as instructions to the VerbBot program. To accurately calculate task effectiveness, it is essential to reiterate the difference between an ‘utterance’ and an ‘instruction’(2.1). An ‘utterance’ encompasses anything from words to paragraphs expressed by the user to the dialogue system, while an ‘instruction’ refers to a specific utterance designed to complete a particular task.

In cases where slots are missing, the user is required to provide additional utterances to fill those slots. It’s important to note that these supplementary utterances used for slot filling are considered part of the same instruction.

Therefore, in our assessment of task effectiveness, it is crucial to determine both the total number of instructions issued and the subset that was successfully executed. Out of a comprehensive set of 391 instructions (as detailed in Table 5.2), the dialogue system recognized 327 instructions as suitable for transmission. It is worth mentioning that, although all 391 instructions were deemed correct from the user’s viewpoint, only 327 of them were actually dispatched to the simulation program. Importantly, these 327 instructions were sent for execution but were not executed in the simulation.

Category	Count
ood Instructions (out of vocab)	24
Root-Verb Present but Not Detected	27
Slot missed with unsupported dependency tree	13
Sent Instructions	327
Total Instructions	391

Table 5.2: Summary of Instructions

Table 5.2 provides an overview of all the instructions, distinguishing between those that were sent and those that were not. The first three entries in the table represent instructions that were not transmitted. The category labeled “Out of Vocab” (refer to 5.2.1) includes instructions classified as out-of-domain due to a lack of vocabulary within the domain classification. Instructions where the root verb was present but not detected (refer to subsection 5.2.2) were a result of limitations in the library utilized for generating dependency trees. Additionally, 13 instructions where the slots were missing did not support our dependency tree structure. The first two entries in the table should have been sent but were not due to system constraints and inadequacies. Conversely, the instructions in the third entry were correctly withheld, aligning with the thesis’s defined scope. This categorization sheds light on the nuances behind instruction transmission, illustrating instances where system limitations and predefined boundaries influenced the communication process.

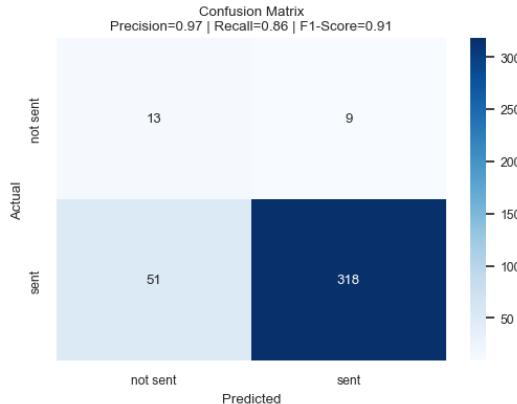


Figure 5.12: Instructions sent

Figure 5.12 displays the confusion matrix and highlights the count of instructions that should have been sent and should not have been sent, revealing that 9 instructions sent to the VerbBot program were inadvertently dispatched when they should not have been. These instructions should have been classified as “no-intent” but subsequently misclassified as having a “shake” intent (as illustrated in Figure 5.10), which contributed to this discrepancy.

To assess the effectiveness of the dialogue system in sending instructions, we calculate it by summing the true positives and true negatives and subsequently dividing this sum by the total number of instructions. This calculation yields an effectiveness rate of 84.65%.

Total Instructions	391
Instructions Sent	327
Instruction Success	179
Instruction Failed	147

Table 5.3: Summary of Instruction Outcomes

Until now, the focus has been on dispatched instructions. Now, the examination shifts to instructions executed by the simulation. Table 5.3 presents the breakdown of the outcomes of the instructions carried out by the simulation.

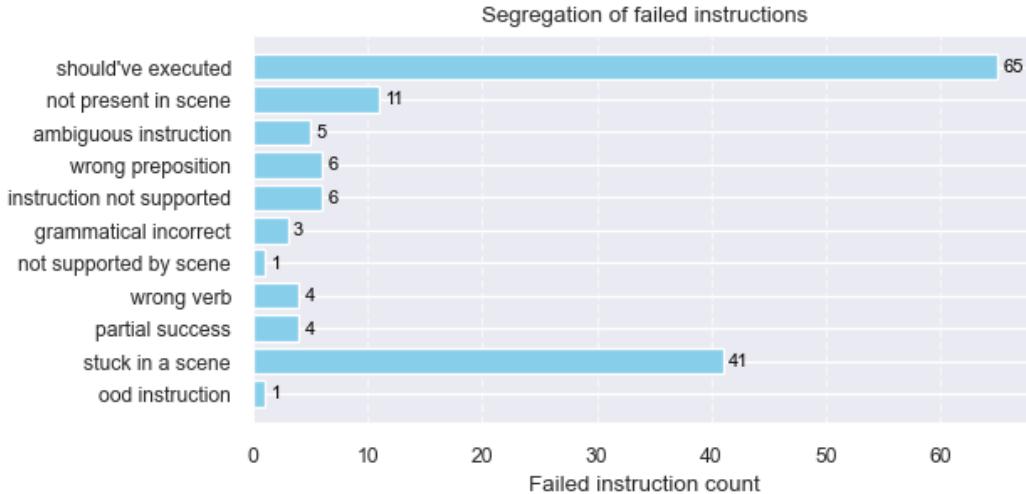


Figure 5.13: Segregation of failed instructions

Additional insight into this analysis is available in Figure 5.13, which provides a breakdown of the unsuccessful instructions, each of which will be subject to individual examination.

- *ood instruction*: In the overall assessment, there was a single user utterance that was erroneously categorized as in-domain despite being intended for out-of-domain processing. Remarkably, this utterance, "Can I see red color in the front?", successfully navigated the dialogue pipeline and was transmitted to the simulation program. The misclassification occurred due to several words in the utterance falling within the system's vocabulary, leading it to pass the domain classification. Furthermore, the utterance was identified as having the "shake" intent during intent classification, and the necessary slots required for executing

the "shake" action were found to be present. Consequently, the utterance proceeded through the pipeline and was sent to the simulation program for further processing.

- *Stuck in a scene*: Certain scenarios within the simulation posed challenges, leading to situations where the simulation became unresponsive and necessitated a complete scene reset. Regrettably, even valid and accurate instructions often failed to progress beyond this point, resulting in disruptions in task execution.
- *Partial success*: The instructions provided were occasionally not executed precisely as intended. For instance, in the instruction "Put the red cube in the crate in the third position", the red cube was relocated to the first position instead of the intended third position. While the red cube was indeed moved, it was placed inaccurately, deviating from the specified position in the instruction.
- *Wrong verb*: Incorrect verb was used to complete the task. For instance, in the instruction "put liquid of blue cylinder inside the green cylinder", the verb "put" was used instead of the more appropriate "pour" verb should have been used but "put" is used.
- *Not supported by the scene*: Certain tasks proved unfeasible due to the existing state of the scene. For instance, in the instruction "put green cubes inside the crate", the scene already contained red cubes within the crate. Consequently, fulfilling the task was impossible as the crate was already occupied, rendering the instruction unachievable within the given context.
- *Grammatically incorrect*: Task failures occurred due to grammatical inaccuracies in the given instructions. For instance, in the instruction "put all red cube in the container", the term "cube" should have been pluralized to "cubes".
- *Instruction not supported*: Intents that were wrongly classified as "shake" when the utterance had unsupported intents. For instance, in the instruction "switch the place of the red and green cubes".
- *Wrong preposition*: Mistakes occurred in the choice of prepositions for task completion. For example, in the instruction "put yellow cube above the blue cube", the preposition "above" should have been replaced with "on".
- *Ambiguous instructions*: Instructions with unclarity. For example "put yellow cube with other cube".
- *Not present in the scene*: Objects referred to in the utterance are not present in the scene.
- *Should have executed*: The instructions with no problems but could have failed due to limitations of the VerbBot program. For instance, the instruction "fill green cylinder with blue cylinder".

