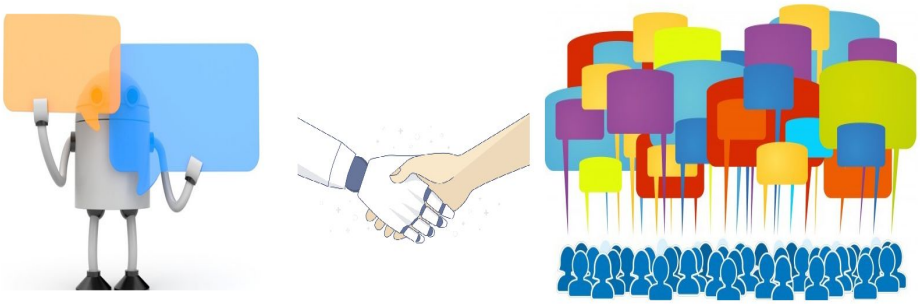
Helping Chatbots To Better Understand User Requests Efficiently Using Human Computation

Master's Thesis



Rucha Bapat

Helping Chatbots To Better Understand User Requests Efficiently Using Human Computation

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Rucha Bapat
born in Pune, India

Web Information Systems
Department of Software Technology
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
http://wis.ewi.tudelft.nl

Helping Chatbots To Better Understand User Requests Efficiently Using Human Computation

Author: Rucha Bapat
Student id: 4502183
Email: R.M.Bapat@student.tudelft.nl

Abstract

Chatbots are the text based conversational agents with which users interact in natural language. They are becoming more and more popular with the immense growth in messaging apps and tools to develop text based conversational agents. Despite of advances in Artificial Intelligence and Natural Language Processing, chatbots still struggle in accurately understanding user requests, thus providing wrong answers or no response. An effective solution to tackle this problem is involving humans capabilities in chatbot's operations for understanding user requests. There are many existing systems using humans in chatbots but they are not capable to scale up with the increasing number of users. To address this problem, we provide insights in how to design such chatbot system having humans in the loop and how to involve humans efficiently.

We perform an extensive literature survey about chatbots, and human computation applied for a chatbot, to guide the design of our reference chatbot system. Then we address the problem of cold starting chatbot systems. We propose a methodology to generate high quality training data, with which, chatbot's Natural Language Understanding (NLU) model can be trained, making a chatbot capable of handling user requests efficiently at run time. Finally we provide a methodology to estimate the reliability of black box NLU models based on the confidence threshold of their prediction functionality. We study and discuss effect of parameters such as training data set size, type of intents on automatic NLU model.

Keywords- Chatbot, Natural Language Understanding, Human Computation

Thesis Committee:

Chair:	Prof. dr. ir. Geert-Jan Houben, Faculty EEMCS, TUDelft
University supervisor:	Dr. ir. Alessandro Bozzon, Faculty EEMCS, TUDelft
Daily supervisor:	Dr. Pavel Kucherbaev, Postdoctoral Researcher EEMCS, TUDelft
Committee Member:	Dr. ir. Huijuan Wang, Faculty EEMCS, TUDelft

Preface

I would like to thank my supervisor **Dr.Alessandro Bozzon** for his guidance, support and for providing me with all the resources that were needed during my thesis work. His enthusiasm while discussing a problem statement motivated me to take up this project and successfully carry it to the end.

My sincere gratitude to my daily supervisor, **Dr.Pavel Kucherbaev** for helping me during each stage of this thesis project. I am thankful to him for providing me with his valuable suggestions and comments, without which completion of this work would not have been possible. His timely advice to make an outline, has not only helped me to go into detail concepts but also provided me a vision to look at a broader picture. I strongly appreciate his feedback during the writing of this thesis work. I am truly glad to have a supervisor like him.

I would also like to thank my other supervisors, **Dr.Erik Boertjes**, **Dr.Achilleas Psyllidis** and **Jie Yang** for their guidance during initial stages of my thesis. I would also like to thank all the thesis committee members for assessing my work and evaluating the report.

Moreover, I would like to thank my fiance **Akshay Oka** for encouraging me during down time in my thesis project. I am also grateful to my friends **Nilaya, Prajakta, Kshiteej, Sakshi and Arjun** for supporting and providing the valuable inputs. Furthermore, I would like to thank my class mates **Aditi** and **Sambit** for all the encouraging discussions during the thesis work.

Last, but not the least, I would like to thank my family members **Mandar Bapat**, **Madhura Bapat**, **Rugved Bapat** and **Suvidha Bapat** for supporting me through the thick and thin. Their trust has always motivated me to give my best performance.

Rucha Bapat
Delft, the Netherlands
August 21, 2017

Contents

List of Figures

Chapter 1

Introduction

Conversation in natural language enables people to exchange thoughts and information efficiently and quickly come to a shared understanding. Human-machine conversation aims to facilitate communication between users and computers via natural language. A conversational agent is a software program which interprets and responds to human user requests in natural language. Its goal is to take a user request, understand it and respond to the user with required information in a such a way that the user perceives the communication as similar to the one with another human. Conversational agents are widely used in broad range of applications in education, healthcare, customer support, marketing, and entertainment [44]. They can be used as a personal assistant like Siri, Cortana, Google Now; which tries to help users in performing different actions such as booking a flight or a movie ticket, reserving a table in a restaurant, scheduling a meeting, setting alarm etc. They have different mediums of communication such as text, speech, graphics, gestures and other modes on both the input and output channels.

Chatbots are a class of conversational agents that prevalently use text as a interaction medium [60]. Chatbots are not new. The first ever chatbot ELIZA [78] was created in the 1960s which was a program designed and developed to interact with it's users like a psychotherapist would. It used a script to recognize certain patterns and keywords and generate a response accordingly. The purpose of this chatbot was to prove how the communication between a man and a machine could work. ELIZA inspired a whole list of chatbots after it and resulted in the creation of chatbots such as ALICE [76], Albert One [12] and SmarterChild. ALICE and Albert One were based on pattern matching techniques same as ELIZA [1] while SmarterChild, developed in 2000 added an ability to process natural language by parsing a user input into meaningful words. Since '60, numerous chatbots were developed with different design techniques [1] and they are becoming more effective with the improved user interfaces, natural language processing and machine learning techniques. Chatbots have recently received lots of attention in the industry with the evolution of conversational platforms like messaging apps.

Despite the popularity and improved techniques, chatbots still face various challenges in understanding user's requests, processing them, generating an appropriate response and maintaining a conversation with users. One of the main challenges is natural language understanding. Users can ask requests in different ways, and a style of conversing also varies from person to person. One can ask'What is the weather today in Delft?' and another person can ask 'How is it in Delft today?'. These two requests have the same *intent* of asking weather information but they are differently formulated. In the first request, user has specified the word *weather*, while in the second user request, user expects the information about weather, without mentioning any weather related word. Sometimes this ambiguity of natural language makes a chatbot inefficient and it cannot correctly understand the context and meaning of the user requests. This makes efficiently handling user requests in a natural

language a consistent challenge [47].

Chatbots perform well if they are asked to provide conversational answers based on pre-defined input-output vocabulary pairs in a particular domain, where the matching rules are derived from heuristics. But they fail when users start asking questions beyond the boundary of this vocabulary and the chatbot cannot interpret what the user is actually asking. To fill this gap, researchers have tried to develop chatbot systems for different domains to form a single agent framework [82]. These systems are flexible enough to adapt the trained model from one domain to another [15, 75] and can handle cross-domain conversations. However there is still a problem to scale up the chatbot for generic conversations and not only limited to the specific domain.

Handling a conversation needs efficient dialogue management. For the chatbot-initiated dialogue [55], it is comparatively easy to handle the conversation. This technique has a straightforward sequence of statements to ask to user and chatbot also knows what to expect as an answer for that particular statement. This sequence of statements is designed based on pre-defined scenarios. Chatbot drives user through this sequence until it has all the information it needs to supply to the user with the relevant information. In this case, the chatbot is able to handle possible answers to users'questions as it understands everything correctly and knows how to handle the conversation. The complexity increases when the chatbot and the user, both can initiate the conversation. In this case, it becomes hard for a chatbot to understand the context of the user statement. It faces problems in understanding whether the coming statement is in a sequence with the previous statement or an answer to the chatbot's follow-up question or altogether a new sentence with some different topic. It becomes hard for a chatbot to keep the state of a user request and an answer. And developing a chatbot that can hold open conversations with a human becomes very challenging.

A chatbot requires a capability of solving multiple problems and various challenges simultaneously. These challenges can represent a separate research area into themselves. And despite extensive research in chatbots, they are still limited in *understanding users*, *consistently following a conversation* and *serving in a wider domain*.

1.1 Problem Statement

The very first step after receiving a user request in a chatbot is to understand the request correctly. The chatbot needs to understand the correct intent of a user request to respond to the user with required information. A further task of the chatbot of providing appropriate response to the user requests by performing some set of actions is also dependent upon correct understanding of a user request. If the chatbot fails to understand the meaning of the user's request, it won't be able to provide an appropriate response. It may lead to providing an irrelevant response or no response at all, leaving the user unsatisfied and unanswered.

Many different approaches are used to understand the user request correctly. One of the common methods is to write scenario-based pattern matching rules. An incoming user request is matched by already written rules to the available statement in a database or in a knowledge base. Then the answer associated with that particular statement is provided to the user. For example, an ALICE knowledge base uses Artificial Intelligence Mark-up Language (AIML) files of conversational patterns for a conversation with users [77]. The simple pattern matching is performed to understand the user request. Nowadays, machine learning approaches are increasingly integrated into a chatbot to improve natural language understanding of user requests. Chatbot systems are no longer dependent on rule-based pattern matching. Machine learning approaches use training data to train a natural language understanding model and each incoming user request is processed with this model.

It provides intent that represents a mapping between what a user says in his/her request and the action that the chatbot should perform. This mapping can already be specified in a training data. Machine learning approach also tries to extract the entities in user requests so as to map natural language phrases to pre-defined entities in a training data. Entities are the natural language phrasings present in a user request which collectively capture the meaning of a request.

To serve diverse user requests, rule-based pattern matching and machine learning need a large amount of rules and training data respectively. And even having lots of rules or large training dataset, understanding of user requests in a chatbot is still not perfect. On the other hand, it is very easy for a human to understand what another human user asks in a natural language. Individuals can easily recognize context-dependent terms or phrasing and provide the correct meaning of user requests. Humans can help chatbot systems by contributing in requests understanding tasks that they can easily solve but chatbots cannot solve. This can be done by means of *human computation*, an approach of asking humans to help in performing tasks where machines cannot yet be effective [71]. Human computation in a chatbot has been proved as an effective approach [42, 64, 9, 18, 41, 39] because it can mitigate the problems which fully automatic solutions struggle to solve. A chatbot system like Chorus [42] has been proved to be able to hold long and sophisticated conversations, understanding the complex statements in natural language. Another human-powered system Facebook M [22] uses professional employees for responding to user's queries with follow-up questions and updates as tasks move towards the completion. By leveraging human input, these systems are able to work well across a number of domains.

Human computation can empower a chatbot with human intelligence but it introduces some challenges such as cost, latency and scalability. Falling always back on human computation tasks would definitely provide the better understanding of a user request. But this method would keep involving humans even for the repetitive and already processed requests. A system like Chorus [42] relies only on human involvement for the each interaction with its users. This makes a chatbot system cost inefficient. The approach of involving humans in the chatbot would ultimately improve the robustness of understanding user requests but the problems still remain in efficiently leveraging such systems including cost and scalability. Chatbot systems like Facebook M are available only to a few hundred selected Bay Area residents and they are not scalable enough to the broader audience. There is a need to find out a solution of efficiently involving human computation in a chatbot for understanding user requests.

To find out the better solutions to these problems, we need to understand when and how the human computation should be involved in a chatbot. For this, our goal is to understand the appropriate entry point of human computation in a chatbot and find out an efficient way to involve it for understanding user requests when the automatic component of chatbot system fails to understand the user requests reliably. Based on this goal, we formulate the main research question of this thesis work as the following:

RQ: How to improve *understanding of user requests* in a chatbot *efficiently* using *human computation*?

We refer to a chatbot system having human computation involved in it, as a ***Hybrid Chatbot System***. To pursue an objective of our research question, First we need to learn the state-of-art of the chatbots, human computation and human computation applied for a chatbot. Then, we need to investigate how hybrid chatbot system could be designed, when we would need the help of humans for understanding user requests. Based on these findings, we want to understand how we could involve human computation efficiently at design time and at run time in a chatbot system. To achieve the main goal, we divide our work into 4 objectives as discussed below.

1.2 Research Objectives

In order to tackle the main research question described in Section 1.1, the question is divided into four sub-research questions. In this section, we introduce sub-research questions and their related objectives which are designed to answer these questions.

RQ1: What is the *state of the art* for applying *human computation* in chatbots?

Objective 1: To conduct a literature review about chatbots, human computation and applying human computation for chatbots.

Here we aim to carry out a literature survey about a chatbot methodologies, it's challenges, how to involve human computation in chatbots. We want to thoroughly understand the human computation methodologies, platforms and their use in chatbots. We also want to get the better understanding about the benefits of human computation in a chatbot and also find out the challenges in this approach.

RQ2: What is a possible *architecture* of a *chatbot having humans in the loop*?

Objective 2: To design and develop a hybrid chatbot system

We first aim to understand what could be the automatic and human computation components of such systems and how they look like. We understand how these components are related to each other and what are their design dimensions. We aim to design a hybrid chatbot system. Then we implement such hybrid chatbot system involving human computation for understanding user requests.

RQ3: How to collect *high quality training data* for a chatbot with the help of *human computation*?

Objective 3: To design a methodology to collect high quality training data for a chatbot using human computation

In this sub research question we focus on designing a methodology to collect high quality training data with which, the automatic natural language understanding model could be trained at design time of a chatbot. We aim to design a methodology to collect the training data, design the experiments and perform them to evaluate our methodology.

RQ4: How to *estimate the reliability* of *automatic understanding* of user request, having no knowledge of the used machine learning method?

Objective 4: To identify an approach to determine a confidence threshold level, below which the automatic understanding of a user request is considered as non-reliable

We not only aim to understand the context of user requests with the help of humans, but also to analyze how to involve them in an efficient manner. The goal is to understand when is the right moment to involve humans in a chatbot for understanding user requests. For this, we aim to first understand and analyze, when the automatic user request understanding is not reliable and we need to involve human computation for this, based on the confidence threshold level. We plan to investigate the optimal confidence threshold that would be considered for deciding whether the user requests understanding is reliable or not. We also want to understand the dependency of this optimal threshold and training data set on automatic user request understanding.

1.3 Contributions

Our thesis work provides 4 distinct contributions as follows:

- Literature survey of chatbots and human computation in chatbot
- System design of a hybrid chatbot system for understanding user requests
- An approach to collect high quality and diverse training data for a chatbot with the help of human computation
- An approach to identify reliability threshold for automatic natural language understanding

1.4 Thesis Outline

We organize the thesis work as follows: First, we describe the background related work in Chapter 2 where chatbot background, it's architecture, typical problems and human computation to solve these problems are described. In Chapter 3, we first provide the hybrid system the design followed by implementation of such system for a particular use-case. Chapters 4 and 5 are about involving humans in hybrid chatbot in an efficient manner. In Chapter 4, We discuss about an approach of high quality data collection, its experimental set up and the results of the experiment. Reliability estimation approach, its experimental set up and experiment results are discussed in the Chapter 5. Moreover, the results and limitations are discussed in Chapter 6 and then we conclude the thesis with describing the future scope of this thesis work in Chapter 7.

Chapter 2

Background and Related work

In order to answer the RQ1 described in Chapter 1, we look into the background and architecture of a chatbot. We also study the past work done in the field of the chatbot, human computation and human computation applied for a chatbot.

2.1 Chatbot

People converse with each other to exchange information and thoughts, to discuss an issue, resolve conflicts, increase mutual understanding or just for fun. Conversation plays a vital role in a communication of one human with another human. Humans can comfortably express their thoughts in natural language. People converse and produce utterance of their conversations. Utterances in a conversation are related to each other in different ways. These different ways are represented with the classification of utterances by John Searle [66]. Table 2.1 shows the classification of utterances with examples.