It is important to note that our dialogue system was both constructed and integrated with the pre-existing VerbBot program. Consequently, our evaluation focuses on the task success rate, specifically on the number of instructions that were successfully executed within the simulation environment. This assessment is conducted regardless of any challenges or issues encountered during the simulation environment. The emphasis lies on the completion of instructions in the simulated scenario, providing a comprehensive evaluation of the dialogue system's effectiveness in task execution. However, it was essential to analyze the failures, as doing so could provide valuable insights for enhancing the design of a more effective dialogue system and achieving a higher level of integration for future outlook.

The success rate is computed as follows:

$$\text{Success rate} = \frac{\text{Number of executed instructions}}{\text{Number of correct instructions}}$$

Correct instructions refer to those that should have been transmitted to the simulation. According to the data presented in Figure 5.12, there were a total of 369 correct instructions, out of which 179 were successfully executed, this translates to a success rate of 48.51%.

5.2.4 Mental Effort: Efficiency

In this section, the assessment of mental effort expended during each task is undertaken, employing the SSEE mental effort scale as introduced in Section 5.1[Zijlstra85]. The outcomes of mental effort for each task are presented in the form of box plots. The analysis commences by examining the overall mental effort and subsequently delves into an examination of mental effort across participant groups, as depicted in Figure 5.1.

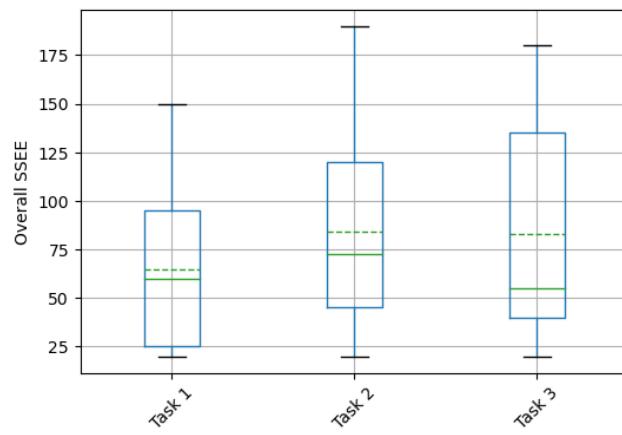


Figure 5.14: Overall SSEE score for all participants

Figure 5.14 illustrates the median of task difficulty scores falling within the range of 50 to 70 points and the mean lies between 60 to 80 points(dashed line), indicative of a spectrum ranging from “little effort” to “some effort”. However, it is noteworthy that mental effort ratings across users exhibited a degree of variability, with some individuals perceiving the tasks as relatively easy and others finding them considerably demanding. This variability is evident in the upper quartile of the distribution, where scores span from 100 to 180 points, suggesting a more pronounced mental workload for certain participants.

Several confounding factors were identified in the analysis. For instance, in the case of task 2, some users experienced heightened mental effort due to the simulation becoming stuck at specific points in the task. While this simulation was not formally considered as part of the evaluation, it exerted an influence on participants’ mental effort, thereby acting as a confounding variable. Similarly, for the third task, difficulties arose when the dependency tree generation process failed to identify the root verb in certain sentences. This issue was attributed to the utilization of the third-party `spacy` library for tree generation, thereby introducing another confounding element into the study. The confounding factors will be discussed in detail in the subsequent subsections.

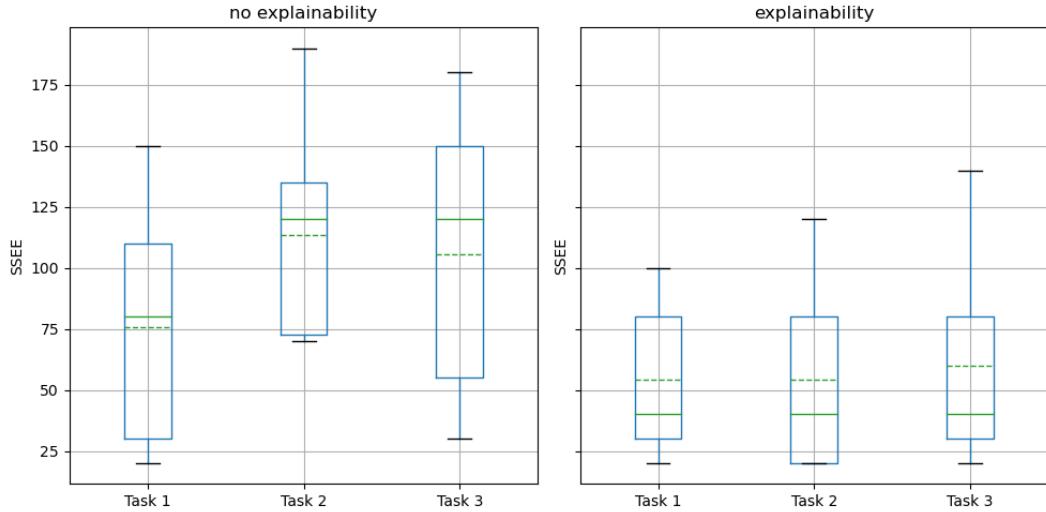


Figure 5.15: SSEE scores for groups with non-explanation and explanation

Figure 5.15 shows the differences in SSEE scores between the two groups one was given the explainability for the responses and the other was not. The group with explainability scores less in the SSEE with all mean, median lying between 40 to 60, whereas the group with non-explainability experiences a higher effort with mean and median ranging around 80-120 “rather much effort” in the SSEE scale. This actually shows that the group with explainability faced less effort in the completion of tasks.