These classes of utterances help to provide the context, sequence and understanding of the conversation. People just do not produce utterances while conversing with each other, they perform some actions by asking questions, making promises, sharing compliments etc. And while interacting in natural language, they express and convey much more than just the meanings of the words spoken[50]. Their speech conveys their emotional state and their personality. Face-to-face interaction conveys meaning with their nonverbal behaviors such as their gestures, facial expression and body posture. This makes conversation more expressive and understandable, and people can effectively share their thoughts and provide

Class	**Description**	**Examples**
Greetings	To open the conversation by user or by agent	Hello, Welcome
Assertives	Committing the speaker to something's being the case	Suggesting, putting forward, swearing, boasting, concluding
Directives	Attempts by the speaker to get the addressee to do something	Asking, ordering, requesting, inviting, advising, begging
Commisives	Committing the user to some future course of action	Promising, planning, vowing, betting, opposing
Expressives	Expressing the psychological state of the speaker about a state of affairs	Thanking, apologizing, welcoming, deploring
Declarations	Bringing about different state of world by utterance	I quit, you are fired, You are talking rubbish

Table 2.1: Classification of Utterances

information to each other. Can we also converse with computers to make them better understand us?

The idea of being able to hold a conversation with a computer has fascinated people since a long time. Conversational interfaces enable people to interact with the computer using spoken language just like engaging in a conversation with a person in a natural way. They can share information, answer questions, ask questions and even perform some actions for users. Conversational interfaces have a long history; it started in the '60 with text-based interface. Speech-based dialog systems began in the late '80 and spoken dialog technology became a key area of research within the speech and language communities [50]. At the same time, Voice User Interfaces (VUI) emerged in commercial spoken dialogue systems. After that Embodied conversational agents (ECA) were developed which support facial expression, gestures, body stance and speech in order to provide more human-like and more engaging interactions. Various technological advances in the fields of artificial intelligence techniques, language technologies, an emergence of semantic web and rise in usage of mobile applications have contributed to a growth of conversational interfaces.

2.1.1 Conversational Agents

Conversational agents are software programs those support conversational interaction between humans and machines in natural language. They help, assist, organize, manage or perform user's day-to-day tasks like booking airline/movie tickets, participating in meetings and conferences, knowing about weather conditions, getting updates about traffic in the city, checking for the best restaurants/shops around, talking to customer care service and many more. Conversational agent is a collection of behavioral components that can interpret, sense, trigger and respond or reply to user.

2.1.2 Classification of Conversational Agents

In this section, we represent classes of a conversational agents. By this classification, we understand how the conversational agents are used on different devices and what are their mediums of communication with the users. We also looked at how they are used for different purposes with different domains. Please refer to Figure 2.1 which shows the classification of conversational agent.

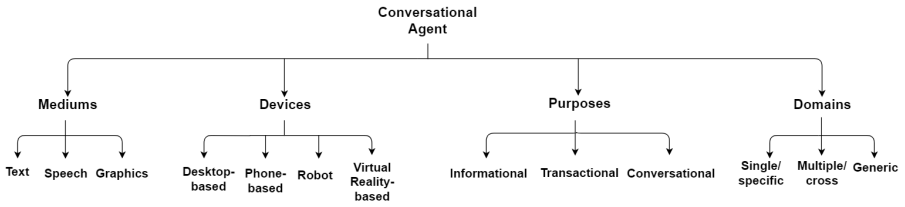


Figure 2.1: Classification of Conversational Agents

Mediums

Mediums of communication is a channel of sensory input/output between a user and conversational agent. The interaction between a user and conversational agent can be text-based or speech-based or it can also be based on a graphical user interface. In Text-based interaction input-output is mainly in text, in spoken dialogue systems it is speech is and in graphical user interface, interaction is based on graphical elements. The system can also be developed for multiple mediums.

Devices

Previously, conversational agents were used on desktops or laptops. With the increasing usage of mobile phones, they are now widely used in phone based applications. The robot is another type of device which is capable of carrying out a complex series of actions automatically. Virtual reality devices are realistic simulation of a three-dimensional environment that are created using interactive software and hardware. They can be experienced or controlled by the movement of the body. This is the very effective way for the interaction with the user.

Purposes

Conversational agent might serve 3 purposes. First one is *informational* conversational agents those try to provide information to its users like travel information and timetable, weather information etc. They are used in Question-Answering systems to provide informational answers to user's questions. Second is *transactional* where users can perform actions with conversational agents such as purchasing a train or movie ticket, reserving a table in a restaurant etc. And third is *conversational* which is designed just for the sake of conversation with their users.

Domains

Conversational agents are used in different domains. Based on the requirement of an application, they can be designed and developed for different domains like weather information, transport/travel information, educational domains like tutors [49], medical domain like clinical uses for mental health [52]. We call conversational agents which serve in only one domain as single domain agents; Multiple domains agent which serves in many domains, and we call them generic, if they serve for general conversations like small talk or conversation.

2.1.3 Chatbot as a Text-based Conversational Agent

A text-based conversational agent is called a *chatbot*. It is using text as the only medium of communication and is used on desktops/laptops, majorly on mobile phones in different messaging applications. Messaging applications such as Facebook Messenger, WeChat, Telegram, LINE are widely used by millions of people to communicate with their friends, colleagues and companies [28]. The growing popularity of messaging platforms is driving a new era of interaction with chatbots. Messaging platforms have begun to provide functionality to develop and discover different chatbots. We currently recognize two ways of using chatbots; first are those integrated into a messaging application and second, those are using existing platforms, such as Facebook Messenger, Telegram, Slack, WeChat, Viber etc. The difference is, for the second type, you can chat with the chatbots via already existing messaging application on your phone, but for the first one, you need to install and use a dedicated application to converse with a chatbot. Integrating chatbots allows these applications to communicate on the scale of one-to-one or one-to-many. Some of these messaging platforms like Telegram, Facebook Messenger, Twitter are widely used as they provide their own framework for developers to develop chatbots and integrate them easily into a messaging application. We perform a short survey on four widely used messaging platforms so as to understand the methodology they provide while developing a chatbot, their features, interfaces and functionalities. We perform this survey to understand which messaging platform would be a suitable platform for implementing a chatbot in our thesis work. Table 2.2 shows messaging platforms and their different attributes.

These messaging platform frameworks make the development and integration of a chatbot into different applications easy and also provides the nice user interface. And they are

	Telegram	**Facebook Messenger**	**WhatsApp**	**Twitter**
Requirements/ set up	Get API token from BotFather	Set up and integrate webhook	Using some reverse-engineered a library like Yowsup that enables developers to build an application which use WhatsApp Service	Getting keys and access Token, Account verification by phone number
Input Format	Text, graphics	Text,speech, graphics	Text,graphics	Tweets
Output Format	Text, graphics	Text, speech, graphics	Text, graphics	Automated tweets, Direct messages, Re-tweets
Starting Point	By the user on mobile phone/ Telegram Web, User should search the Bot	By the user on mobile phone, by searching in messenger/ scanning code/ Linked ads	By user, on mobile phone/ web WhatsApp by searching for the particular phone number associated with the bot	Automatic following, Bot automatically responses to the user
Communication management	Communication with intermediary telegram server via simple HTTPS interface	Webhook, The Web Service API, web plugin to embed Messenger bots right on your website	Interception of incoming messages from a user and then making an API call to the server and returning the response to that message	REST API queries over a long-lived HTTP connection. Receives updates on the latest Tweets matching a search query
Development support	Telegram API	Own API named as Wit.ai Bot Engine	No own official API, some libraries are available	Twitter API
Message Updates	Messages updates in JSON format with Message information and logs	POST logs in JSON, Entries in Webhook	Detects the commands which provides the results	With your own tweet repository for your bot and logs
Special features	Inline mode, bot can be added in group chat	Prediction for next actions, Natural Language Understanding Assistance, AI features	No special features rather it has some terms and conditions and issues	Game and toys development, domain expertise, IoT interfaces, virtual assistance

Table 2.2: Survey of Existing Messaging Platforms

used in wide range of applications such as **E-commerce** in which chatbots can answer

queries from the users about particular products and services. They also guide users in selling the article and even order it from company web page. In **website navigation,** chatbots help users for navigating and finding information in complex websites with a very high number of links. For the **help desk**, chatbots help users by providing guidance and help about certain technical problems with a product.

2.1.4 Architecture of a Chatbot

Here in this section we introduce architectural components of a chatbot and their interconnection between each other [50]. Figure 2.2 shows the architecture of a chatbot.

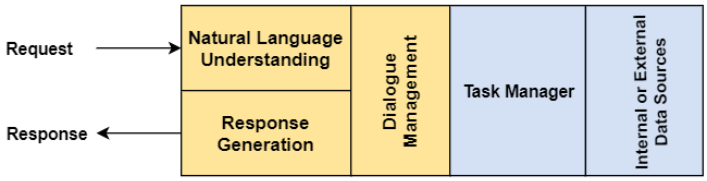


Figure 2.2: Architecture of a Chatbot

A chatbot gets a request from a user, then this request is first interpreted to infer user's intention and associated information by Natural Language Understanding (NLU) component. When user input is understood, based on the purpose of a chatbot, further action is taken. Dialogue manager keeps the state of the conversation to know if the request is related to previous conversation or brings some a new topic in the conversation. If chatbot is informational, required information is retrieved from its data sources. If chatbot is conversational, required response is retrieved from the data sources or with the chatbot's own strategy to answer the user. If the chatbot is designed for transactional purposes, then further actions are executed by task manager. Task manager executes the action which user asked to perform or retrieves the required information from the data sources to provide to the user. After this, a response message is generated by response generation component and sent back to the user. This is the high-level description of different components of a chatbot. Functioning of each component is discussed in detail in sub-sections below. We divide the architectural components in 2 groups. In first group (Group1- shown in yellow color in the Figure) we will have 'Natural Language Understanding', 'Dialogue Manager ' and 'Response Generation component', while in the second group (Group2- Shown in the blue color in the Figure) we will have 'Internal/External Data Sources' and 'Task Manager'. In this thesis, we concentrate more on the first group as we are focusing on components which are related to natural language conversation.

Natural Language Understanding

Every incoming user request is first processed by natural language understanding component. Different strategies are used for Natural language understanding in a chatbot. Natural language understanding component parses the user request and tries to get a meaning out of it by understanding user's intention and information associated with that intent. Some chatbots first pre-process incoming requests with natural language processing techniques to get structured data from the user request. First, in *tokenization* user input is divided into tokens which can be used for further processing and understanding. Tokens can be words, numbers, identifiers or punctuation. For example tokenization for user request 'What is the color of that girl's car?' is divided into separate tokens as: 'What' 'is' 'the' 'color' 'of' 'that' 'girl' ''s' 'car' '?'. Then *lemmatization* uses a language dictionary to perform an accurate reduction to root words. Lemmatization for the above user request is 'What be the color of that girl 's car ?'. *Stemming* uses simple pattern matching to simply strip suffixes

of tokens such as 'what is the color of that girl ' s car ?'. *Entity extraction* is identifying and extracting entities (people, places, companies, etc.) from the user request. After this pre-processing, structured data can be fed to natural language understanding strategies which actually take out the meaning of the structured data. Chatbots sometimes directly involve natural language understanding strategies by skipping the pre-processing of user requests and feeding user request directly to NLU unit as it is. Natural language understanding strategies used in chatbots are:

- **Rule-based Pattern matching**: Here chatbots use pattern matching to classify the text and match the user input to already defined patterns. Many chatbots use 'Artificial Intelligence Mark-up Language '(AIML) for such pattern matching [76, 63]. Chatbot tries to match user request to the AIML predefined pattern template. Every pattern template has the possible response associated with it. Chatbots can also use their own pattern-matching methods by applying some heuristics. This pattern matching methods may vary with the requirement and scope of the chatbot.
- **Machine Learning**: NLU unit here tries to parse the user requests and classifies it in categories to correctly understand the user. The first category is *Intents*. Intents represent a mapping between user 's intention and the related action that should be performed by the chatbot. Second is *Entities* those represent concepts that are often specific to a particular domain. Entities are the natural language phrases in the requests that can be mapped to canonical phrases which collectively capture the meaning of a user request. And the third category is *Actions* specifying the steps that chatbot will take when specific intents are triggered by user inputs. An action contains parameters for specifying detailed information about it. Machine learning strategies already have the trained model with some initial training data including a set of intents, entities and actions. Every incoming request is tested against this trained model.

Nowadays various automatic tools, frameworks and APIs are available for natural language understanding for the different methods mentioned above. Some of these tools and APIs are: RASA NLU, Wit.ai, API.ai, Microsoft LUIS, AIML, Google Natural Language API, Init AI, ChatScript etc. These tools simplify the natural language understanding process in a chatbot.

Dialogue Manager

It maintains the state of a dialogue. Dialogue manager is responsible for handling the context of the conversation which is basically a current state of the user expression. It helps chatbot to actually understand that every incoming statement is in a connection with previous statement or answer of a bot's follow-up question or all together a new statement. Dialogue manager coordinates with all other components of a chatbot. Basically, Dialogue manager is involved in four main tasks [69] such as updating context of the conversation, providing a context for interpretation, co-ordinating other components and deciding the information to convey and when to do it. Following are the dialogue management strategies are used for handling conversations in a chatbot:

- **Finite-state dialogue management**: In this strategy has the states of conversation that are already defined and dialogue manager keeps the state of the conversational interaction according to these states. It has a finite state sequence and chatbot can exactly know at which state the conversational statement is. This is the simplest method of designing a dialogue manager [57].
- **Frame-based or Form-based dialogue management**: With a frame-based dialogue manager, a chatbot asks set of questions to a user and keeps the state of every

question-answer pair. This method is similar to finite-state dialogue management; the difference is that it has a finite series of questions to ask to the user.

- **Initiative-based dialogue management**: This dialogue management is taken care by the initiative strategy. The initiative is all about 'who has the control of the conversation'. When a user-directed initiative is used, the user always starts the conversation, and chatbot just responds to the requests asked by users [20]. In system-directed initiative chatbot starts and controls the conversation. In the system-directed communication, the system completely controls the conversation with the user. It asks the user a series of questions and it just ignores anything which is not a direct answer to the system's question, and moves to the next question. The main advantage of this strategy is to restrict the user's input, having more efficient set of conversation. Conversational initiative can shift between the system and user for the mixed initiative strategy. The advantage is that the chatbot can provide answers to the user requests, and the user can also take initiative by asking questions, providing responses and introducing new topics. [21].

Task Manager

Task manager executes actions based on the purpose of a chatbot. It actually carries out the sequence of steps which are necessary to perform a particular action. In the case of transactional chatbots, task manager executes the tasks with the help of internal or external data sources, e.g. purchasing a train ticket from train application. Task manager acts as an information retrieval manager in case of informational chatbots. It performs an action of information retrieval from internal or external data sources. This same applies for conversational chatbots while retrieving data from input-output templates or from a database.

Data Sources

Chatbots use different data sources according to its requirements and domain. They sometimes use available AIML templates or available rules structure for understanding user requests and provide answers accordingly. A Chatbot can also have it's own database that is built from scratch, or it may use already available databases matching with its domain and purpose. We call these databases as internal data sources of a chatbot.

Chatbots sometimes use external data sources such as third party services like Web APIs to provide the required information. For example, chatbots use weather APIs like 'openweathermap' to provide the current weather information and weather forecast. It can also rely on external applications like regional transportation applications, weather applications etc. to provide the information to users when chatbot is informational. In the case of the transactional chatbot, task manager co-ordinates with these external services or applications to perform some tasks, e.g. performing a task of booking hotel with booking.com

Response Generation

After information is retrieved or an action is performed, the chatbot needs to prepare a response message. This task is done by response generation component. There are two types of the response generation models as follow.

- **Retrieval based models**: These models use a repository of predefined responses to answer user requests. They use a predefined heuristic to pick an appropriate response based on the user request and context of the request. The heuristic can be as simple as a rule-based expression match [31], e.g. specified rule as [MY NAME IS <regex> [a-z]+ [a-z]], where 'regex' is regular expression and required name and surname containing 'a' to 'z' characters are generated. It can be a rule-based pattern matching

like AIML e.g. [<pattern>BYE * SEE YOU</pattern>], where '*' can be used for any other phrase, but the remaining pattern is same. These two methods are simple but the response generation in retrieval based models can be as complex as an involvement of machine learning classifiers. These systems don't generate any new text, they just pick a response from a fixed set of vocabularies.

- **Generative models**: Generative models don't rely on predefined responses. They try to generate new responses from scratch. Generative models are typically based on Machine Translation techniques [48]. Machine translation techniques usually have strategies for translating an input from one language to another. But in the generative models, user inputs are translated to a *response* that chatbot provides to the user, based on machine translation techniques.

2.1.5 Chatbot Related Challenges

Despite popularity and large growth in chatbots, they still face a lot of challenges and far from being perfect. In this section, we try to understand the challenges those current chatbots face. To analyze these challenges thoroughly, we try to discuss them based on architectural components introduced in subsection 2.1.4.

Natural Language Understanding

Rule-based pattern matching techniques and machine learning techniques both need a large amount of rules and training data to understand diverse user requests. And even if we provide a large amount of data to them, they still face challenges in understanding user requests correctly. Machine learning techniques are not perfect in identifying intents and extracting entities. They miss some entities and predict incorrect intents though they are trained well. Natural language processing techniques face challenges in processing complex and long tail user requests as those are hard to parse and they can conclude little from the user requests which again stop agent to proceed in right direction. Current natural language understanding techniques work reasonably well for formal English language but it is challenging to achieve same performance for other languages and language variations [2].

Dialogue Manager

Chatbots struggle to differentiate the new state of a user and cannot keep the context of the conversation. For the finite-state dialogue management and frame-based dialogue management, it is easy to keep track of the conversational states, but they are not flexible as they rely only on the previously defined states. The user just cannot go beyond these states of conversation. Initiative based strategies also come with their own challenges. When the user-directed initiative is used, then the users are free to say whatever they want and it becomes very hard for NLU to understand user requests and for a dialogue manager to maintain their context. Similarly, system-directed initiative is not flexible, as the user is restricted with the chatbot's expectations. It becomes very hard to keep track when chatbot system in the mixed initiative as dialogue manager just cannot predict when and what user will ask. In this strategy, the user can potentially say anything, ask random questions, introduce new topics etc. But this may cause the dialogue manager to lose track of the conversation. These limitations lead to various problems like interpreting the wrong context, going away from the topic of a conversation, entering into infinite loops of conversations.

Task Manager

Task manager can perform the tasks which are easy and straightforward, but it faces difficulties in performing tasks which are long, complex and not accessible due to some data

security issues. For information retrieval tasks, sometimes task manager does not have access to the information to be retrieved.

Data Sources

While designing and implementing a totally new chatbot for a new domain, choice of the database is an important aspect. The database should be flexible and scalable enough to accommodate a large number of user requests and possible responses. As the same set of conversations are rarely going to repeat with a particular user, chatbots should be able to handle new set of requests from users. To achieve this, data sources should be scalable enough to serve a huge number of diverse user requests. However, the well-defined structure of the data sources of current chatbots comes with a high cost of extending it [70]. For external services and applications, chatbots not always have an access to information to be retrieved and then it cannot retrieve the required information.

Response Generation

A chatbot sometimes just fails in understanding how to represent information it retrieved in an understandable way. The information retrieved from external APIs is not always in an understandable format and response generation component just cannot process it correctly and generate an understandable response from it. As we see that there are different models are used for response generation in a chatbot, they are still facing challenges. Retrieval based techniques are easy to implement in a closed domain with rule-based pattern matching, but they are very hard to implement in the open domain, as it will need a large amount of rules to produce appropriate responses for diverse requests. Response generation techniques for an open domain are also very hard to implement, as generating correct responses belonging to correct domain is very challenging for generative models.

These challenges in chatbots prove that they still have lots of space of improvement. Most of the chatbots face challenges in natural language conversation that is in understanding user natural language request, following a consistent dialogue and response back in natural language. Existing chatbot systems rely on fixed input vocabulary, restricted phrasings, the limited amount of training data, the limited memory of past interactions and fixed output vocabulary. While, people can contain a large amount of context dependent phrasing, past history of interactions and shared experiences, partial or incomplete but meaningful sentences [42] in the real conversations. These conversational aspects are difficult for a chatbot to understand but very easy for a human to understand and maintain natural language conversation. So what if we take a help of humans to solve these problems in a chatbot? This can be done by means of Human Computation.

2.2 Human Computation

In this section, we discuss human computation in a chatbot. We first introduce a Human computation, quality and latency aspects of human computation and related work done in these areas. Then we discuss the past work done in the field of chatbots using human computation and their challenges.

A paradigm for utilizing human processing power to solve problems that machines cannot yet solve effectively is called as a human computation. Human computation is a phenomenon where a machine performs its function by outsourcing certain steps to humans [71]. The initial reference to this concept can be found in 2003 by Ahn et al. [72] while introducing the concept of the captcha. Then various Human computation definitions have been introduced. Quinn introduced it as ‘systems of computers and large numbers of humans that work together in order to solve problems that could not be solved by either

computers or humans alone' in 2009 [58], Chandrasekar in 2010 defined it as 'a new research area that studies the process of channelling the vast internet population to perform tasks or provide data towards solving difficult problems that no known efficient computer algorithms can yet solve' [30]. With the growth of human computation, there are many human computation platforms are also developed which can be mainly categorized based on their nature of collaboration as explicit and implicit platforms [13]. In explicit platforms, workers are aware that they are collaborating in a task. Mechanical Turk(MTurk) and CrowdFlower are commercial explicit platforms. In implicit platforms such as ESP game or Recaptcha, nature of collaboration is implicit because participating users are not aware of the fact that they are participating in human computation process.

Human computation takes place with the intention of attracting individuals to make a contribution to the firm's production process for free or significantly less cost [35] by outsourcing particular tasks to the general public in the form of an open call over the internet. In such a scenario, both humans and computers work together as one cannot solve without the contribution of the other [71]. Large problems can be divided into small problems called as tasks. Human computation platforms allow us to distribute these tasks to human workers [29] and benefit from their intelligence, creativity and sensing abilities. To distribute a task, the requester who is the owner of a task submits the task to a human computation platform [3]. Workers accomplish the task and choose to work on it and devise solutions. Tasks are performed partially or completely by humans. The requester then assesses the contributions of the workers and rewards those workers whose contributions is satisfied and can be accepted. The reward can be monetary, material, psychological or some other. An outcome of these tasks can be one or more individual contributions or a combination of accepted contributions. Requesters choose contributions that have reached the expected level of quality.

Human computation is growing in multidisciplinary fields like data and knowledge management, system development, social sciences and computer supported collaboration applications. Human computation has various real-life applications such as monolingual book translation platform using human computation [23], Word Sense Disambiguation [67], Captcha, the ESP game, Peekaboom, Verbosity and Phetch [72] and many more.

2.2.1 Quality Aspect of Human Computation

To conduct Human computation activities efficiently, we must understand how the quality of these activities can be controlled and maintained. In the context of this thesis, we mainly concentrate on 'quality' aspects of human computation by means of an accuracy of contributions produced by human workers and speed/latency in producing those contributions. In short, we mainly focus on an assurance of quality contributions in a timely manner. We first discuss accuracy aspect and then speed aspect.

Accuracy

There are different factors which contribute to achieving the accuracy. The overall accuracy of outcome of a task depends on the task design and the contributing workers' profiles [32, 33]. These two have their individual measures described by Allahbakhsh [3]. Task design measures are task definition, user interface, granularity and incentives and compensation policy. *Task definition* should have a clear and precise description of a task, explaining its nature, time limitations and qualification requirement of workers contributing to the task. The *user interface* should be user-friendly and attractive to engage more workers to contribute to the task. The complex task can be divided into *simple sub-tasks* where human workers perform these sub-tasks and complete them, and then completed sub-tasks are chained together to build overall solution of the main task. These sub-tasks

can be carried out in parallel or iteratively [37, 46]. These work flow design strategies affect the outcome quality greatly [32, 33]. Task design should consider fair and suitable incentives and compensations for workers. Unlike machines, humans need a motivation to perform these tasks. Previous studies show that encouraging people to participate and to produce quality data is a challenging procedure [16, 7]. So there is a need for right incentives to motivate workers in performing tasks and provide quality contributions. These task design measures should be analyzed carefully while designing the task. Along with the task design, workers should be carefully selected based on their profiles who would produce better contributions. Doing so may require requesters to filter workers by their profiles. Reputation and expertise are the two measures of workers' profiles. *Reputation* in this context is relationship of trust between a requester and some specific worker. This reputation reflects the probability of the worker to contribute as expected by the requester. *Expertise* is worker's ability to perform a particular task. Identifying these two measures correctly would help in selecting efficient workers and eliminating cheaters.

Researchers and practitioners have proposed several quality control strategies that fall under different quality dimensions and factors of human computation [3]. Some strategies to identify the quality measures for task design and workers are suggested by [19]. These strategies are either reactive or proactive. *Reactive* strategies act when task contributions do not meet given quality thresholds and *proactive* strategies do not need a triggering event to be applied. Based on this idea, we divide these strategies into two categories first is at design time which is proactive and second is at run time which is reactive. These categories together try reach the higher possibility of receiving high-quality outcomes. Table 2.3 shows the different existing quality control strategies at both design and run time for two different quality aspects as task design and workers pre-selection. [3].

Latency

Latency in human computation can be defined as a delay in completion of a task. Current human computation applications face latency issues due to the lengthy process of identifying and recruiting suitable workers [51]. Thus, both practitioners and researchers are finding new techniques of handling latency challenges in human computation processes. Latency issues are critical when human computation applications are leveraged in real-time applications. Early human computation systems leveraged human computation through batches of tasks that were taking hours or days to complete the tasks. Approaches like ESP Game [73] paired workers synchronously and allowed them to play an interactive image-label guessing game. But this did not provide a response for an individual label with low latency. These approaches take a lot of time and hence they are not always suitable in interactive real time applications.

VizWiz [6] was one of the first systems that provided the near real-time response from human workers to its users. It introduced a queuing model which ensured that the workers were available quickly and on demand. Prior systems have shown that multiple workers can be recruited for the collaboration. This was done by waiting for sufficient number of workers [10, 73]. The strategy of queuing and waiting to recruit workers from existing sources of workers collectively could reduce the latency which was proved by Bernstein et al. [4]. Further work using this queuing strategy showed that the latency can be reduced to a second and can also provide reliability on using human computation in this manner [5].

Lasecki et al. [40] introduced continuous real-time human computation in Legion, a system that allowed workers to interact with a UI control task over an unbounded, on-going task. The average response latency of control actions in Legion was observed under a second. Scribe [38] provides real-time captions for deaf and hard of hearing users. System has latency per word less than 3 seconds. New tasks-specific mediation strategies that further

	Approach	Description	Application
Design Time	Effective task design preparation	Provides an clear description of the task; defines effective evaluation and compensation criteria	[59, 33, 8, 14]
	Worker selection- Open to all workers	Everyone can contribute in the task	ESP Game [73]
	Worker selection- Based on the workers' reputation	workers only with pre-specified reputation are allowed	[65]
	Worker selection- Based on workers' credentials	workers only with pre-specified credentials are allowed	[65]
Run Time	Output agreement	Contributions are considered to be correct, if workers provide same output independently and simultaneously for a particular input	ESP Game [73]
	Input agreement	Workers receive an input and describe it to each other. Mutually decided input is accepted as a quality answer	Tag-A-Tune [43]
	Ground truth	Compares answers with the known answers to check the quality	CrowdFlower, MTurk
	Majority consensus	The judgments with a majority are accepted	TurKit [46], MTurk
	Real-time support	Supports workers in real time to increase their contribution quality	[14]
	Workflow management	A suitable workflow for a complex task is designed and monitored	[34, 37]
	Contributor evaluation	Contributions are evaluated based on the contributor's quality	MTurk

Table 2.3: Existing Quality Control Approaches in Human Computation [3]

reduce overall task completion time in robotic control tasks are provided by Salisbury et al. [62]. These different techniques have been proved as considerable advancements in handling latency issues in human computation.

2.2.2 Human Computation Applied for Chatbots

Advantages of human computation techniques, platforms and their problem-solving capacities can be blended into chatbots to solve the chatbot challenges. We call chatbots which utilize human computation in at least one of the architectural components (refer Figure 2.2) of a chatbot as 'Chatbots having human in the loop', we call such systems as 'a Hybrid Chatbot Systems'. In our thesis, we focus more on natural language understanding related components, hence we focus more on the literature providing the information of chatbots using human computation in these components. In this section we look at a literature where chatbot systems are developed using humans in the loop. We analyze their architecture, working and different styles of using human computation in them.

We represent our literature review in Table 2.4 which shows the study of different existing chatbot systems using human computation in them in various components. the purpose of systems which is informational, transactional or conversational which are already described in 2.1.2. We then divide each chatbot example into two categories such as 'Human-powered chatbot' and 'Human-assisted chatbot'. We call chatbots as human-powered which are using only humans for performing different architectural tasks related to the three natural language understanding components in group 1 we introduced in 2.1.4. Another category is human-assisted chatbots which are using human computation in combination with some automatic techniques. We split the domains of chatbots as generic where chatbots are ready to answer nearly any user request; crossed-domain in which chatbots can serve user requests in multiple domains and third is domain-specific where chatbots operate in a single domain. For each chatbot using humans in the loop, we describe how natural language understanding, dialogue management and response generation techniques work. At the end, we provide the human computation specifications used in the particular chatbot. This study is inspired by chatbot related work presented by [56].

From the Table 2.4 we understood various examples that involve human computation for taking help in different architectural components and in different phases of a chatbot. In our thesis, we mainly focus on understanding user request and hence we want to thoroughly study how the natural language understanding strategies are designed in these chatbot systems. We can see that *Chorus*, *Instructable Crowd* and *Legion* rely completely on human workers for understanding user requests. systems like *Guardian*, *AskWiz*, *FacebookM* and *Microsoft Calendar.help* have machine learning and human computation approach for understanding user requests. *SnapTravel* and *Insurify* utilize pattern matching techniques too along with machine learning and taking help from humans. When the chatbot systems having humans in the loop utilize the combination of different strategies, we want to know how they decide when to use which strategy. We want to learn if there is any condition to decide to go from machine learning to human computation and how this condition is decided and achieved in practice. We want to know if the condition is based on some threshold level or it is based on some qualitative evaluation, as understanding this aspect is important in our thesis.

When machine learning approach is used for natural language understanding of a chatbot, it mainly tries to identify the intent of a user request and extracts the entities in the request as we have discussed in Section 2.1.4. With the help of the already trained model, natural language understanding module processes every incoming request by extracting the intention of a request and relevant information from the request. It has an output as a prediction of identified class and extracted entities with some confidence measure. This

Chatbot	Purpose	Domain	NLU	Dialogue Management	Response Generation	HC Category	Worker Source	Real-time Support	Scalability
Chorus- Personal Conversational Assistant [42]	Conversational, Informational	Generic	By human workers	Crowd workers handle entire full conversation session, hence they can keep the conversation context	Workers propose response, vote on each other's responses to decide which one to send to the user	Human-powered	Crowd workers (MTurk)	Task redundancy	Depends on human computation platform- MTurk
Instructable Crowd- A mobile application to create trigger-action rules [25]	Transactional	Domain specific	By human workers	Dialogue is built managed around creating if/then rules, context kept by the crowd workers	By human workers	Human-powered	Crowd workers (MTurk)	Task redundancy	Depends on human computation platform- MTurk
Legion- A mobile application for helping visually impaired people to control their phones with voice cues [40]	Transactional	Domain specific	By human workers	Dialogue is build around performing actions that user wants, context kept by crowd workers	Mediator selecting appropriate action to take based on inputs of other workers	Human-powered	Crowd workers (MTurk)	Retainer model	Depends on human computation platform- MTurk
Guardian- Web-based chatbots. Crowd maps parameters from the user conversations to set of web APIs [26]	Informational, Transactional	Cross-domain	Human workers + Machine learning	Dialogue is build around identifying and mapping conversation to API, context kept by crowd workers	By human workers- clarifying parameters and replying based on web API results	Human-assisted	Crowd workers (MTurk)	Retainer model	Depends on human computation platform- MTurk
AskWiz- Facebook messenger based tech support agent, which helps in solving technical queries of users [goo.gl/RWMRf6]	Informational	Cross-domain	Human workers + Machine learning	Tech queries conversation sessions are handled by crowd workers.	Machine learning + Human Workers	Human-assisted	Freelance remote agents	Near-real time support	Hiring more agents
FacebookM- Facebook messenger based personal assistant performing custom tasks [22]	Informational, Transactional	Generic	Human workers + Machine learning	Human workers + Machine learning	Machine learning + Human Workers	Human-assisted	Full-time employees	Workers working in shifts	Hiring more employees
Microsoft Calendar.help- email based personal assistant for scheduling meetings considering the suitable timing for all participants [11]	Transactional	Domain specific	Machine learning + Human workers for microtasks	Machine learning + macro-tasks handled by crowd workers	Machine learning + human workers	Human-assisted	Microsoft crowdsourcing platform	Workers working in shifts	Up to Microsoft crowdsourcing platform
Insurify-Facebook messenger based personal assistant suggesting insurance quotes [https://insurify.com]	Informational, Transactional	Domain specific	Pattern matching+ Machine learning + Human workers	Human workers'responses to user requests are fed back to the machine learning algorithms	Machine learning + human workers	Human-assisted	Third party insurance agents	N/A	Hiring more insurance agents

Table 2.4: Existing Chatbot Systems Using Human Computation

confidence measure expresses how much the model is confident about the prediction. This confidence value is important to know if the information is correctly extracted. And this value could be a possible way to decide if to trust on the automatic prediction or ask humans for help.