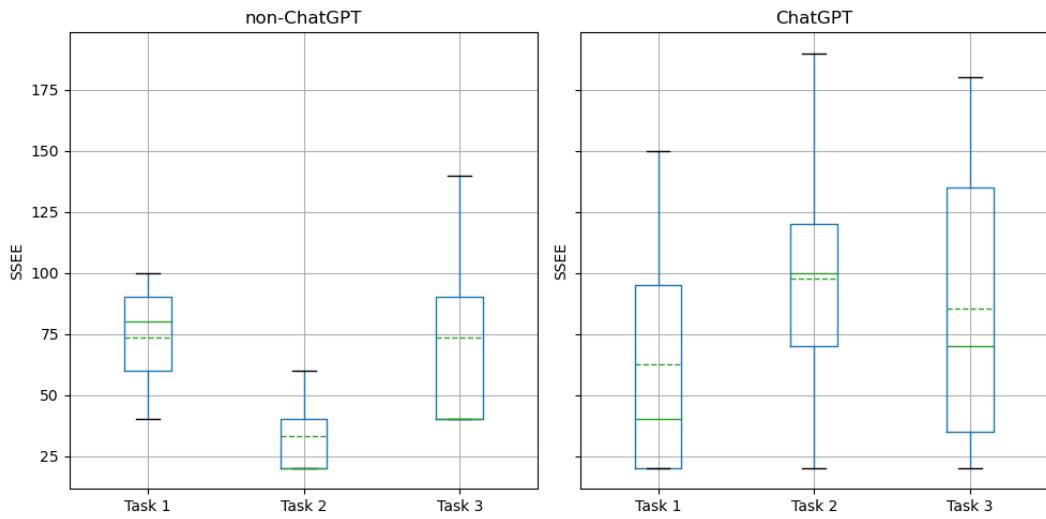


Figure 5.16: SSEE scores for groups with ChatGPT background

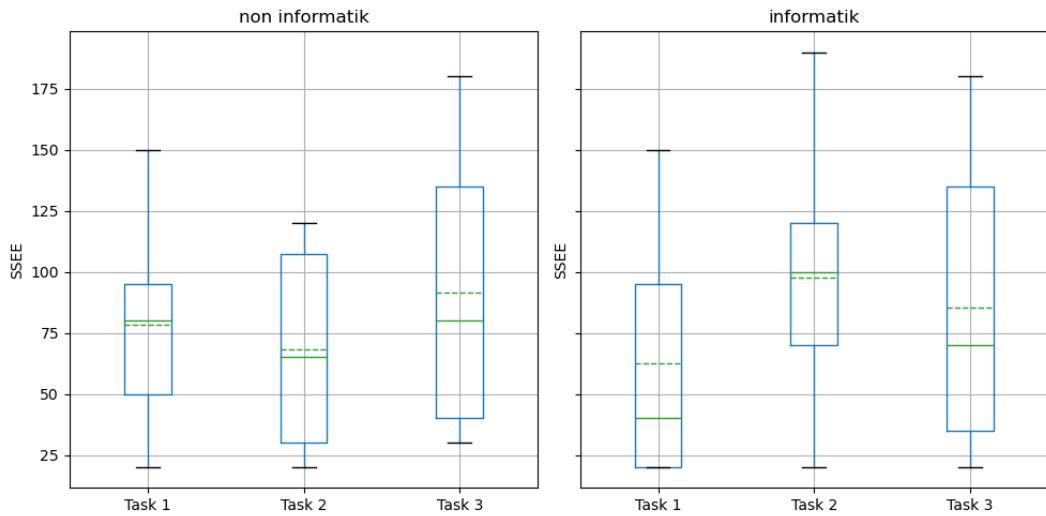


Figure 5.17: SSEE scores for groups with Informatik background

The objective of the evaluation for Informatik and ChatGPT users is to compare them in order to discern the distinctions in the efforts exerted by experts and non-experts. Participants with Informatik and ChatGPT backgrounds possess varying degrees of familiarity with how chat systems operate and how to engage with them so they can be deemed as experts. In our thesis introduction (Chapter 1.2), we envisioned that individuals lacking expertise in this domain would be able to utilize dialogue systems effortlessly. This evaluation serves as a means to assess whether non-experts encounter higher levels of effort in comparison to experts. It should also be mentioned

that all 3 non-ChatGPT participants were also part of the non-Informatik group and all non-ChatGPT participants were shown explanations.

Figure 5.16 illustrates the distinctions among groups characterized by their familiarity with ChatGPT. It's worth noting that there were only three participants who lacked any prior experience with ChatGPT. While the degree of effort exhibited by these groups varied depending on the specific tasks, it is noteworthy that non-ChatGPT users encountered minimal effort when performing the second task. For the remaining two tasks, the average effort exerted by both groups was nearly comparable. However, it's important to acknowledge that the smaller sample size of non-ChatGPT users limits our ability to draw conclusive insights from these findings.

In a similar vein, as depicted in Figure 5.17, there are discernible differences in the efforts expended across various tasks. Notably, the group lacking a background in Informatik exhibited higher mean and median effort values for the first and the third tasks, while for the second task, the non-Informatik group demonstrated lower effort levels. It is important to mention explainability was equally distributed among the groups with 4 out of 8 participants in the Informatik group were shown explanation and 3 out of 6 in the non-informatik group. In summary, these observations suggest that the Informatik group encountered slightly less effort in completing the task.

5.2.5 User Satisfaction

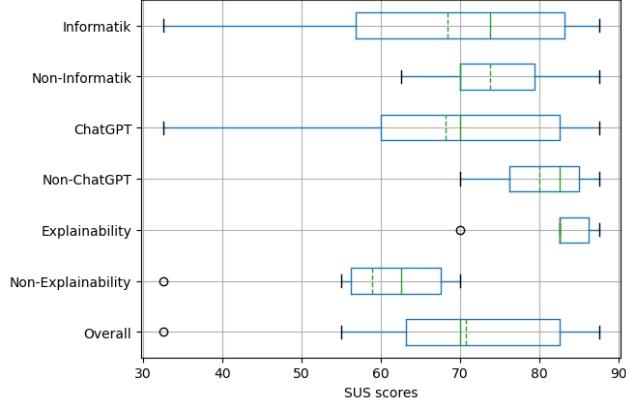


Figure 5.18: Overall SUS score and group-wise SUS scores

To gauge user satisfaction, we employed the SUS scale (5.6 [Brooke96]). We begin by discussing the overall SUS score, followed by the SUS scores specific to each user group. Additionally, we will delve into the qualitative feedback provided by users regarding the naturalness of the dialogue system.

Figure 5.18 displays the mean of all SUS scores, amounting to 71.37 (above 68 is considered above average [Brooke96]), indicating that the dialogue system is

considered acceptable for use. However, it is noteworthy that there is an outlier, with a participant providing a SUS score of 32.5. The average SUS scores exhibit a notable disparity between participants who received explanations (82.5) and those who did not (58.93), underscoring the positive impact of explainability on user satisfaction.

Furthermore, non-ChatGPT participants tend to report higher SUS scores in comparison to ChatGPT users (nonconclusive, as all non-ChatGPT participants were shown explanations), as is also observed in the case of Informatik and non-Informatik participants where average SUS score for the non-Informatik group was 73.75 whereas for Informatik group it was 68.18. This divergence could potentially be attributed to feedback from non-expert participants, who had more modest expectations of the dialogue system. It's worth noting that Informatik participants exhibited a slightly lower mental effort in task completion, but their SUS scores tended to be on the lower end. This is because the SSEE primarily assessed the efficiency of task completion, whereas the SUS score encompassed a broader assessment, taking into account overall satisfaction, including interactions outside the specific task domain.

Naturalness

For the qualitative evaluation of the dialogue system, participants were queried about their perception of whether the conversation felt robotic. Among the 14 participants, 5 indicated that the conversation did not feel robotic, while the remaining 9 participants expressed that they perceived the conversation as being with a robot-like entity. Two out of the 5 (who indicated to be not-robotic) participants did not have a background in ChatGPT and Informatik, but all 5 participants were shown explanations.

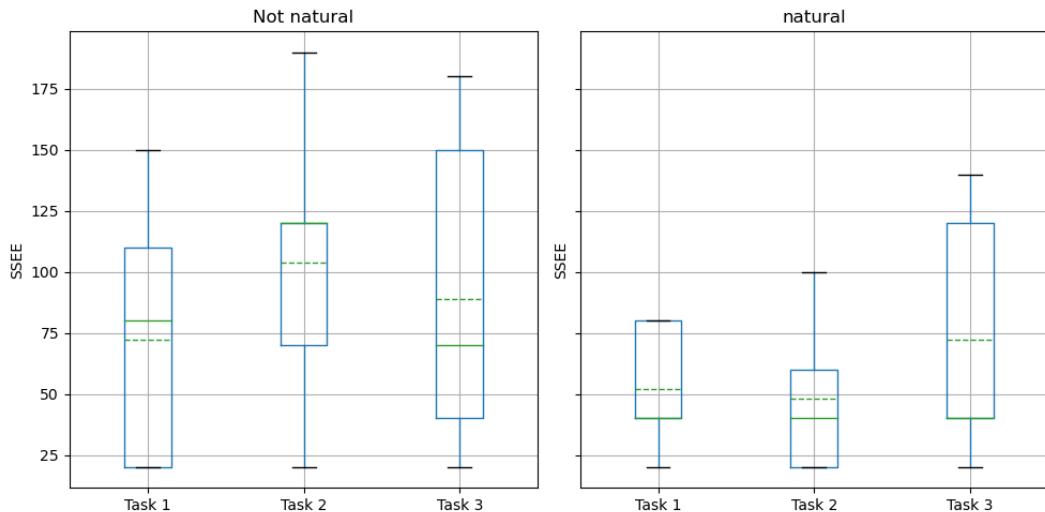


Figure 5.19: SSEE scores of natural vs unnatural response of participants

Figure 5.19 provides a clear depiction of the relationship between participants' perceptions of the dialogue system as either non-robotic or robotic and the mental effort expended during task completion. Notably, participants who regarded the system as non-robotic tended to require less mental effort in task execution. Moreover, participants who reported a natural feel to the interaction yielded an average SUS score of 82.5, while those who described it as unnatural averaged a score of 64.17.