We want to learn in detail and study the past work done about the machine learning techniques for natural language understanding in a chatbot, starting from the training data generation for NLU model to estimating if the prediction of understanding of user request, generated by this model is trustable or not.

There are various ways to generate a training data set for chatbots. Some methods suggest to think as a user and produce training data that you would expect to get as an request to your chatbot, some suggests to conduct surveys and collect the training data by providing these surveys to end-users. The methods to collect training data and train a chatbot are provided by [79] in different stages. The method of data collection with the help of human computation is proved as an effective approach. Gathering data using human computation platform has become now a mainstream process [27]. It provides a scalable way to gather data that can be used to train and evaluate machine learning systems. Collecting examples of possible user requests to train a chatbot is a creative task on human computation platform, which asks workers to brainstorm the possible user requests they can ask to the chatbot. Such creative tasks can be designed by guiding with some element of a the idea space [24]. Workers on human computation platform can provide analogies for finding out the solutions from one domain to solve problems in another, with large set of ideas [81] and those analogies can provide seed for more innovative ideas [80]. Some systems focused on maximizing quality or creativity [68] while some systems such as BlueSky [24] focused on the complete and uniform coverage of examples or ideas. The combination of both should be considered to train a chatbot with high quality and diverse user requests.

Once NLU model is trained with the training data, it can predict the meaning of incoming user requests. But it is important to judge if this prediction is trustable or not. One of the effective methods is to judge the prediction based on confidence scores associated with prediction. This score provides a numerical value of how confident the system in finding out the result. Many natural language classifier tools and systems provides the confidence score of their classification. It cannot be assumed that all the results generated with high confidence score are correct and all the results generated with low confidence score are incorrect. Hence the threshold should be identified and assigned to identify the reliability of identified results. Developers of IBM Watson discussed the method to estimate confidence threshold by assigning numerical reward to each possible outcome [53]. They also suggested a method of designing heuristic rules instead of outcome rewards, based on which the threshold values could be decided. The methodology to assess the performance of text classification is provided by [54], which provides an idea of introducing the probability threshold, based on which the predictions can be judged. Some methods suggested to design highly confident rules and make a collective and all-around decision for the predictions [45]. Identifying the reliability of predictions would help in having trusted predictions and the errors can be minimized. There is a need to validate the performance of automatic NLU tools in chatbots, as in context of our study, they are the black boxes having no detailed information of how the meaning of a user request is predicted. We need to understand them and then ask humans for help whenever they do not produce reliable results.

2.2.3 Challenges of Human Computation Applied for Chatbots

Human computation in chatbots seems to provide the better solution for understanding user requests, but there are still some challenges in this method, such as scalability, real-time support and cost efficiency. In this section, we analyze those challenges.

Scalability

Here we divide scalability in two types such as domain scalability and workforce scalability.

- **Domain Scalability**: Human workers help to make chatbots efficient in narrow domain, but sometimes they face difficulties to support charbots in generic and multiple domains. This is because they may lack in knowledge of variety of domains. Expanding a scope for chatbot domain run-time is still very challenging. Workers are provided with precise domain specific information and requirements; they will face problems in analyzing queries which are in a different domain than the original task.
- **Workforce Scalability**: Chatbots like Facebook M serve users only in California, but they are not scalable out of that locality. Existing human-powered or human-assisted chatbots can scale on demand up to a certain limit as they need more workforce to serve increasing number of users. Chatbots like AskWiz which are dependent on freelance remote agents, scale with some delay caused by finding and recruiting new agents. Human-powered chatbots relying on full-time employees struggle the most in scaling up. Lack in a number of workers will cause a delay in completion of tasks and chatbots cannot serve user requests in a timely manner.

Real-time Support

Current human-powered and human-assisted chatbots serve well and try to overcome the challenges those automatic chatbots have. But they still lack in answering user requests in real-time. Many of the chatbots using human computation are able to provide near real-time support, but in the case of a big demand of tasks, human computation can only grow gradually with moderate workers supply. This would reduce the speed of task execution and chatbot cannot serve users well in a real-time manner.

Cost Efficiency

The cost comes with the scalablity. To serve millions of users, chatbots would need more workers to help them. Increasing number of workers would increase the cost. Human-powered chatbots like Chorus always rely on human workers for every interaction with users, this would make the chatbot cost-inefficient. Relying always back on humans would make chatbot system costly even if it is serving in a narrow domain.

Scalability, real-time support and cost efficiency are the aspects where human computation in chatbot have a scope of improvement and research.

2.3 Chapter conclusion

Starting from the conversation and conversational agents, we introduced different classes of conversational agents. Then, we discussed chatbots, which are text-based conversational agent. We discussed their definition and architectural components in detailed manner. This study of architectural components provided us insights how the internal components of a chatbot look like and work. Then we identified challenges in each architectural component.

In next section, we introduced human computation and its quality and latency aspects. We focused on design time and run time strategies for achieving quality outputs from human computation platforms because we wanted to understand how we can involve humans efficiently at the design and run time in a chatbot system. Study of latency aspect helped us

to understand how human computation can be involved in real-time applications for achieving low latency outputs.

Then we studied the existing chatbot systems which involve human computation for understanding user request, dialogue management and generating response refer Table 2.4. We studied their methodology to involve humans in the system where some systems are totally human-dependent, some have combination human computation and automatic techniques. We also studied that how these systems have real-time support and scalability aspect for human computation component. We can consider and apply suitable quality and latency aspect we studied in Subsection 2.2.1 to ensure the quality of responses with low latency in a chatbot system. We then identified the challenges that these chatbot systems still face with respect to domain scalability, worker scalability in human computation platforms, real-time support and cost efficiency.

From this literature survey, we understand that there is need of a chatbot which effectively and efficiently exploits the advantages of humans capabilities to solve it's problems. For this, chatbot should involve human computation with scalability, low latency responses and cost effective methods.

Chapter 3

Design and Implementation of a Hybrid Chatbot System

This chapter describes design and development of a hybrid chatbot system. To address RQ3, we present our vision of chatbot system having humans in the loop and we call it as 'Hybrid Chatbot System'. For this, we first discuss our system components and their interactions. Then we discuss an implementation of a prototype for a specific use case.

3.1 System Design

In the related work, we have looked at various chatbot systems that involve human computation in different architectural components. Some systems solely rely on humans or some have the combination of automatic strategies and humans. The way they utilize human computation in their chatbot system is also different. To understand how this hybrid chatbot system should look like, we first design a higher level view of the system. Figure 3.1 shows the high-level view of a hybrid chatbot system where different architectural components are connected with different tasks in human computation component. Every architectural component can be connected with one or more than one human computation tasks via HC platform adapter for getting help. HC platform adapter allows data exchange between the architectural components and different tasks on human computation platform. In the Figure 3.1, we show one human computation task for each architectural component. Task 1 is for helping natural language understanding component, where chatbot can take help from human computation for understanding user requests, e.g. in understanding user's intention. Dialogue management module can take help from human computation regarding the dialogue management, keeping the context of the conversation like if the current user request is new or it is in the flow with the previous conversation. A dedicated task can be designed for this. Task manager can ask human computation for helping it in retrieving some information like finding out the restaurants nearby or performing some actions. Information retrieval related or performing action related task can be designed for this purpose. Human computation component can help chatbot's data sources component by providing required data from some third party services. And the task related to response generation in human computation can help a chatbot in generating responses to user requests.

In our thesis, we focus on understanding user requests and hence we focus more on the interaction of chatbot and human computation for natural language understanding component. We studied in the related work that different systems have different strategies in understanding user requests. Some systems solely rely on humans, some have a combination of automatic strategies like machine learning and pattern recognition with human computation.*Our goal is to design a system where we can efficiently involve human computation in our automatic chatbot system for understanding user requests.* For this, we first

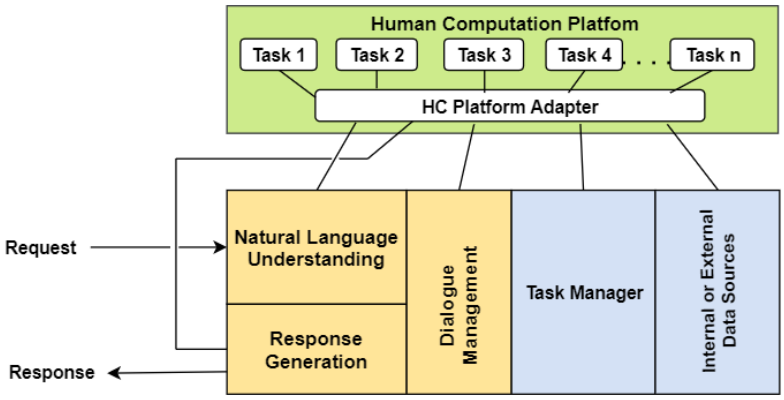


Figure 3.1: High-Level View of a Hybrid Chatbot System

design our components of the hybrid chatbot system and then provide their interaction at both- design phase and run time phase.

3.1.1 Components of the system

In this section, we first design the components of our system. We have an automatic component for natural language understanding, we call it as *Machine Component*. Then we have our second component as *Human Computation* component to help the machine component in understanding user requests. We represent these components in detail in following sub sections.

Machine Component

Machine component is an automatic component used for understanding user requests in a chatbot. Figure 3.2 shows the machine component and its sub components. In our case, we consider machine learning technique for understanding user request and hence we design our sub components according to this. We mainly have 2 sub components. The first is *Trained Automatic model* which is responsible for predicting the meaning of every incoming user request and this prediction is passed to the second sub component called as *Reliability Estimator*. We call this prediction as 'automatic understanding of a user request' by machine component. Reliability estimator component actually decides whether to proceed for further actions with the machine component or involve the human computation component for understanding the user request. It does this by evaluating the prediction given by trained automatic model for the user request. The output of this evaluation represents if the user request understanding by automatic trained model is reliable or not. If it is identified as non-reliable, then control of a system is passed to human computation component. If it is identified as reliable, then further actions are taken by machine component. So the input to machine component is every incoming user request and then reliability estimator component decides whether to take a help from human computation component or not. And the output of the machine component is the *reliably understood meaning of a user request* which then provided to the next architectural component. And if it not reliable, then the request is passed to the human computation component.

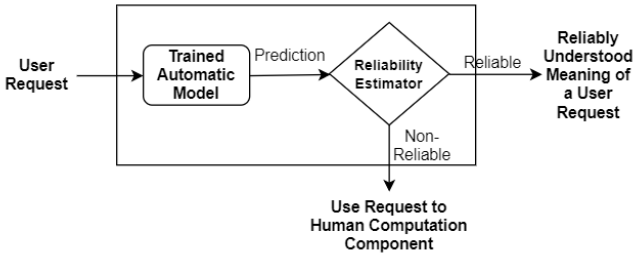


Figure 3.2: Machine Component of a Hybrid Chatbot System for Understanding User Requests

Human Computation Component

Figure 3.3 shows the human computation component and its sub components. Human computation component consists of 2 sub components such as a task and human computation (HC) platform adapter. The task is hosted on human computation platform and the HC platform adapter connects the task to the machine component. This task is designed to help the machine component in understanding the user requests. Human workers perform this task and produce the judgments to provide to machine component. The input data for the task is provided by the HC platform adapter and the judgments produced by human workers are given back to the HC platform adapter. For our system, this task is about helping machine component in understanding user request. But the task can be changed and designed as per requirement of a system. Thus, the input for a human computation component is a user request from a machine component which is provided to the task via HC platform adapter and output is the judgments produced by human workers on human computation platform providing the meaning of a user request. These contributions are provided back to the machine component via HC platform adapter as an output.

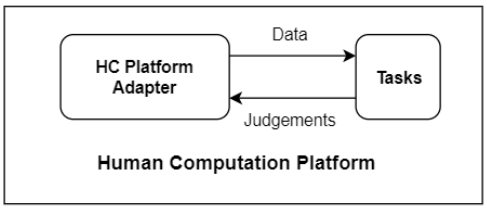


Figure 3.3: Human Computation Component of a Hybrid Chatbot System for Understanding User Requests

This is how we have designed our two system components as *Machine Component* and *Human Computation Component* including their sub components for chatbot user requests understanding. So the input to the machine component is a user request and output is the 'understood user request'which is sent to the next architectural component for performing further actions.

3.1.2 Phases of a Hybrid Chatbot system

In this section, we focus on how the components of the system interact with each other in different phases of the systems. We have two phases of our system such as design time phase and run time phase. We discuss how the system components play their roles in these different phases.

Design Phase

In this design phase, shown in Figure 3.4, we actually make our machine component ready to serve user requests at a run time. And to do this, we take a help from Human Computation component. We train the automatic natural language understanding model in our machine component with a training data. This training data is collected from humans via human computation platform. We provide the seed data examples as an input to the data collection task, where workers can look at these examples and understand the requirements of the judgments they need to provide. We design a data collection task according to the training data requirements. We then train the model with the specified training parameters, we evaluate our model with some validation data, which can also be collected from human computation platform. Evaluation of model would give us the insights how this trained model performs and we can improve it at design time only. So in this phase, we start from the training data requirements for our chatbot, then we collect it with the human computation component to build an automatic training model in machine component.

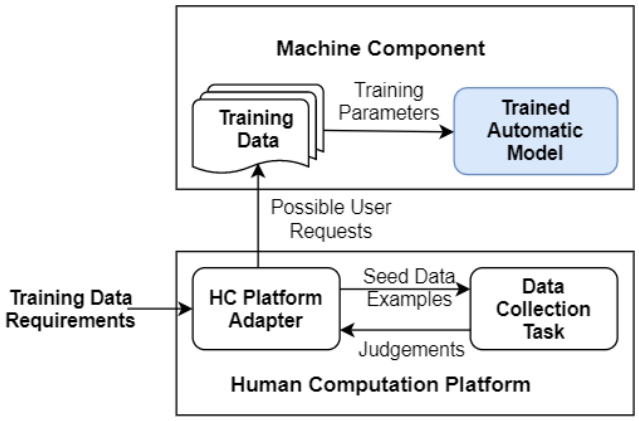


Figure 3.4: Design Phase of a Hybrid Chatbot System for Understanding User Requests

Run time Phase

In run time phase, shown in Figure 3.5, we start with machine component and get help from human computation component when needed. We send a user request to machine component and it first goes to an automatic trained model and predictions are generated for the request with a confidence value. This prediction is then sent to the decision component. Then decision component decides if this prediction generated by automatic training model is satisfactory then further set of actions are performed by next architectural component. And if it is not satisfactory, then we consider it as not reliable, and passed to human computation component.