Several participants had varying perceptions of the open-domain conversations within the dialogue system. Some participants described them as natural, while others found them amusing, citing occasional nonsensical responses. Interestingly, participants also provided specific feedback indicating that open-domain conversations felt natural to them, whereas closed-domain conversations were characterized by precision. Conversely, some participants held the opposite viewpoint, contending that closed-domain interactions felt more natural, aligning with typical instructional communication, while open-domain conversations were criticized for their inability to maintain context, thus feeling unnatural. The user feedback highlighted a collective desire to enhance the naturalness of the dialogue system. This topic will be thoroughly discussed in the next section.

5.3 Discussion

In this section, we will delve into the confounding factors and explainability impact. Towards the conclusion, we will outline the strengths and weaknesses of the proposed dialogue pipeline.

5.3.1 Confounding Factors

The assessment of user satisfaction indicates that the dialogue system is indeed usable. However, it is important to acknowledge the presence of various confounding factors that might have had an impact on the user satisfaction scores. In this subsection, we enumerate and discuss the factors that were beyond the control of the proposed dialogue system but could have influenced user satisfaction.

1. *System specifications:* To run a strong LLM model, the system specifications were not sufficient. For our dialogue system, we used "blenderbot-400M distilled" (3.18), but the best model available for Blenderbot is 9.4B, which means it is trained on 9.4 billion weights. Such a model is closer to ChatGPT, but running it requires extensive computational resources. With the 400M model, the quality of responses was not satisfactory, which impacted the participants' satisfaction.
2. *Google voice response:* The dialogue system possesses the capability to accept voice input and provide voice output. However, users were instructed not to use voice input due to potential issues with Google APIs' speech recognition,

which could impact user satisfaction. Nonetheless, participants continued to receive voice responses in addition to text for each interaction. Notably, one participant provided feedback indicating that their satisfaction might have been higher if the voice responses were softer in volume.

3. *Spacy dependency tree library*: The dependency tree generated by the Spacy ([Honnibal17]) library sometimes failed to identify verbs in sentences, even when they were present. This led to many utterances being misclassified as having no intent, which was frustrating for the participants.
4. *Instruction failures in the VerbBot program*: Many instructions became stuck and many should have been executed (5.2.3), but they failed due to internal issues in the simulation. These problems resulted in frustration among many participants.

5.3.2 Explainability

The explainability user interface (UI) was evenly distributed among both informatik and non-informatik groups. While it's observed that explainability correlated with reduced mental effort and enhanced user satisfaction, it's important to note that during the study, several participants did not actively engage with the explainability UI. Consequently, it's challenging to assert definitively that it was solely the explainability factor at play. It's plausible that the presence of explainability had some subconscious impact on participants, contributing to improved satisfaction, although this cannot be conclusively established.

Moreover, valuable feedback was collected regarding explainability. Some participants suggested that rather than presenting explanations in a separate UI, integrating them within the chat framework, alongside system responses, could be more engaging and prominent, making it less likely to be overlooked.

5.3.3 Advantages and Disadvantages

Table 5.4 presents both the strengths and weaknesses of the dialogue system. In this subsection, each of these points will be systematically reviewed and elaborated upon, including an examination of the rationale behind the weaknesses and potential solutions. These strengths and weaknesses are determined based on the research scope(3.5) of the thesis.

The dialogue system includes support for social conversations, a feature aimed at enhancing its naturalness. However, it's noteworthy that 9 out of 14 participants still perceived the system as somewhat "robotic". Some of the reasons for this perception can be attributed to confounding factors, particularly related to system specifications, as the utilization of a more potent LLM model wasn't feasible. Nonetheless, there are

Table 5.4: Advantages and disadvantages of proposed dialogue system

Advantages	Disadvantages
✓ Supports social conversation	✗ Still robotic
✓ Generic UI and easy to understand	✗ Needs good error messages
✓ Voice-based output	✗ Slow for social conversations
✓ Can detect missing information	✗ Doesn't possess scene information

several avenues to further improve the system's naturalness. Here are some potential enhancements taken directly from the participant's feedback:

- Research conducted by MIT ([Darling15]) has shown that providing names, backstories, avatars, and voices can make people feel more human-like when interacting with a robot. Incorporating such elements could significantly enhance the naturalness of conversations.
- Observations from the user study indicate that participants appreciate the ability to ask questions related to their tasks, facilitating easier task completion. Although our presented framework handled in-domain utterances without intent, the actual implementation of this component was not within the current scope. Integrating this feature can contribute to a more human-like interaction.
- Feedback from participants suggested that initiating social conversations should not solely rely on users. For instance, after completing an instruction, the system could proactively engage in social dialogue.
- Another valuable suggestion was to enable the system to process mixed instructions, combining task-related queries with social conversation. For instance, “Hi VerbBot, can you move the red cube, please?”.
- Participants also expressed a desire for the system to assist them in completing tasks, such as correcting typos and providing hints. These enhancements would contribute to a more natural conversational experience.

The participants found the UI to be highly intuitive, and they quickly grasped how to interact with it. The inclusion of voice responses was aimed at enhancing the naturalness of conversations, although further refinements could be made to select voices that align better with user preferences.

Additionally, the system demonstrated the capability to detect missing information and prompt users to provide it. The evaluation identified 14 instances where participants missed filling a slot, with 13 of these cases attributed to differences in dependency tree structures within the utterances. To enhance this feature, support for a broader range of dependency tree structures could be considered.

The other disadvantages, as highlighted in Table 5.4, and also from user's feedback, participants wanted better error messages. For example, when the root verb was

not detected, the system response could be “potential root verb is identified as an adjective”, or when the VerbBot simulation couldn’t complete the task, it should explain why it couldn’t. The solution would be the detection of possible failures and wrapping them in better error messages.

Participants also raised concerns about the slow social responses. However, it’s important to note that response time depends on system specifications. A more effective approach would involve running a superior Large Language Model (LLM) on a powerful machine or in the cloud, as our system can directly fetch responses from an independently operating LLM.

Furthermore, many utterances were classified as out-of-domain when participants attempted to ask questions related to the scene. This occurred because the domain classification lacked scene information and was limited to a specific vocabulary. To improve this, additional words could be incorporated into the vocabulary, or scene information could be provided to the domain classifier to enable better assessment of such utterances.

5.4 Chapter Summary

In this chapter, the evaluation criteria for the proposed dialogue system were introduced. The evaluation process, guided by ISO 9241-11 ([23a]) guidelines, encompassed the assessment of “effectiveness”, “efficiency”, and “user satisfaction”.