Human computation component has a user request understanding task which helps machine component to understand the user request. The task is designed for providing a meaningful information for a given user request. Humans on the human computation platform actually perform the task and output of this task provides the meaning of the user request. This task can be designed as per requirement of a chatbot system, e.g. it can be an intent identification task or entity extraction task or identifying missing information in user requests task. The output of these tasks can be identified intention of a user request or identifying entities present in user request or identifying what entities are missing in the user request. Human computation component provides the output of this task for the

particular user request, back to the machine component and helps it to understand what the user has asked or said. We first collect the output of the user request understanding task in *Post-processing component*. It processes this output and sends it to decision component with a higher confidence value. While designing our user request understanding task on human computation platform, we make sure to get quality output for this task. We design a task in such a way that the output we get is correct and can be assigned with a higher confidence value. Decision component then considers this output as a 'reliable understanding of a user request' and this understood user request is then provided to the next architectural components for further actions.

We have one more design aspect in our system. We feed the processed results generated by post-processing component as a training data to the automatic trained model. By retraining automatic trained model with this data generated by human computation component, we believe that we make our machine component ready for handling the similar kind of requests for the next iterations. This is how the data which was unseen or was predicted with low confidence in the previous iteration, would be no more unseen or incorrect for the next interactions.

This is how our system components function at run time phase for understanding the user request.

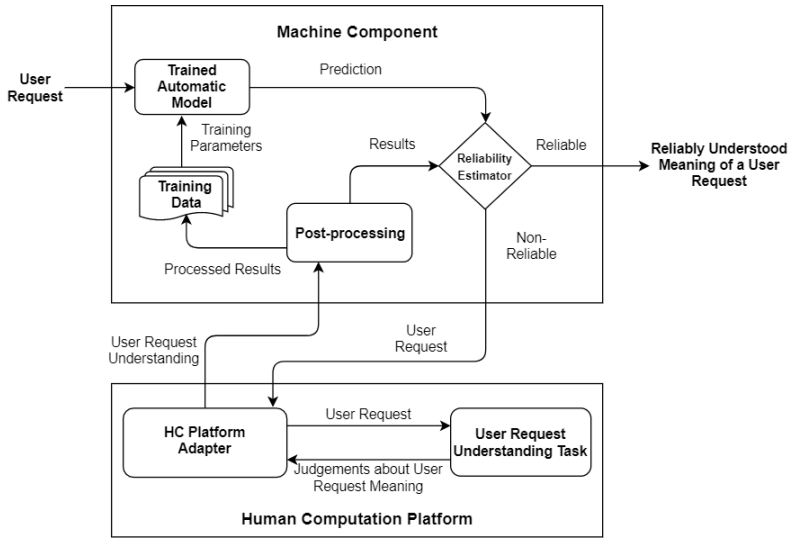


Figure 3.5: Run Time Phase of a Hybrid Chatbot System for Understanding User Requests

3.2 Implementation

As we are developing a chatbot having humans in the loop, in our thesis, we choose the domain as transportation. Then for the transportation domain we discuss the requirements that our chatbot should support in order to satisfy this use case and implementation part of RQ2. After this, we discuss the system components of our chatbot along with their interaction with each other at design phase and run time phase. We then discussed the architectural components of the system.

3.2.1 Use Case

We consider a hypothetical user named 'John'. John has a smart phone and he has widely used messaging application *Telegram* installed on his phone. Currently, he leaves in The Hague and often travels to Delft and sometimes to Rotterdam. He usually travels by train. But he also has a bike and wants to travel some time by bike. He needs to get an idea whether to go by bike or by train or rent a car and go by car. What kind of travel information would John be interested in? Such as when is the next train from The Hague to Delft? How to go to Rotterdam by bike? What time it takes to reach Delft by car? etc. It would be helpful for John if he asks these kinds of requests to a transportation chatbot on Telegram and gets the travel information in response. He can specify his travel requirements in natural language and can express them well.

Keeping this scenario in mind, we decided to implement a chatbot working in a *single domain* as a transportation. We chose this domain as there is a rise in chatbots in this area and we believe this is an interesting domain. Our chatbot is *informational* chatbot. The chatbot is providing transportation information to a user for a route from his/her departure place to arrival place, according to user's mode of transport. The user can request the chatbot including his place of departure (source), place of arrival (destination) and mode of transport in natural language. Chatbot understands these parameters in user's request and provides the required information to the user. It supports 3 modes of transportation as driving, bicycling and public transport such as train, tram, bus, metro whichever is available for a particular source-destination pair. We have *text* as a medium of communication between users and the chatbot. And users can communicate with chatbots with their *mobile phones* or web version of a messaging platform on *desktops*. Figure 3.6 is a use-case diagram of the transportation chatbot, showing the high-level view of functionalities.

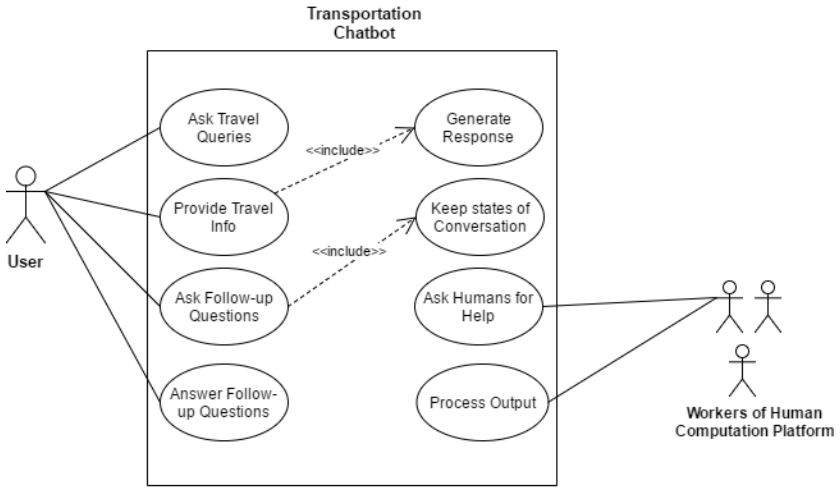


Figure 3.6: Use Case Diagram of the Transportation Chatbot

3.2.2 Requirements

We first identified requirements for the implementation of a chatbot having humans in the loop. Here we have two aspects as an implementation of a chatbot and involving human computation in our chatbot. We analyzed the requirements for each aspect separately.

Chatbot Related Requirements

Based on the scenario and our use case we discuss chatbot requirements of chatbot platform and functionalities that chatbot should support.

- **Messaging platform requirements**: We first identify best suitable messaging platform for our work and the functionalities that it provides. For this task, the survey about existing messaging platform helped us to choose the suitable messaging platform. From the survey we performed and presented in Table 2.2, we chose chatbot platform as Telegram as it actually satisfies our requirements. It is actually very easy to implement and it does not have to go through any review process which would limit us in implementing different functionalities. We chose our programming language as Python for the development of a chatbot.
- **Chatbot functionalities requirement**: We identified the functionality requirements of our chatbot in two categories such as identifying front-end requirements and identify ting back-end requirements.
  Front-end requirements:
  - Starting conversation with a chatbot as user initiates the conversation in our chatbot
  - Asking and saving user's home place and work place so as to simply further communication
  - Providing real-time response to user
  - Asking follow-up questions if something is missing in a request
  - Providing some extra information

  Back-end requirements:
  - Having an integration with Natural Language Understanding engine for processing and understanding user requests
  - Keeping the state of a conversation
  - Retrieving travel information from the from internal/external data sources
  - Asking humans for help for understanding user request
  - Generating a response with the pre-defined answer templates

Human Computation Related Requirements

Here we identified the requirements of human computation aspect in our chatbot. We need a help of humans to understand user requests in our chatbot when chatbot itself cannot understand them correctly. First, we identified CrowdFlower as the human computation platform on which we designed and hosted our tasks. We chose CrowdFlower as a human computation platform because, it is very easy to design and publish different tasks on it via API. CrowdFlower API is a HC platform adapter in our implementation.

3.2.3 System Components and phases

We have implemented two components in our system as per our system design in Section 3.1. First is machine component and second is human computation component for user request understanding in our system. We first describe them briefly and their interaction during design time and run time.

Machine Component

Our machine component for understanding user request has RASA NLU model as a trained automatic model, which is explained in detail in Subsection 3.2.4. This model predicts the intent and entities of the user request with a confidence value. This prediction is then passed to the decision component. Decision component has the confidence threshold level defined for this intent and entities prediction. We define a confidence threshold level. If the intent and entities are identified below this threshold level, machine component asks human computation component for a help in understanding user request. For implementing our system we first consider this threshold as '0.5' but later we will define this confidence threshold from reliability estimation experiment, described in Section 5.1.

In design phase, we train our RASA NLU model with a training data. This training data is collected from a data collection task on CrowdFlower. We first decided upon on our intents and we designed a task on CrowdFower to collect the training data related to these intents. We asked CrowdFlower workers to provide us examples of requests for each intent that our bot supports. This data collection task is a part of our high quality taining data collection experiment, which is explained in more detail in Section 4.1. After collecting this data, we checked its quality for possible repetitions and unreasonable examples. We removed them and trained our RASA NLU model with the training data. Please refer Figure 3.7 which shows design phase of our transportation bot.

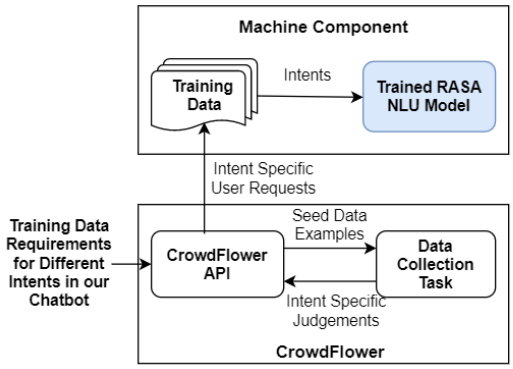


Figure 3.7: Design Phase of the Transportation Chatbot for Understanding User Requests

In run time phase, every user request is predicted by trained RASA NLU model. The prediction is evaluated based on confidence value by the decision component. And it then interacts with dialogue manager if it is above the threshold level. If it is below the threshold level, the request is forwarded to human computation component.

Human Computation Component

Human computation component of our system has 2 tasks: *data collection* used *at design time* which is already explained in Section 3.2.3; the second task is *user request understanding task* used *at run time*. User request understanding task is helping in understanding the intents and entities of a user request which is forwarded by the machine component. Workers of CrowdFlower perform this task and provide the insights of what user has asked or said. We ask workers to provide correct intent and entities in user request. We sent this output of a task to the post-processing component which is extracting intents and entities from the response JSON file in our case. We then forward these extracted intents and entities to decision component with confidence value set as **1**. Then decision component definitely passes them to dialogue manager for further actions. We incorporate the user request and

its intent into a training data set. We retrain our automatic model with this data to handle the similar type of requests for next iterations. Figure 3.8 which shows run time phase of our transportation bot for understanding user requests.

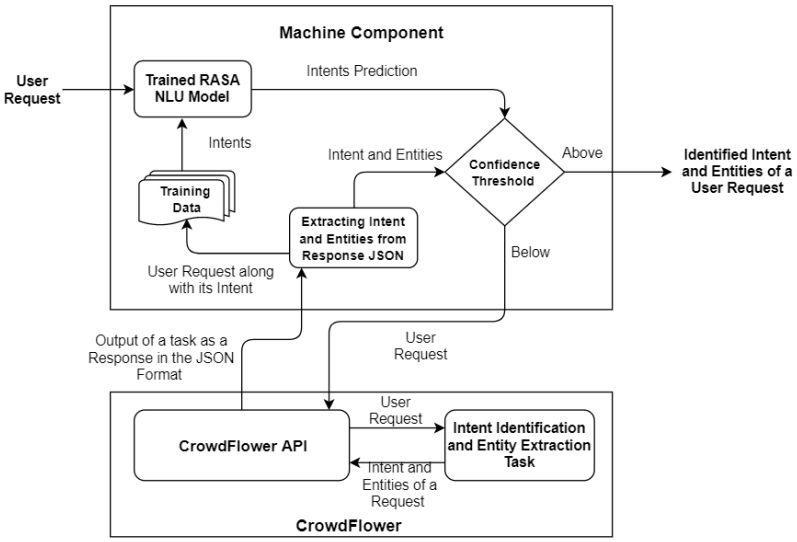


Figure 3.8: Run time Phase of the Transportation Chatbot for Understanding User Requests

3.2.4 Architectural Components

To implement our chatbot, we initially identified an architectural component requirement for our chatbot. As we are focusing more on natural language understanding in our thesis, we identified the system components for the same which we already explained in Subsection 3.2.3. We worked on identifying dialogue management and response generation strategies and then identified data sources that would help us in providing required travel information to the user. After identifying the requirements, we implemented these architectural components. Each architectural component is described below in a detailed manner. Among the all architectural components in our chatbot, the natural language understanding and dialogue management use both machine component and human computation component, while, rest all other architectural components are only machine components that is they are fully automatic. Figure 3.9 shows the transportation chatbot architecture. In our transportation chatbot, we involve human computation in Natural language understanding module and in dialogue manager.

Natural Language Understanding

We have two components in this architectural component such as RASA NLU which is automatic machine component and second is human computation component which is CrowdFlower task. Both are explained below in detail.

Understanding user requests with RASA NLU: *RASA NLU* is one of the popular open-source machine learning tools for intent classification and entity extraction. It has a set of functionalities for building our own language parser using existing natural language processing libraries such as *SpaCy* to clean up and parse the text, and machine learning libraries *scikit-learn* to build the models. Unlike other NLU tools such as API.ai and Wit.ai, RASA NLU doesn't require to make an *http* call to parse every request, it can work standalone and does not need internet connection. A RASA NLU model can be tuned well for

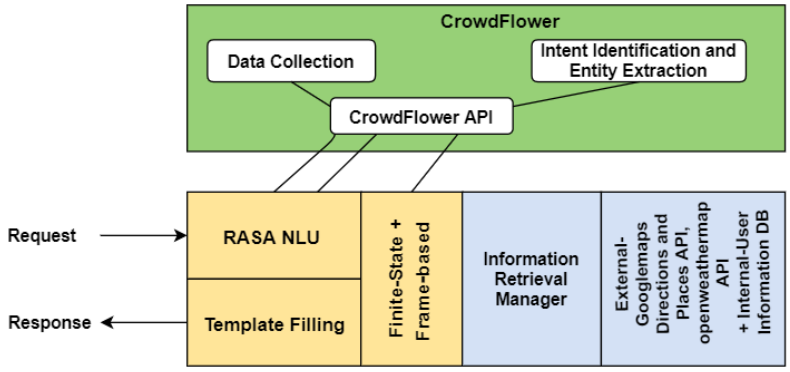


Figure 3.9: Architecture of the Transportation Chatbot

a particular use case. We trained our RASA model with the requests we collected from CrowdFlower data collection task at design time which is already explained in detail in Subsection 3.2.3. Every incoming request is checked against this model providing intent of a request with a confidence value and entities are extracted from the user request. RASA NLU provides a structured data from an unstructured user request. Figure 3.10 shows the example of producing structured data in JSON format from an unstructured user request. The intent prediction by RASA NLU model is then evaluated by decision component and the user requests below defined confidence threshold level are forwarded to CrowdFlower task.

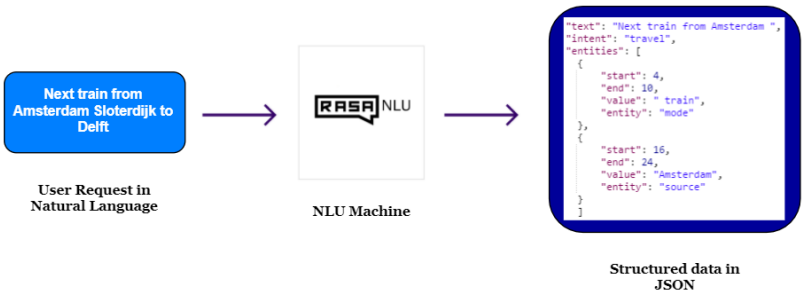


Figure 3.10: RASA NLU Producing Structured Data from a User Request

Understanding user requests with the help of CrowdFlower workers: Here we design a task on CrowdFlower for identifying correct intent and extracting the entities from the user requests. We send the user request and identified intent by RASA NLU model to this task via CrowdFlower API. We first ask CrowdFlower workers to verify if the identified intent is correct. If it is incorrectly identified we ask workers to provide correct intent of the user request from the list of intents we provided. If the intent is not in the list, we ask them to type the correct intent. This will help us to include the newly identified intents in our model by retraining the model. After intent, we ask workers to extract the entities from the user request. For each intent, we have some specified entities list that we ask workers to fill in. For example, for travel intent we ask workers to provide a source, destination and mode of transport from the user request. We also ask workers if there are some entities missing. After providing all this information, workers submit the task and natural language understanding component gets the response back via CrowdFlower API, in JSON format.