The evaluation process began by outlining the procedures of the user study (section 5.1) and detailing how it was conducted. Participants were selected based on their backgrounds in Informatik and ChatGPT to assess system usability, making distinctions between experts and non-experts. Participants were provided with information about the dialogue system and tasked with completing three tasks based on the intents supported by the system. Participants were informed that they could engage in casual conversation at any point during the study. Following each task, participants rated their mental effort using the SSEE scale. Towards the conclusion of the study, participants were asked to complete SUS questionnaires and provide feedback regarding their experience with the system.

In the following section (5.2), the results of the user study are examined. The evaluation begins with the internal components of the dialogue system, specifically focusing on domain classification, intent classification, and slot-filling, as these components directly influence task effectiveness. The evaluation revealed that the system’s ability to correctly convey instructions to VerbBot and avoid sending incorrect ones achieved an effectiveness of 84.65%. However, the success rate of task execution at the simulation was determined to be 48.51%.

To assess efficiency, the SSEE mental effort scale was employed, revealing that the overall effort expended ranged from “little effort” to “some effort”. Efficiency

comparisons were made across various groups, including those with and without explanations, participants with Informatik and non-Informatik backgrounds, and those with and without prior ChatGPT experience. The findings indicated that groups with explanations required less mental effort to complete the tasks. However, no definitive conclusions could be drawn for ChatGPT groups due to the limited sample size, with only 3 participants in the non-ChatGPT group. Additionally, the SSEE scores for the Informatik groups were found to be slightly lower than those for the non-Informatik group.

The user satisfaction assessment, as measured by the SUS scale, indicated that the dialogue system is indeed usable, with a score of 71.35. Upon further examination of SUS scores across various groups, it became evident that the explainability group consistently achieved higher scores compared to the non-explainability group. Additionally, in the comparison between the Informatik and ChatGPT groups, the non-Informatik and non-ChatGPT groups achieved higher scores compared to their counterparts. This difference may be attributed to the fact that the non-Informatik group had more modest expectations from the system. Additionally, the naturalness of the dialogue system was explored by analyzing the qualitative feedback provided by the participants.

Lastly, in the discussion section(5.3), all the confounding factors that could have potentially influenced user satisfaction were presented. This section also included an exploration of how participants actually perceived explainability during the user study. The section concludes by highlighting the advantages and disadvantages based on the user study, and it also proposes potential solutions to address the weaknesses based on the feedback provided by the participants.

CHAPTER 6

Conclusion

In this chapter, the work and its results are recapitulated. For this purpose, the created concept and its evaluation are summarized in subchapter 6.1. Subchapter 6.2 discusses the results obtained and describes the extent to which they provide a solution to the problems defined in subchapter 1.3. Finally, subchapter 6.3 offers an outlook for future extensions.

6.1 Summary

Conversational systems have become ubiquitous across various domains, ranging from virtual assistants to non-player game characters. Over the past decade, there has been a significant surge in enhancing conversational systems by making them more social. Granting these systems the ability to engage in free interactions with users not only renders conversations more natural but also boost user satisfaction. Consequently, many task-oriented dialogue systems(TODS) have transitioned from merely focusing on task completion-related conversations to incorporating a social dimension. In this thesis, our objective was to develop a task-oriented dialogue system tailored for tasks associated with a robotic arm and subsequently augment it with a social interaction component.

Chapter 1 provides an introduction to the thesis. Section 1.1 discusses the motivation behind imbuing the TODS systems with social capabilities, primarily because it enhances user experience, personalizes interactions, enables better handling of ambiguity, and fosters empathy, ultimately creating more lifelike and engaging human-machine interactions. Subchapter 1.2 presents the envisioned outcome where both experts and non-experts can engage in free-flowing conversations with the TODS system and provide instructions in a more human-like manner, aiming to make the interaction feel more natural. Furthermore, subchapter 1.3 outlines the specific tasks

and research objectives of the thesis, primary tasks of this thesis are two folds: firstly, correctly dispatching task-oriented commands to the robotic arm for execution and providing feedback in case of failures, and secondly, managing casual utterances separately to prevent unintended actions by the robotic arm. The research objectives include domain classification, intent and slot identification, response generation for missing slots, integrating Large Language Models (LLMs) for social conversation, and assessing the impact of response explainability on interaction with the dialogue system.

The chapter 2 presents the state of the art with the recent developments made in the NLP. The subchapter 2.3 highlights the existing work done in the domain of making a dialogue system social, on top of which we build our work. Subchapter 2.4 presents our research contribution to the community where we present an effective framework to handle in-domain utterances, we present a simple way to separate TODS utterances with social utterances and show the LLMs integration to handle the social utterances. We present our hybrid approach integrated with the existing approaches to figure out intent from the instruction and further, we create a slot structure by which necessary information required for the instruction completion can be represented efficiently in the form of slots.

Chapter 3 shows the constructed framework for the dialogue pipeline, and the subchapters delve into individual components in the pipeline, presenting multiple approaches and mentioning the reasoning behind the chosen approach. Section 3.1 presents the existing approaches for domain classification and lists the strengths and weaknesses of these approaches. It shows that the “rule-based classification” is best suited for our approach. The rule is based on the threshold T which is between 0 - 1, if $T \times 100\%$ of the words, after removing the stop words from the utterance, are present in the domain vocabulary, then the utterance is classified as in-domain. Subchapter 3.2 presents the existing approaches for intent classification, and since the existing approaches don’t classify the intent very accurately, a hybrid approach was proposed where our approach is used in combination with the existing approach to improve classification results. In hybrid first root verb from the utterance is checked within the supported intents(intent vocabulary), if it is present in the vocabulary then intent is found otherwise existing approach is used. Subchapter 3.3 highlights the importance of slots in completing the instruction. It also elucidates why the existing approach will not suffice in this context and presents the dependency tree structure of instruction and how slots are represented in the JSON format. A novel algorithm for slot identification is proposed. Further subsections of the section discuss conversation context within TODS and response generation. This includes presenting the response structure and explaining how a response is created based on the missing slots. An updated algorithm is also introduced, where slot filling and response generation with conversational context work in tandem. Section 3.4 shows the LLM models used for giving the dialogue system the ability to make social conversations. Three models and their multiple variants were explored along with

the OpenAI. The models were DialogGPT, GODEL, and, Blenderbot. The pros and cons of each model were highlighted, and "Blenderbot-400M distill" was chosen based on our system specifications and its reasonable responses.

Figure 6.1 depicts an end-to-end case of the instruction flow through the dialogue pipeline for a successful scenario. In this case, the utterance is an instruction classified as in-domain, with the correct intent, and containing the required slots to execute the instruction.

Chapter 4 emphasizes the design decisions made in selecting the chat UI and delves into the individual chat components. It also showcases the utilization of explainable AI, as explainability has been demonstrated to enhance user satisfaction. The chapter covers the UI for explainability and includes several examples for clarification.

The dialogue system was integrated with the VerbBot robotic arm simulation program for the evaluation (Chapter 5) of the dialogue system, which included a user study. Through this user study, the effectiveness, efficiency, and user satisfaction of the participants were assessed. The effectiveness of the dialogue system to send the instructions to simulation was measured to be 84.65%. Additionally, the success rate for task execution was calculated to be 48.51%. The evaluation employed the SSEE scale to gauge efficiency, revealing that the mental effort required to complete tasks ranged from "little effort" to "some effort". User satisfaction was measured using the SUS scale, which indicated that users were content with the dialogue system, as reflected by the average SUS score of 71.35 (see Section 5.2).

6.2 Discussion

The primary goal of this thesis was to develop a dialogue system tailored for the efficient handling of instructions related to robot arm operations, with an added emphasis on making the interactions more socially engaging. Several key objectives were outlined in Chapter 1.3 to guide the construction of the dialogue framework. These objectives were: *Domain Classification* utterances to TODS/non-TODS (O1), *Intent Classification* (O2), *Slot Identification* for missing information in instruction (O3), *Response Generation* to facilitate in filling missing slots (O4), *Explainability impact* to improve conversation (O5), and *LLM integration* to facilitate social interaction (O6),

Distinguish between open and closed domain utterances (O1), a task addressed by the creation of a domain classification module. In total, there were 478 utterances out of which 380 utterances were correctly classified as in-domain and 51 were correctly classified as out-of-domain. That gives the correct classification accuracy to be 90.17% (subsection 5.2.1). While the proposed approach exhibits satisfactory accuracy, certain participants in the user study expressed dissatisfaction when utterances were misclassified. Therefore, improving the accuracy remains a priority,

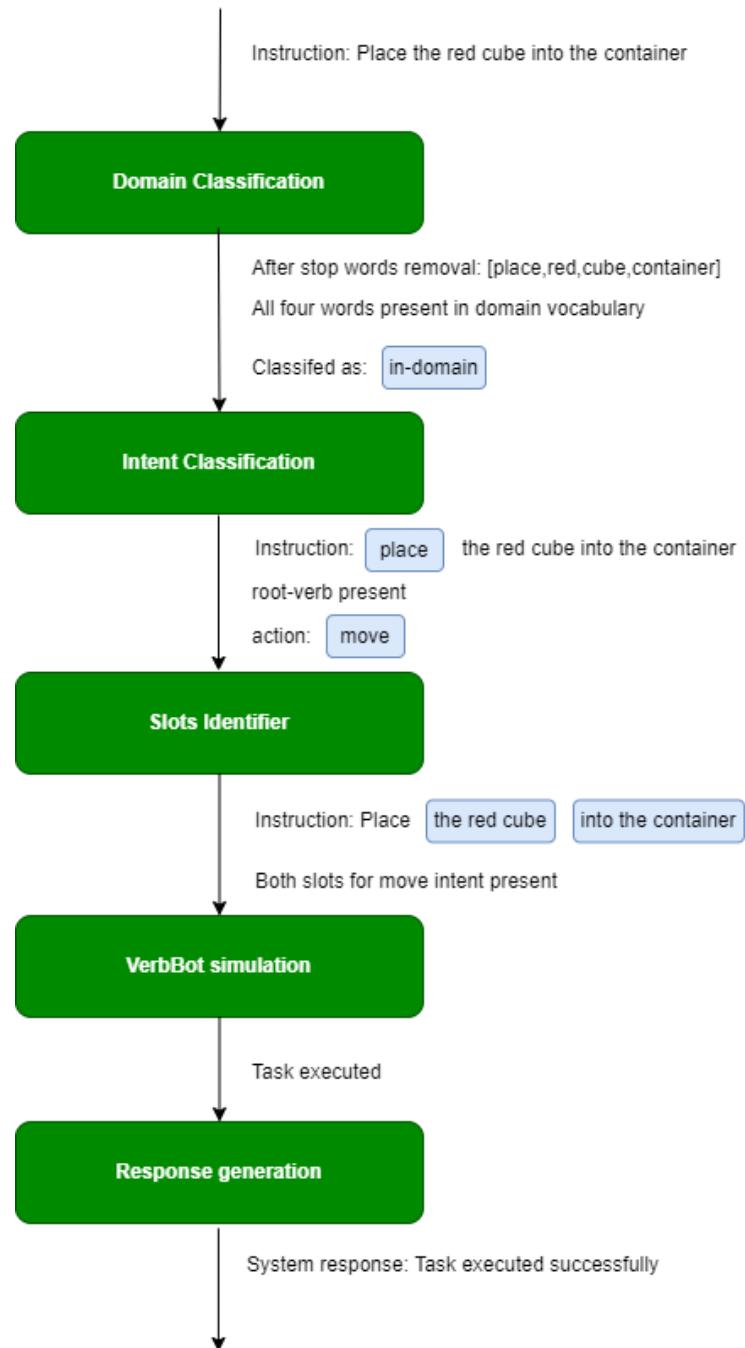


Figure 6.1: Flow of utterance through the dialogue pipeline: Success case

potentially through the exploration of alternative approaches, as suggested in the future extensions outlined in this thesis (6.3).

Identifying the underlying intent behind user instructions (O2), which was achieved through the development of an intent classification component. Out of the total 381 in-domain utterances, 340 were correctly classified giving the accuracy to be 89.24% (5.2.2). The classification accuracy for the “move” intent/class was notably high at 98%; however, accuracy for other classes was comparatively lower. There is a need to enhance this accuracy. One potential improvement could be the implementation of intent confirmation, as discussed in the future extensions outlined in this thesis (6.3).

Recognizing the missing information within user instructions and prompting users to provide the necessary details (O3 and O4). This task was accomplished using slot-filling and response-generation components. Out of 381 in-domain utterances, 14 utterances had slots missing, and only once the user had actually forgotten the slot other 13 were due to the instruction having different dependency tree structures that were not supported (5.2.2). This can be improved by adding support to the instructions with other dependency trees (6.3).

Assessing whether the inclusion of explainability features could enhance user satisfaction (O5). The results presented in Chapter 5.2 demonstrate that users who interacted with the explainability UI exhibited lower mental effort in completing tasks and a higher satisfaction score of 82.5 compared to users who interacted without explainability and scored 58.9 on the System Usability Scale (SUS).

Finally, integrating an appropriate LLM into the dialogue system to foster more engaging and socially adept conversations (O6). Multiple LLMs were tried and on the performance with our system specifications "Blenderbot-400M distill" was chosen (3.4). Although this model had the ability to hold some context it made it difficult to change the topic later and gave some non-sensical responses. It can be improved by using stronger LLM and offloading it to a cloud (see section 6.3).

It is noteworthy that when it comes to the naturalness of the dialogue system, 64.3% of the participants in the user study did not perceive the conversations as entirely natural, which gives the scope for a lot of improvement in the outlook. However, the results (5.2) that have been examined indicate a generally positive reception of the dialogue system by users.

6.3 Outlook

Based on the feedback provided by the participants during the user study, there is substantial room for enhancing the constructed dialogue system. Many of the solutions to improve the naturalness of conversations were discussed in the subsection presented in Chapter 5.3. Here, we will explore potential extensions to the dialogue system in this regard, the first four below listed possible future extensions are realized

from the user feedback and the latter were the system limitations realized during testing.

- Incorporating a *Question-Answering* Component: Although originally excluded from the scope, the study revealed that users have a preference for asking questions to accomplish their tasks. The integration of a question-answering component could facilitate users in task completion and contribute to more natural interactions.
- *Offloading LLM Processing*: Running the Large Language Model (LLM) on a cloud-based or separate machine, distinct from where the dialogue system is being run, has the potential to enhance response time and enables the use of more powerful models. This can significantly improve the overall quality of social responses.
- *Initiating Communication*: Granting the dialogue system the capability to initiate communication is essential, as users may not always initiate social conversations while completing tasks. Proactive initiation can contribute to a more engaging and natural interaction.
- *Scene Information Integration*: Providing scene information to the Domain classification module is crucial for accurate classification. Attempting to include all possible words into domain vocabulary may not be entirely feasible, so integrating scene information can enhance classification accuracy. Authors in [Sun23] present that LLMs can be used to classify the domain of a text based on some knowledge base, this can be used to do the domain classification based on the scene information which can enhance the accuracy of domain classification.
- *Intent Confidence Confirmation*: When the confidence level of the identified intent in a user's utterance is low, implementing a confirmation step by asking the user for clarification can address instances of misclassified utterances, contributing to a more accurate and natural dialogue.
- *Supporting Varied Dependency Trees*: Although the majority of instructions follow supported dependency trees, there are cases where instructions lead to unsupported dependency trees. Enhancing support for various dependency trees can improve the system's flexibility and natural language understanding.
- *Handling Multiple Intents*: The study revealed that in certain instances, users expressed a preference for employing instructions with multiple intents, such as "Pour red cylinder into green cylinder and shake it." Although none of the participants attempted more than two intents within a single instruction, it is evident that accommodating instructions with multi-intents could lead to enhanced user satisfaction.

the proposed enhancements provide a roadmap for expanding the dialogue system to facilitate more natural and effective user interactions. As informed by valuable feedback, these extensions not only promise a more natural conversation experience but also hold the potential to enhance task effectiveness and reduce the mental load required from users.