We then have an understanding of what user has asked to our chatbot. Screen shot of CrowdFlower entity extraction task can be found in Appendix A.3.1.

Dialogue Management

We use mixed strategy of dialogue management for our chatbot. We have used finite-state and frame-based dialogue management techniques which are already discussed in Section 2.1.4. For the very first time when a user interacts with a chatbot, we identify him/her as an *unknown user* and ask for providing his home place and work place to save in our database. We ask this information to the user in the form of follow-up questions, maintaining a state of a conversation with a frame-based approach. Afterwards, we implement two different strategies for dialogue management. For other intents than travel, we have finite-state dialogue management and for travel intent, we have frame-based dialogue management. For other intents, we keep the state of a conversation with finite numbers of steps. For travel intent, we ask follow-up questions we are missing some of the mode of transport, source or destination in the user request. We get the missing entities in particular user request from the 'Intent Identification and Entity Extraction' Task on CrowdFlower. This is how we involve human computation in dialogue management in our implementation. Once we know the missing entities, chatbot asks users the finite series of questions to fill these missing entities, keeping the state number. For example, if only the mode of transport is missing then we call it as a 'state number 1', if mode and source are missinf then we call it as a 'state number 2' etc. Based on the state number we ask follow-up questions to users. In our chatbot dialogue management is simple and straight-forward.

Data Sources

We have a user database to save user information such as user name, unique id on Telegram, when a user interacts with a chatbot for the first time, chatbot asks user his/her home place and work place, and saves it in a database. We use Postgres database to save this user information such as user's Telegram id, his/her home and work place and preferred mode of transportation. This is because a user does not need to mention his/her home place and work place in user requests again and again. This is our internal data source. For providing travel information to the user, we chose to retrieve real-time information from Googlemaps APIs such as *Googlemaps Directions API* and *Places API*. We also use *openweathermap* weather API to retrieve weather information.

Information Retrieval Manager

Information retrieval manager performs information retrieval task from external data sources. When we have the full information from the user that is the mode of transport, place of arrival(destination), place of departure(arrival) and time of departure(taken 'now' as defaults if not specified), we query *Google Maps Directions API* through an HTTP interface. Query to API is constructed as a URL string, using text strings or with latitude/longitude coordinates of source and destination, along with the unique API key. We receive a response in *JSON format* with different fields such as routes, geo-coded way-points, legs of the journey, transit mode etc. We extract fields which we want to provide to the user for every mode of transport. We also retrieve some extra information for different modes of transport. We have specified below which information we extract for which mode.

For driving mode, after extract information as distance from source to destination and travel time to reach there, we ask the user if wants to receive parking near his destination. If the user opts for this option we retrieve parking names and their map URLs from *Google Places API*.

For bicycling mode, we extract information as a distance from the source to destination and travel time to reach there. If the distance between source and destination is more than 15km, we ask the user if we can provide an alternate option of public transport. At the same time, we retrieve weather data for both the source and destination places of the user, with *openweathermap Weather API* query. If the weather conditions are not favorable for bicycling, like heavy rain or wind at that time, we ask the user if we can provide with alternate public transport. If user opts for this alternate option in both the cases as long distance or unfavorable weather, we provide public transport options to the user by querying *Googlemaps Direction API*.

For public transport, we extract travel distance, the total time to reach the destination, arrival time to destination and directions of the public from the source to the destination with different public transport modes like a bus, tram, train etc.

Response Generation

We designed some templates for response generation for each intent. This is because we have different types of response messages for different intents. We have three types of templates described in detail below:

Template based response generation: For travel intent, when we received the JSON data for the request from Googlemaps Directions API, we extract information from the JSON file and fill the extracted information in a response template. Our response message for travel requests changes with the mode of transport. For each mode, we first send a summary message filling *mode of transport*, source, *destination* by filling them into the summary template. Then we fill data based on the user's mode of transport. For driving mode, we fill *travel distance*, the *total time* to reach the destination in a template. An example of driving mode template is- *So, you want to travel by [Car] from [Breda] to [Eindhoven]. Distance to [Eindhoven] is [59.5 km] And it will take [41 mins] to reach there considering current traffic conditions*. Square brackets show the fields we filled from the retrieved data. For bicycling option, we have template same as a driving mode. For public transport mode *travel distance*, the *total time* to reach the destination, *arrival time* to the destination if the journey is started *now* and a list of *directions* and public transport types like a bus, train, tram etc. from the source to the destination. For the missing information, we designed response messages templates which are asked to the user via follow-up questions. Also for asking extra information like parking near destination for driving mode, providing public transport information if the distance to cover by bicycle is large and weather conditions are not favourable to ride a bike, we designed separate templates.

Fixed response generation: For all other intents except travel, we have foxed response generation. We sent a suitable pre-defined response to his/her request. e.g for user's message "Thank you", we send fixed response as "You are Welcome!"

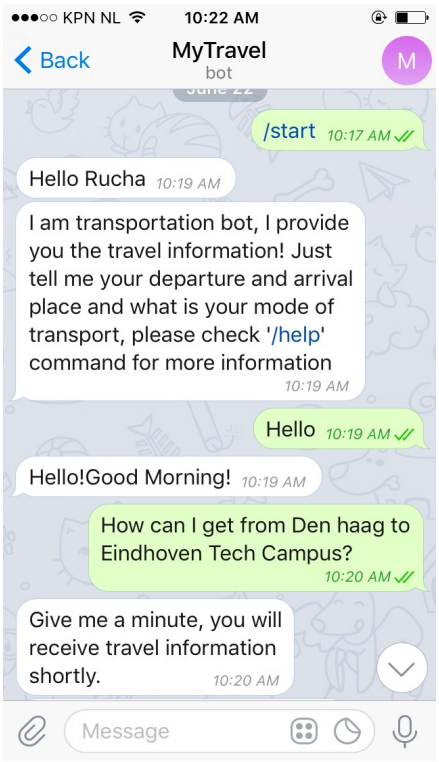
Figure 3.11 shows screen shots of our transportation chatbot, where an user request about travel navigation is shown.

3.3 Chapter Conclusion

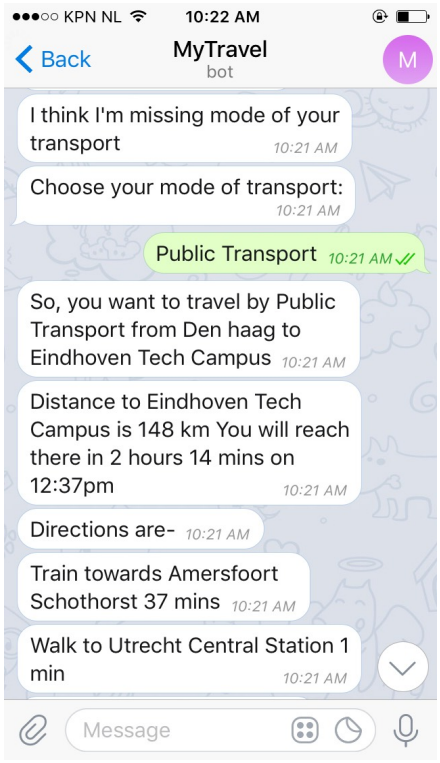
In this chapter, we first discussed the system design of a hybrid chatbot. We identified the components of a system as machine component, decision making component and human computation component. We designed these components and described their functionalities. We discussed how these components would co-ordinate with each other at a design phase and run time phase

In next section, we discussed the implementation of hybrid chatbot in transportation domain. We described how chatbot component requirements, inspired by the system design described in Section 3.1, are addressed by implemented functionalities. Furthermore, system components and architectural components of a chatbot were described, and the design choices are motivated.

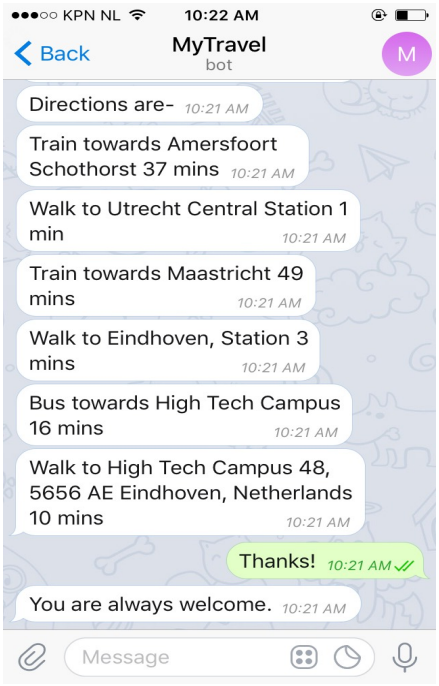
From our design and implementation, we presented that how a hybrid chatbot system would look like and how would it work.



(a) User started the conversation and asking for navigational information



(b) Chatbot asked crowd to extract entities from CrowdFlower, and then it asked follow up question that it needs the 'mode of transport' of the user



(c) Chatbot shows the public transport options from the source to the destination

Figure 3.11: Transportation ChatBot Screen Shots

Chapter 4

High Quality Training Data Collection with the Help of Human Computation

In this chapter, we design an approach towards the efficient involvement of humans in hybrid chatbot. We describe the approach and evaluate it by collecting training data for different domains and intents, in order to achieve the objective 3 of RQ3 that is to collect high quality training data for a chatbot using human computation.

In our hybrid chatbot system, we have automatic trained model for automatic understanding of user requests and human computation task to help this automatic trained model whenever it is not able to predict the meaning of user request reliably. We want to improve in both these areas, where our automatic model would be able to serve well by handling diverse user requests and it will go to the human computation component only when the user request prediction is not reliable. We consider these two areas separately: first, collection high quality training data to improve our automatic NLU model, which is discussed in this chapter and second, to design a reliability estimation strategy for identifying if the automatic prediction is reliable or not which is discussed in Chapter 5.

4.1 Approach

When NLU component for a new chatbot in any domain and purpose is designed with machine learning approach, then the NLU model should be first trained with training data related to the particular domain. This training data contains the representative requests that can be asked by the users. When we need to generate these representative requests from scratch, we only know the information need about the domains and intents. Based on this need, we aim to build an approach to collect the training data for training the NLU model of a chatbot, where we have no data present to train the NLU model initially. This training data should be diverse in nature to make chatbot capable of handling a variety of requests from the users at run time.

We want to make our trained NLU model capable of handling diverse user requests at run time, and for that our goal is to make it better by providing high quality training data at design time. Once we have our automatic NLU model trained with high quality data, we can expect that it will perform well and serve diverse user requests. With this goal, we aim to design a methodology and an experiment to collect high quality training data while cold-starting the chatbot. Figure 4.1 shows the research area of high quality data collection approach to address RQ3.

We aim to collect the training data with human computation component. We design a data collection task on human computation platform by asking human workers to contribute to the task, and we get the training data as an output of the task. Initially, we designed our

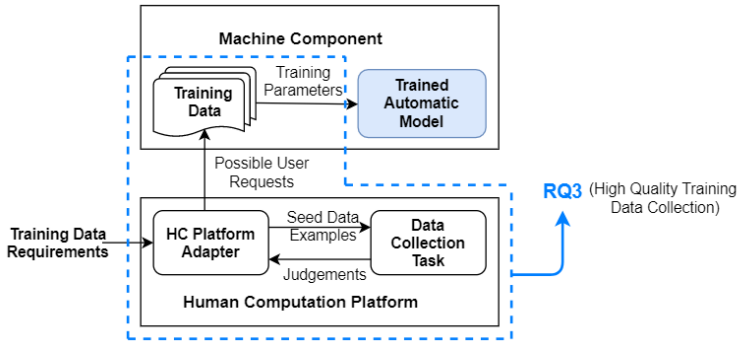


Figure 4.1: Our Research Question Area for High Quality Training Data Collection in a Hybrid Chatbot system

data collection task on human computation platform, by applying the best practices of data collection task [24, 36, 17, 61] in human computation. We ran our task on Crowdflower to collect the possible 'travel navigation' related requests for e.g. 'What are the ways to go to Amsterdam from Rotterdam?' and got the very low accuracy of the data we collected. Data we collected had the garbage requests having no meaning e.g. 'ADASD QWQWQW EYYÃÅP SULTAN CAMÃÂ°ÃÂ° TÃÅRBE MÃÅZE', irrelevant request to the travel navigation e.g. 'Are there many tourists here?', non-English requests and the requests those were not matching to our requirements. Then we improved our task design, by adding some validations to the requests. These validations were about ensuring the requests are meeting our requirements. We added validations such as English language validations by checking if the request is typed in English with checklanguage API , validations related to the presence of expected entities such as source(departure place), destination(arrival place) are also added. We iterated through this procedure improving the task design, every time and we managed to achieve 90% accuracy, we used this task design in our experiment and it is discussed in the next section in detail. With this iterative process, we designed our methodology to collect high quality data with the help of human computation. Figure 4.2 gives the higher level view of our data collection approach. We discuss each step in our approach and explain how we have followed these steps in setting up our high quality data collection experimental methodology.

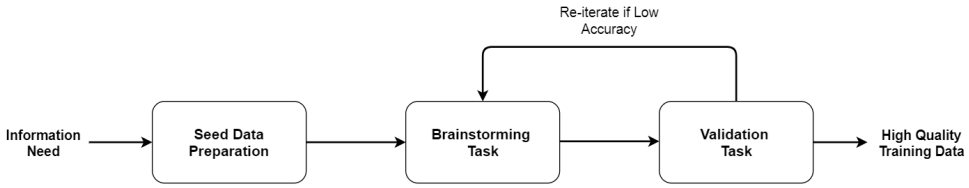


Figure 4.2: Higher Level View of Methodology Design of Collecting High Quality Training Data

Information Need

We first start with the understanding the information need. We identify the domain and intents in those domains, for which we want to collect data. We then identify how the diverse data for particular intent would be collected by identifying different entities associated with that particular domain. We identify the quantity of data we need to collect.

	Domain1-Travel	**Domain 2- Scheduling Meeting**	**Domain3- Software Development**
Intent 1- Informational	Asking for a Navigation Information Entities: Source, Destination, Mode of transport	Providing an information about the unavailability for the meeting Entities: Original meeting time, proposed time, place	Asking for a software development related information Entities: Programming language, Operating System, Package/library/tool
Intent 2- Transactional	Buying a travel ticket Entities: Source, Destination, Type/Purpose of travel, Date	Creating a meeting Entities: Participants, Time, Place, Duration	Taking software related actions(Install/Deploy/ Upload/Download) Entities: Action, Operating system, Size of Software
Intent 3- Transactional Change	Changing a travel ticket Entities: Source, Destination, Date	Modifying an already scheduled meeting Entities: Participants, Time, Place, Duration	Asking for a troubleshooting software related problems/ error Entities: Error/problem, Programming language, Operating System, Package/library/tool

Table 4.1: Domain, Intent and Entities Selection for High Quality Data Collection Experiment

To evaluate our approach, we decided to collect data for intents in 3 very different domains: first, 'travel' domain, as our chatbot works in this domain, Second 'scheduling meeting' domain, as this domain was discussed extensively in existing hybrid chatbot system 'calender.help' [11] and third 'software ' domain, as it is generally perceived that the general audience of human computation workers cannot contribute in such knowledge-intensive domain. We wanted to collect the data for both widely used domains and uncommon domains. After we decided on our domains, we decided the intent types for each domains. We chose 3 intents types as first 'informational', second 'transactional' and third 'transactional change'. Table 4.1 shows the selection of domains, intents and entities in detail. We chose the language for the requests as 'English'.

Seed Data Preparation

After we identify the need of data to be collected, we prepare the input data according to the need. This input data is prepared based on the domain and entities identified in previous step. We seed the input data to the 'Brainstorming task' on human computation platform. This input data provides the guidelines for performing brainstorming task.

For our experiment, we decided to run separate brainstorming tasks for each intent, and we started preparing input files for each tasks with seed examples. Based on number of the entities in each intent, we prepared 'Request Templates' for all possible combinations of entities present in the particular intent. Along with the request template, we put the expected order of entities to be present in the request with the 'Sequence Order' field. We also provided one example request per request template, to give an idea to the workers about our expectations. Table 4.2 shows the example of input file for the intent 'Navigation information' of travel domain. All other seed data input files screen shots can be found in Appendix A.2.

Request Template	**Sequence Order**	**Another Phrasing Example**
How to go to {YOUR DESTINATION} from {YOUR SOURCE} by {YOUR MODE OF TRANSPORT}?	First- Destination, Second- Source, Third- Mode of Transport	Traveling to {Boston Convention and Exhibition Center} from {Van Reypen St Jersey City}, which {train} should I take?
How to go to {YOUR DESTINATION} by {YOUR MODE OF TRANSPORT} from {SOURCE}?	First- Destination, Second- Mode of Transport, Third- Source	Traveling to {Boston Convention and Exhibition Center} , which {train} should I take from {Van Reypen St Jersey City}?
How to go from {YOUR SOURCE} to {YOUR DESTINATION} by {YOUR MODE OF TRANSPORT}?	First- Source, Second- Destination, Third- Mode of Transport	Traveling from {Van Reypen St Jersey City} to {Boston Convention and Exhibition Center}, which {train} should I take?
How to go from {YOUR SOURCE} by {YOUR MODE OF TRANSPORT} to {YOUR DESTINATION}?	First- Source, Second-Mode of Transport, Third- Destination	Traveling from {Van Reypen St Jersey City}, which {train} should I take to {Boston Convention and Exhibition Center}?
How to go by {YOUR MODE OF TRANSPORT} from {YOUR SOURCE} to {YOUR DESTINATION}?	First- Mode of Transport, Second- Source, Third- Destination	which {train} should I take, for traveling from {Van Reypen St Jersey City} to {Boston Convention and Exhibition Center}?
how to go by {YOUR MODE OF TRANSPORT} to {YOUR DESTINATION} from {YOUR SOURCE}?	First- Mode of Transport, Second- Destination, Third- Source	which {train} should I take, for traveling to {Boston Convention and Exhibition Center} from {Van Reypen St Jersey City} ?
How to go to {YOUR DESTINATION} from {YOUR SOURCE}?	First- Destination, Second- Source	Options to travel to {Boston Convention and Exhibition Center} from {Van Reypen St Jersey City}?
How to go to {YOUR DESTINATION} by {YOUR MODE OF TRANSPORT}?	First- Destination, Second- Mode of Transport	Options to travel to {Boston Convention and Exhibition Center} by {train}?
How to go from {YOUR SOURCE} to {YOUR DESTINATION}?	First- Source, Second- Destination	Options to travel from {Van Reypen St Jersey City} to {Boston Convention and Exhibition Center}?
How to go from {YOUR SOURCE} by {YOUR MODE OF TRANSPORT}?	First- Source, Second- Mode of Transport	Options to travel from {Van Reypen St Jersey City} by {train}?
How to go by {YOUR MODE OF TRANSPORT} from {YOUR SOURCE}?	First- Mode of Transport, Second- Source	Options to travel by {train} from {Van Reypen St Jersey City}?
How to go by {YOUR MODE OF TRANSPORT} routes to travel to {YOUR DESTINATION}?	First- Mode of Transport, Second- Destination	Options to travel by {train} to {Boston Convention and Exhibition Center}?
How to go to {YOUR DESTINATION}?	Required Travel Entity- Destination	How to travel to {Boston Convention and Exhibition Center}?
How to go from {YOUR SOURCE}?	Required Travel Entity- Source	How to travel from {Van Reypen St Jersey City}?
How to go by {YOUR MODE OF TRANSPORT}?	Required Travel Entity- Mode of Transport	How to travel by {bus}?

Table 4.2: Example of Input Seed Data for the Task- Navigation Information

Intent	Number of Entities	Number of Request Templates	Number of Requests per Request Template	Total Number of Requests
1.1 Travel- Navigation information	3	15	7	105
1.2 Travel- Buy ticket	4	64	4	256
1.3 Travel- Change ticket	3	15	7	105
2.1 Meeting- Expressing unavailability	3	15	7	105
2.2 Meeting- Creating meeting	4	64	4	256
2.3 Meeting- Modifying meeting	4	64	4	256
3.1 Software- Development information	3	15	7	105
3.2 Software- Perform action	3	15	7	105
3.3 Software- Troubleshooting problem/error	4	64	4	256

Table 4.3: Number of Input Seed Data Per Intent

Based on the combination of entities in each intent, we decided the number of requests we wanted to collect for each request template. Table 4.3 shows the details for number of request templates per intent, number of requests per requests template and the total number of request we expect to be collected per intent.

Brainstorming Task

Brainstorming task is the data collection task, where we ask workers on the human computation platform to provide the data by showing the guidelines in the form of seed example we prepared in previous step. These seed examples help workers to understand the exact need of the data to be provided. We design and launch the brainstorming task and collect the data once the task is finished.

For our experiment, we uploaded the input seed data to the task and designed the brainstorming task. We designed our task where we asked workers to look at the request template and provide the request matching to the template with different phrasing in given response field. We asked workers to imagine and provide the different way in which the request can be phrased for the particular request template. We also asked them to provide the entities in their request separately in individual response fields. Please refer Figure 4.3 showing the screen shot brainstorming task for navigation information intent. We added the validations for response fields to assure that we will get quality requests. The validations that we set for response fields were:

- Request should be in English language, no other language was allowed
- Request should have number of entities as expected in the request template and they are present in an expected order
- Entities provided separately in individual response fields are matching to those mentioned in the request itself

After adjusting task template, we launched the task. We launched each task for 3 times. So we launched 27 brainstorming tasks in total. We collected the data after all tasks got completed.

Validation Task

Validation task is to validate the data we collected in brainstorming task according to our requirements. We provide our validation criterias and data collected from brainstorming task

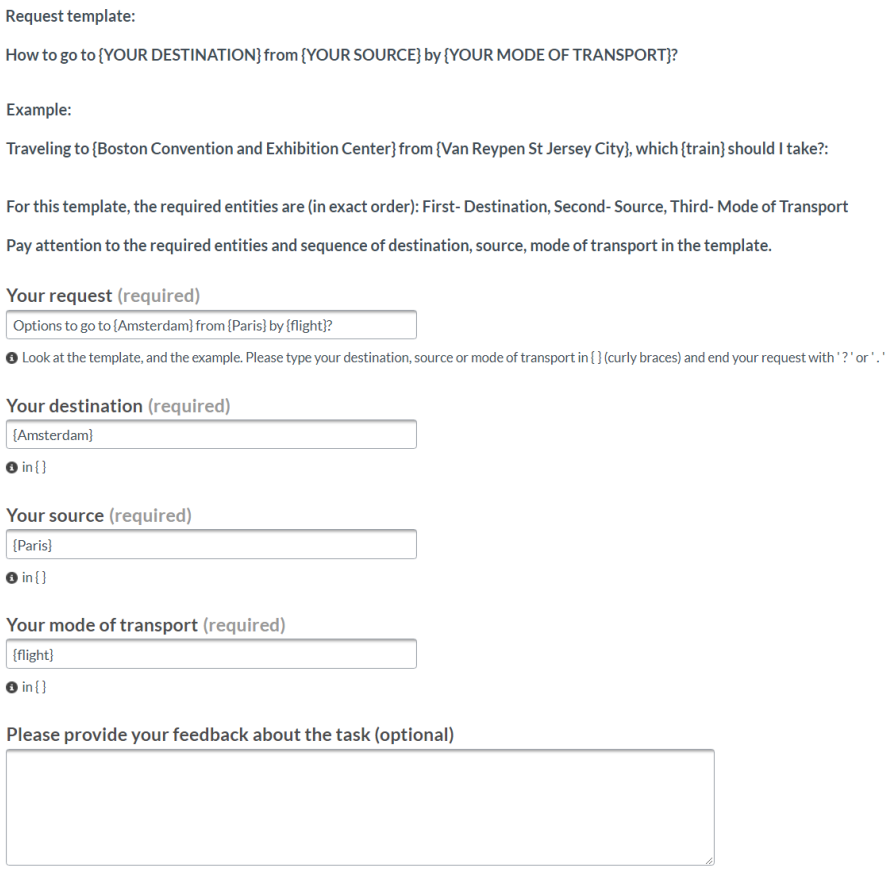


Figure 4.3: Screen shot of Brainstorming Task for Intent- Navigation Information

to the validation task, and ask workers to validate it. If we get a low accuracy, then we re-iterate through the process of brainstorming and validation till we get the high quality data with high accuracy for the required quantity already decided in the 'information need' step.

After collecting the requests from all 9 brainstorming tasks, we prepared the input files for the validation tasks with those collected requests. We designed the validation task where we provided workers the request and asked to provide their judgments about the request, based on the criterias we provided. The criaterias were same as the validations we set for the brainstorming task. We ask workers to mark the request correct if the request is satisfying all the criterias given. We designed task to get 3 judgments per request in each task. The reason for which collected 3 judgments per request as we wanted to consider a possibility to get different judgments for the same request, and we can observe the differences or agreements in the judgments for the same request. Please refer Figure 4.4 showing the screen shot validation task for navigation information intent.

High Quality Training Data

The output of the process is high quality training data set, the required number of high quality use requests examples are generated and those can be used to train the chatbot NLU model.

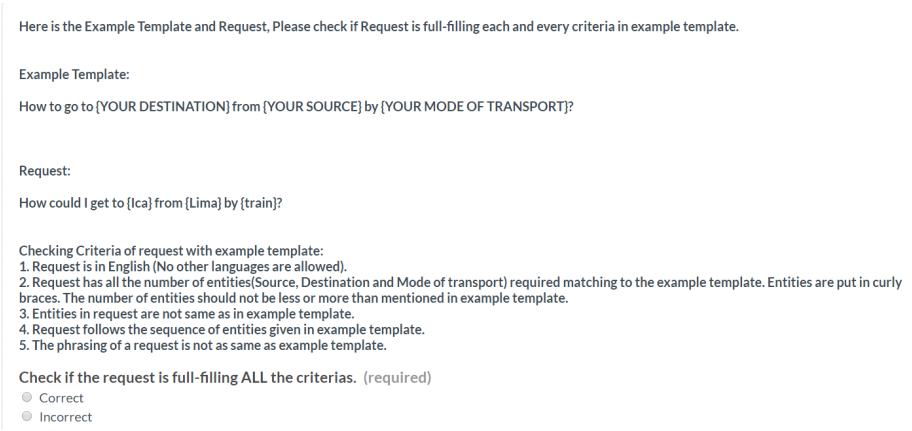


Figure 4.4: Screen shot of Validation Task for Intent- Navigation Information

4.2 Evaluation

In this section we first discuss the evaluation metrics and then discuss the results.

4.2.1 Evaluation Metrics

To evaluate our methodology, we set evaluation metrics for different steps in our methodology. We have 3 evaluation metrics in total; first is accuracy of brainstorming task, Second is accuracy of validation task, before we consider the agreement of the judgments and third is brainstorming + validation task accuracy after we consider the agreement of judgments.

1. **Accuracy of brainstorming task**
   We manually check the requests collected from brainstorming task and mark the requests correct if they are matching all the criterias we set up for the request to be correct. We calculate the accuracy of the brainstorming task as number of correct requests divided by the total number of requests collected from brainstorming task. We calculate the accuracy for each brainstorming task for all 9 intents. Following formula is showing the calculation of brainstorming task accuracy:

   $$A_b = \frac{N_{correct}}{Total_{requests}}$$

   where, A_b is brainstorming task accuracy, $N_{correct}$ is total number of correct requests and $Total_{requests}$ is total number of requests.

2. **Accuracy of validation task**
   Here we calculate the accuracy before considering agreement of judgments. We calculate this accuracy with number of matching judgments from validation task to the judgments of manual evaluation divided by the total number of judgments we collected from validation task. We calculate the accuracy for each validation task for all 9 intents. Following formula is showing the calculation of validation task accuracy:

   $$A_v = \frac{N_{matching}}{Total_{judgments}}$$

   where, A_v is validation task accuracy, $N_{matching}$ is number of matching judgments with manual evaluation and $Total_{judgments}$ is number of total judgments in a validation task.

3. **Accuracy of brainstorming + validation task**
   This accuracy would give us the final accuracy of brainstorming plus validation task. We calculate this accuracy after considering agreements of judgments for the requests from the validation task. We consider 3 strategies for agreement of judgments from validation task and calculate the accuracy for each of them for the tasks of all 9 intents.

   a) **Strategy 1 - At-least 1 out of 3 is correct**: In this strategy, we consider request as correct if at-least 1 out of 3 judgments for the request is marked as correct by the workers in the validation task.

   b) **Strategy 2- At least 2 out of 3 are correct** : In this strategy, we consider the request as correct if at-least 2 out of 3 judgments for the requests are marked as correct by the workers in the validation task.

   c) **Strategy 3- All 3 out of 3 are correct** : In this strategy, we consider request as correct only if all 3 out of 3 judgments are marked as correct by the workers in the validation task.

   The accuracy of brainstorming + validation task is calculated as the number of requests which are actually correct out of the requests validated as correct divided by the total number of requests validated as correct. We calculate this accuracy for all 3 strategies for all the tasks. Following formula is showing the calculation of brainstorming + validation task accuracy:

$$A_{final} = \frac{N_{actualCorrect}}{Total_{correct}}$$

   where, A_{final} is final accuracy of brainstorming+ validation task, $N_{actualCorrect}$is a number of requests which are actually correct out of the requests validated as correct and $Total_{correct}$ total number of requests validated as correct.

   The aim is to get only the requests which are identified as correct in these 3 strategies, and we want to re-iterate the brainstorming and validation task process for the requests which were identified as incorrect by these strategies.

These accuracy values would give us insights about the quality of user requests we collected as a training data for our chatbot. If the accuracy is low, our methodology recommends to re-iterate through the brainstorming task and validation task, for the incorrect requests, till we get the high accuracy. Results of this experiment are explained in next section in detail.

4.2.2 Results

Figure 4.5 shows the percentages of requests validated as corrects or incorrects for different strategies for different intents. We can clearly see in the figure that as we move from strategy 1 to strategy 3, number of requests identified as correct decreases and number of requests identified as incorrect increases. This is because our strategy 1 is an optimistic approach, where we loosely consider the agreement of judgments, thus we have more number of requests which are identified as correct by validation task. While the number drops as we consider majority for agreements and number even more drops as we consider strategy 3, where we strictly consider request correct, if all the 3 judgments for the request are correct. On the other hand, number of requests identified as incorrect, grows as we move from strategy 1 to strategy 3. With our methodology, we re-iterate for the process of brainstorming + validation for the number of requests which are identified as incorrect.

We then summarize the results of our data collection experiment in Table 4.4. The table shows the different accuracies for brainstorming task, validation task and brainstorming

+ validation task. The details of calculating different accuracies are already discussed in Section 4.2.1. We calculate standard deviation in accuracy for each task across all 3 runs and those standard deviation for brainstorming task and validation task are mentioned in the table. Table 4.4 contains the aggregated results of all 3 runs we conducted for the both brainstorming and validation tasks. We also calculate the standard deviation in accuracies of brainstorming task and validation task across all intents. We then provide accuracies for different agreement strategies and cost per request in each agreement strategy. We calculate cost per request as:
Cost per request = (Total cost of brainstorming task + validation task)/(number of requests validated as correct). The cost is in US Dollars(USD).

Brainstorming Task

We calculate the accuracy of brainstorming task based on our manual evaluation of requests for each task. As we can see in the table, for the brainstorming task of domain 'Travel', we could achieve the accuracy for 'navigation information' intent and 'buy ticket' intent more than 90%, while the accuracy for 'change ticket' intent was lower than other 2 intents. For the 'Meeting' domain, 'create meeting' intent we could achieve 98.82% accuracy while for other 2 intents it is lower than the create meeting intent. For the 'Software domain', 'Perform action' intent has the higher accuracy than two other intents in software domain. The accuracies for informational and transcational types of intents were higher than the transactional change intents. The average accuracy we achieved for all the tasks is 88.66% with a standard deviation of 5.72 for all the accuracies of brainstorming task.