References

- [Barnes23] A. Barnes. *Pros and Cons of Using OpenAI in App Development*. Sept. 2023.
- [Bernazzani22] S. Bernazzani. *18 examples of Chatbots for Customer Service (amp; How You should be using them)*. Mar. 2022.
- [Bertrand23] A. Bertrand, T. Viard, R. Belloum, J. R. Eagan, and W. Maxwell. “On Selective, Mutable and Dialogic XAI: a Review of What Users Say about Different Types of Interactive Explanations”. In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 2023, pp. 1–21.
- [Bishop06] C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.
- [Brooke96] J. Brooke. “Sus: a “quick and dirty’usability””. In: *Usability evaluation in industry* 189.3 (1996), pp. 189–194.
- [Brown20] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].
- [Chen17] H. Chen, X. Liu, D. Yin, and J. Tang. “A Survey on Dialogue Systems: Recent Advances and New Frontiers”. In: *SIGKDD Explor. Newsl.* 19.2 (Nov. 2017), pp. 25–35.
- [Darling15] K. Darling. “‘Who’s Johnny?’Anthropomorphic framing in human-robot interaction, integration, and policy”. In: *Anthropomorphic framing in human-robot interaction, integration, and policy (March 23, 2015)*. *Robot Ethics* 2 (2015).
- [De Marneffe21] M.-C. De Marneffe, C. D. Manning, J. Nivre, and D. Zeman. “Universal dependencies”. In: *Computational linguistics* 47.2 (2021), pp. 255–308.
- [Dev23] J. Dev. *Title of the Article*. 2023. URL: <https://ordinarycoder.s.com/blog/article/react-chakra-ui>.

-
- [Džeroski04] S. Džeroski and B. Ženko. “Is combining classifiers with stacking better than selecting the best one?” In: *Machine learning* 54 (2004), pp. 255–273.
- [Fellows22] R. Fellows, H. Ihshaish, S. Battle, C. Haines, P. Mayhew, and J. Deza. “Task-Oriented Dialogue Systems: Performance vs Quality-Optima, A Review”. In: *Computer Science and Information Technology*. Academy and Industry Research Collaboration Center (AIRCC), July 2022.
- [Hassib17] M. Hassib, D. Buschek, P. W. Wozniak, and F. Alt. “HeartChat: Heart rate augmented mobile chat to support empathy and awareness”. In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 2017, pp. 2239–2251.
- [Honnibal17] M. Honnibal and I. Montani. “spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing”. To appear. 2017.
- [23a] ISO 9241-11. ISO, Sept. 2023.
- [Jurafsky] D. Jurafsky and J. H. Martin. *Speech and Language Processing*. <https://web.stanford.edu/~jurafsky/slp3/>.
- [Kamps22] H. J. Kamps. *Google makes robots smarter teaching them their limitations*. Aug. 2022.
- [Khurana21] A. Khurana, P. Alamzadeh, and P. K. Chilana. “ChatrEx: Designing explainable chatbot interfaces for enhancing usefulness, transparency, and trust”. In: *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE. 2021, pp. 1–11.
- [Khurana23] D. Khurana, A. Koli, K. Khatter, and S. Singh. “Natural language processing: State of the art, current trends and challenges”. In: *Multimedia tools and applications* 82.3 (2023), pp. 3713–3744.
- [Klüwer15] T. Klüwer. *Social talk capabilities for dialogue systems*. Saarländische Universitäts-und Landesbibliothek, 2015.
- [Lewis19] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. 2019. arXiv: 1910.13461 [cs.CL].
- [Liu19] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL].
- [Madumal18] P. Madumal, T. Miller, F. Vetere, and L. Sonenberg. “Towards a grounded dialog model for explainable artificial intelligence”. In: *arXiv preprint arXiv:1806.08055* (2018).

-
- [MCTC] MCTC. *ATIS Language Understanding Example*. <https://github.com/microsoft/CNTK/tree/master/Examples/LanguageUnderstanding/ATIS>.
- [Mikolov13] T. Mikolov, K. Chen, G. Corrado, and J. Dean. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).
- [23b] *OpenAI Playground* · openAI. <https://platform.openai.com/playground>. 2023.
- [Pan10] S.-Y. Pan. “Q,” A Survey on Transfer Learning, ”. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359.
- [Peng22] B. Peng, M. Galley, P. He, C. Brockett, L. Liden, E. Nouri, Z. Yu, B. Dolan, and J. Gao. *GODEL: Large-Scale Pre-Training for Goal-Directed Dialog*. 2022. arXiv: 2206.11309 [cs.CL].
- [Pennington14] J. Pennington, R. Socher, and C. D. Manning. “Glove: Global vectors for word representation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.
- [Qin19] L. Qin, W. Che, Y. Li, H. Wen, and T. Liu. “A stack-propagation framework with token-level intent detection for spoken language understanding”. In: *arXiv preprint arXiv:1909.02188* (2019).
- [Radford18] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. “Improving language understanding by generative pre-training”. In: (2018).
- [Radford19] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. “Language models are unsupervised multitask learners”. In: *OpenAI blog* 1.8 (2019), p. 9.
- [Rajpurkar16] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. “SQuAD: 100,000+ Questions for Machine Comprehension of Text”. In: *arXiv e-prints*, arXiv:1606.05250 (2016), arXiv:1606.05250. arXiv: 1606.05250.
- [23c] *roberta-base-squad2* · Hugging Face. <https://huggingface.co/deepset/roberta-base-squad2>. 2023.
- [Roller20] S. Roller, E. Dinan, N. Goyal, D. Ju, M. Williamson, Y. Liu, J. Xu, M. Ott, K. Shuster, E. M. Smith, Y.-L. Boureau, and J. Weston. *Recipes for building an open-domain chatbot*. 2020. arXiv: 2004.13637 [cs.CL].
- [Socher13] R. Socher, M. Ganjoo, C. D. Manning, and A. Ng. “Zero-shot learning through cross-modal transfer”. In: *Advances in neural information processing systems* 26 (2013).
- [Sonos] Sonos. *Sonos/NLU-benchmark*. <https://github.com/sonos/nlu-benchmark>.

-
- [Spangenberg17] M. Spangenberg. “Intuitive Roboterkommandierung basierend auf Verbalisierten Physikalischen Effekten”. PhD thesis. Universität Bayreuth, 2017.
- [Sun23] X. Sun, X. Li, J. Li, F. Wu, S. Guo, T. Zhang, and G. Wang. *Text Classification via Large Language Models*. 2023. arXiv: 2305.08377 [cs.CL].
- [Tiwari21] A. Tiwari, T. Saha, S. Saha, S. Sengupta, A. Maitra, R. Ramnani, and P. Bhattacharyya. “A dynamic goal adapted task oriented dialogue agent”. In: *PLOS ONE* 16.4 (Apr. 2021), pp. 1–32.
- [UXPin23] UXPin. *Chat User Interface Design – A Quick Introduction to CHAT UI*. Apr. 2023. URL: <https://www.uxpin.com/studio/blog/chat-user-interface-design/>.
- [Vaswani23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL].
- [Wang18] Y. Wang, Y. Shen, and H. Jin. *A Bi-model based RNN Semantic Frame Parsing Model for Intent Detection and Slot Filling*. 2018. arXiv: 1812.10235 [cs.CL].
- [Weizenbaum66] J. Weizenbaum. “ELIZA—a Computer Program for the Study of Natural Language Communication between Man and Machine”. In: *Commun. ACM* 9.1 (Jan. 1966), pp. 36–45.
- [Wezel21] M. M. van Wezel, E. A. Croes, and M. L. Antheunis. ““I’m Here for You”: Can Social Chatbots Truly Support Their Users? A Literature Review”. In: *Chatbot Research and Design: 4th International Workshop, CONVERSATIONS 2020, Virtual Event, November 23–24, 2020, Revised Selected Papers 4*. Springer. 2021, pp. 96–113.
- [Williams18] A. Williams, N. Nangia, and S. Bowman. “A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, 2018, pp. 1112–1122.
- [Wölfel20] K. Wölfel and D. Henrich. “Simulation-Based Validation of Robot Commands for Force-Based Robot Motions”. In: *KI 2020: Advances in Artificial Intelligence* (Sept. 2020).
- [Wu20] D. Wu, L. Ding, F. Lu, and J. Xie. “SlotRefine: A fast non-autoregressive model for joint intent detection and slot filling”. In: *arXiv preprint arXiv:2010.02693* (2020).
- [Yildiz21] E. Yildiz, S. Bensch, and F. Dignum. “Incorporating Social Practices in Dialogue Systems”. In: *International Workshop on Chatbot Research and Design*. Springer. 2021, pp. 108–123.

-
- [Yin19] W. Yin, J. Hay, and D. Roth. *Benchmarking Zero-shot Text Classification: Datasets, Evaluation and Entailment Approach*. 2019. arXiv: 1909.00161 [cs.CL].
- [You23] Y. You, C.-H. Tsai, Y. Li, F. Ma, C. Heron, and X. Gui. “Beyond Self-diagnosis: How a Chatbot-based Symptom Checker Should Respond”. In: *ACM Transactions on Computer-Human Interaction* (2023).
- [Yu17] Z. Yu, A. W. Black, and A. I. Rudnicky. “Learning conversational systems that interleave task and non-task content”. In: *arXiv preprint arXiv:1703.00099* (2017).
- [Zhang20] Y. Zhang, S. Sun, M. Galley, Y.-C. Chen, C. Brockett, X. Gao, J. Gao, J. Liu, and B. Dolan. *DialoGPT: Large-Scale Generative Pre-training for Conversational Response Generation*. 2020. arXiv: 1911.00536 [cs.CL].
- [Zijlstra85] F. Zijlstra and L. Van Doorn. *The construction of a scale to measure perceived effort*. University of Technology, 1985.

APPENDIX A

Additional documents

The chapter contains the additional tasks users had to perform in the user study since subchapter 5.1 only presented task A, tasks B and C are presented here.

Task B featured a repetitive assignment in which users were required to provide instructions for moving the red cubes inside the crates first and subsequently placing the green cubes on top of them. This task could be completed either by individually moving the red and green cubes one at a time or by grouping and moving all the red or green cubes together through a single set of instructions.

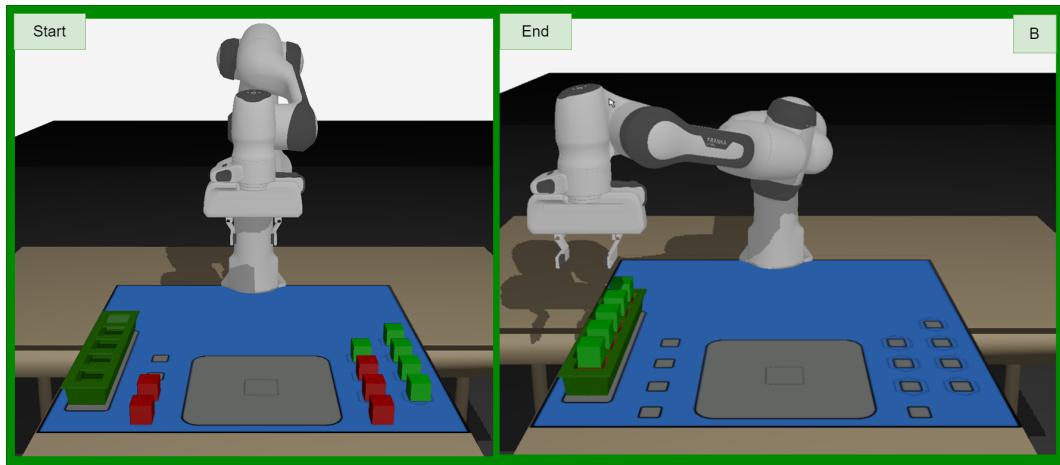


Figure A.1: Task B User study: start state (left) and target state (right) to be reached by an instruction

In the case of Task C, the initial state and the desired target state appeared identical. The provided diagram did not illustrate the process involved. However, the task entailed the following sequence: users were required to pour the red and blue cylinders

into the green cylinder and subsequently shake the green cylinder. Upon completion of the task, the change occurred within the contents of the cylinders rather than the physical positions of the objects within the scene. It is worth noting that while the scenes did not depict liquid within the cylinders due to simulation constraints, participants were informed of this aspect as they carried out the task.

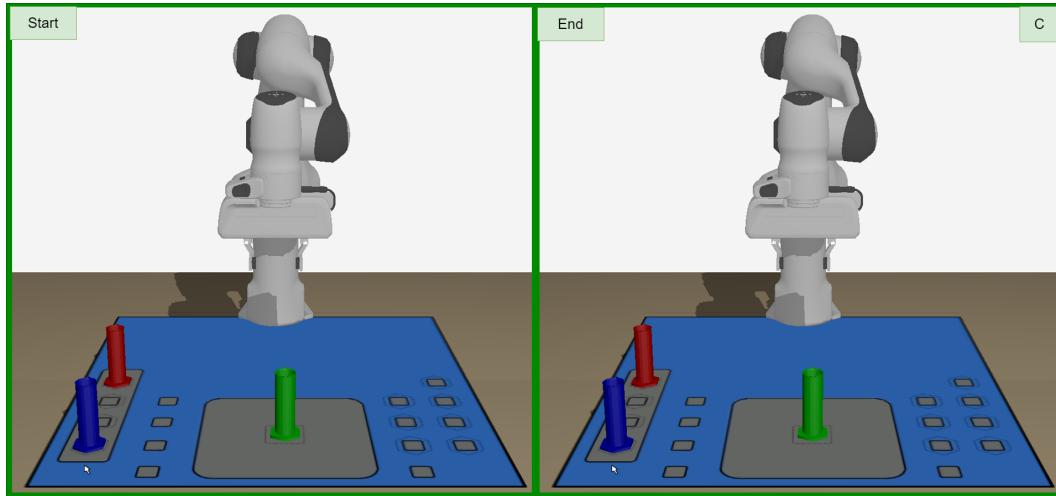


Figure A.2: Task C User study: start state (left) and target state (right) to be reached by an instruction

Declaration

I hereby affirm that I have authored this thesis independently and have not relied on any sources or aids beyond those explicitly acknowledged. All portions of this work that have been directly quoted or paraphrased from other sources are clearly and appropriately cited. Furthermore, this thesis has not been previously submitted to any examination authority in either the same or a similar form.

Bayreuth, October 2, 2023



Vivek Kumar Mahto