Validation Task

For the validation task, we calculated the accuracy based on the number of matching judgments to the manual evaluation of requests. This accuracy is calculated before we consider the agreement of judgments. The standard deviation for transactional change intent in travel domain and meeting domain are comparatively more than other 2 types of intents in these domain, while it is minimum is software domain. We can see that the average accuracy for all validation tasks across all intents is 90.52%, and each separate validation task could achieve the higher accuracy in validating the requests collected from brainstorming tasks. The standard deviation is 4.17 for all the accuracies of validation task.

Brainstorming + Validation Task

We now calculate the accuracy for brainstorming + validation task considering the agreements of the judgments from the validation task. We call this accuracy as final accuracy because, in this stage we get the total final number of requests which are of high quality and can be used in a chatbot. We calculate this accuracy as total number of requests which are actually correct from the requests validated as correct divided by total number of requests validated as correct. Hence this accuracy gives us insights about the percentage of high quality requests we managed to get from the brainstorming task and those are validated by validation task.

We observed that the accuracy got improved as we move from strategy 1 to strategy 3. For the travel navigation intent, for the strategy 1, where we loosely consider the agreement of judgments, we have achieved 97.57% accuracy. If we focus on strategy 2, where we consider majority in agreement, we have achieved 98.92% accuracy and in strategy 3, where we strictly consider the requests we can achieve up to 99.58% accuracy. We could already achieve 97.57% accuracy for navigation information intent and even higher for

Intent	Brainstorming Task- Data Collection		Validation Task- Before Agreement		Agreement Strategy 1		Agreement Strategy 2		Agreement Strategy 3	
	Avg. Accuracy 3 Runs	**SD across 3 Runs**	**Avg. Accuracy 3 Runs**	**SD across 3 Runs**	**Avg. Accuracy 3 Runs**	**Avg. Cost per Request**	**Avg. Accuracy 3 Runs**	**Avg. Cost per Request**	**Avg. Accuracy 3 Runs**	**Avg. Cost per Request**
1.1 Travel- Navigation information	90.15%	3.84	92.48%	0.66	97.57%	0.073$	98.92%	0.075$	99.58%	0.087$
1.2 Travel- Buy ticket	94.39%	5.38	93.23%	2.82	97.14%	0.066$	98.70%	0.068$	99.63%	0.075$
1.3 Travel- Change ticket	79.36%	1.24	85.39%	7.15	86.53%	0.072$	90.31%	0.078$	94.87%	0.095$
2.1 Meeting- Unavailability information	88.25%	1.45	87.29%	3.22	91.47%	0.067$	96.46%	0.073$	97.78%	0.090$
2.2 Meeting- Create meeting	98.82%	0.78	97.73%	1.24	99.47%	0.063$	99.73%	0.064$	99.85%	0.067$
2.3 Meeting- Modify meeting	81.85%	8.33	92.93%	7.78	92.46%	0.074$	95.87%	0.080$	99.20%	0.096$
3.1 Software- Development information	89.83%	3.96	89.09%	3.99	93.71%	0.0674$	97.09%	0.072$	98.21%	0.089$
3.2 Software- Perform action	90.79%	2.19	91.53%	5.32	94.17%	0.066$	96.49%	0.070$	97.16%	0.079$
3.3 Software- Troubleshooting problem/error	84.50%	5.33	84.98%	2.49	89.59%	0.072$	93.72%	0.077$	97.63%	0.10$
	Mean= 88.66%		Mean= 90.52%		Mean= 93.57%		Mean= 96.37%		Mean= 98.21%	
	SD= 5.72		SD= 4.17		SD= 4.10		SD= 2.91		SD= 1.59	

Table 4.4: Results- Data Collection Experiment

create meeting intent 99.47%, by loosely considering the agreements of judgments. While troubleshooting problem/error intent in software domain achieved 89.59% with strategy 1 which is lower, and strategy 3 could achieve 97.63% accuracy. The mean accuracy for agreement strategy 1, is 93.57% and the standard deviation is 4.10, mean accuracy for strategy 2 is 96.37% and standard deviation is 2.91, which is smaller than standard deviation in strategy 1. Strategy 3 has mean accuracy as 98.21% which is higher than other two strategies and it has very small standard deviation 1.59. This shows that all the tasks could consistently achieve higher accuracy with strategy 3. The cost per request grows from strategy 1 to strategy 3 for each task. Cost per request in strategy 1 is cheaper than strategy 2 and strategy 3.

4.3 Discussion

The accuracies for brainstorming task and validation task vary for different type of intents. The accuracies of brainstorming and validation tasks for informational intents and transactional intents are higher than the transactional change intents. We observed that the workers often got confused between transactional and transactional change intents. For example, we got some requests of buying ticket for change ticket intent in travel domain and some requests of creating meeting intent for modifying meeting intent in meeting domain. This made the requests in change ticket and modify meeting semantically incorrect, which were not conveying the actual meaning of the intents. Hence the accuracy of for change ticket and modify meeting got dropped in brainstorming task. We observed that, even for the validation task, they got confused while validating the requests in modify meeting and change ticket intents.

Accuracies for travel and meeting domain are comparatively higher than software domain. Workers can provide the requests easily in widely used domains, and not in uncommon domains like software, where they lack the knowledge of the domain. For the software domain, workers could not provide correct requests regarding software entities such as programming language and package or libraries used. Some of the requests were vague in this domain, and hence the brainstorming tasks accuracies for software domain are comparatively lower than travel and meeting domain. We assumed that the workers would face difficulties in providing requests in knowledge-intensive domains such as software domain, and we observed that workers did not perform well for this domain.

Standard deviation for both brainstorming tasks and validation tasks are very small and show that our approach could collect requests consistently across all the intents for 3 runs. We could achieve diversity in user requests by designing request templates. We have received positive feedback from the workers that the brainstorming tasks were interesting, nicely designed and seed example provided useful guidelines to brainstorm the requests with different phrasings. It could also be useful to ask workers only for different phrasings but not for the entities, and entities could be programmatically generated as some workers found it difficult to follow the order of entities while providing the requests. The task design of brainstorming task could be improved based on the workers' feedback and more diversity could be introduced. We could also design an approach such as Games with a Purpose (GWAP) [74], where we can ask one worker to provide a request, and next worker should provide the phrasing which is different from he previous requests. This will help in achieving more diversity where no request would have the same phrasing. But, this task would become more difficult for workers, and may be we need to pay more, choose expert workers and improve the design of the task.

The intents, for which we have achieved higher accuracy with agreement strategy 1, we can use strategy 1 for those intents as strategy 1 can produce more number of requests

in lower cost. We can save the cost of re-iterating the brainstorming and validation task as we have less number of requests identified as incorrect in strategy 1. If the accuracy is considerably low with the strategy 1, we need to consider strategy 2 or strategy 3 to have higher accuracy, but with more cost for each requests. To reduce the cost that we need to pay for strategy 3, and to improve accuracy than strategy 1, we can consider agreement strategy 2, where we consider the majority of validation judgments. Designing different strategies and analyzing the accuracy versus cost trade off, would be helpful to choose the appropriate strategy for different tasks. We have observed from our results that some intent such as navigation information, buy ticket and meeting could already achieved higher accuracies with strategy 1 and with the low cost per request. While intents like troubleshooting problem/error, change meeting should consider strategy 3 to achieve higher accuracies, but the total cost also increases as the cost per requests in these intents is more.

With our data collection approach, we managed to collect chatbot requests in 3 different domains for 9 intents with high accuracies. We designed the agreement strategies for validation judgments and we discussed accuracy versus cost trade off for these 3 strategies. We successfully collected 4158 high quality requests out of 4647 total requests that we requested with different brainstorming tasks.

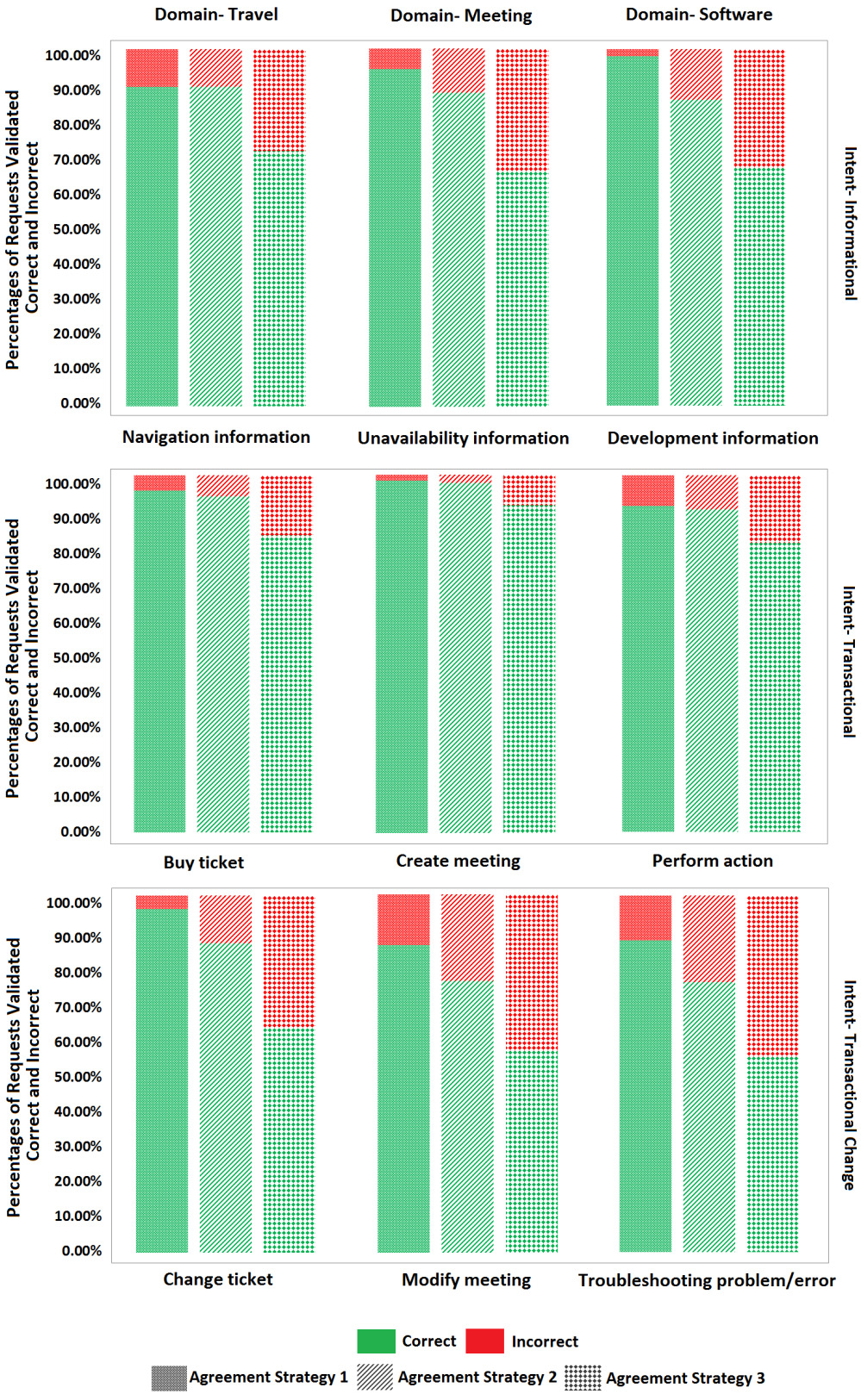


Figure 4.5: Percentages of Requests Validated as Correct and Incorrect For 3 Different Validation Agreement Strategies for all Domains and Intents

Chapter 5

Reliability Estimation of Intent Classification by Automatic Trained NLU model

In this chapter, we describe an approach and experimental methodology designed and applied in order to achieve the objective 4 of RQ4 which is to determine a confidence threshold level, below which the automatic understanding of a user request is considered as non-reliable. This is the next part of efficient involvement of humans in hybrid chatbots.

5.1 Approach

After collecting high quality training data in data collection experiment, we train NLU model with that data and the model predicts the intent of a user request with a confidence value. By using this confidence value we aim to design an approach to identify a confidence threshold above which, the prediction by automatic NLU is considered as reliable, otherwise as non-reliable and is sent to the human computation component. Thus we design and set up an experiment focusing on estimating the reliability of intent classification of user requests by the trained automatic NLU model. We also discuss how to account for parameters such as different data set sizes and types of intents to train the NLU model. And we study how these parameters affect the intent classification of NLU model. Figure 5.1 shows the research area of reliability estimation approach to address RQ4.

5.1.1 Setup

In this section we describe the set up of the reliability estimation experiment.

We start with training and test data Selection. We first selected the training data that should be used to train the automatic NLU model and test data to test the intent classification of this model. We selected our training and test data from the high quality data we collected from the data collection experiment.

In the data collection experiment, we ran brainstorming tasks to collect total 4647 user requests for a chatbot for 9 intents explained in Table 4.1. Out of those 4647, 4158 requests were identified as high quality requests by our evaluation of data collection experiment explained in Section 4.2.1.

So we considered 4158 high quality requests for this experiment. We divided it into 2 data sets as 90% training data set and 10% test data set. Thus, our test data set was of 416 user requests. We made sure that the test data is balanced by considering, same number of

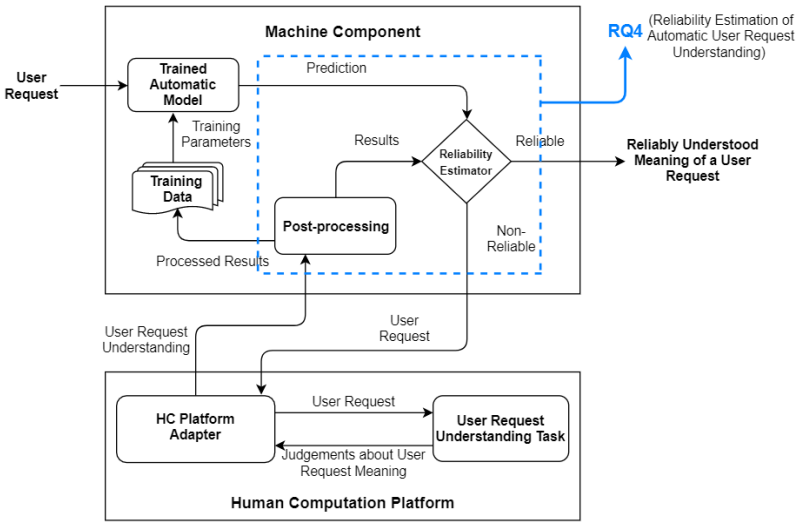


Figure 5.1: Our Research Question Area for Estimating the Reliability of User Request Prediction by the Trained Automatic NLU model

Training Data Set Size	Number of Folds
15	5
30	5
59	5
117	5
234	5
468	5
936	3
1871	2
2700	1
3742	1

Table 5.1: Training Data Set ups for Intent Classification

requests for all the 9 intents. And each intent requests have all possible combinations of different entities sequence present in them. From the total training data set of 3742 requests, we started by considering the lowest first as 15 and each time we doubled the training data set size, till we reach the maximum as 3742 requests. Table 5.1 shows the different training data set ups and the number of folds we used in our experiment.

For the each training data setups, we made sure that the data set is balanced with equal number of requests of all 9 intents with the all possible combination of entities. We used RASA NLU as our trained automatic NLU model. And we trained different RASA NLU models for different training data set ups having the requests associated with their intents. So for each setup, we had 9 different classes corresponding 9 different intents.

5.2 Evaluation

In this section we first discuss the evaluation metrics and then discuss the results.

5.2.1 Evaluation Metrics

Once we train the RASA NLU models, we test all of them for the same test data. RASA NLU models predict the intent for each request in a test data and provide confidence value ranging from 0 to 1 for the intent prediction for each request. We want to find out which confidence value should be chosen to decide whether the prediction is reliable or not. We draw intent classification matrix for each training data set up. Intent classification matrix has different confidence values from 0.1 to 0.9, and we call them as confidence thresholds. Then we calculate 'corrects', 'incorrects' and 'unknowns' for each confidence threshold in following way.

1. **Corrects**: Number of requests identified correctly with a confidence greater than a given confidence threshold
2. **Incorrects**: Number of requests identified incorrectly with a confidence greater than a given confidence threshold
3. **Unknowns**: Number of requests identified with a confidence below than the given confidence threshold

We consider a number of unknown requests as non-reliable intent classification by the RASA NLU model and we need the help of human computation for those requests. We consider the prediction of corrects and incorrects as reliable intent classification as those are the requests which are predicted correct and incorrect respectively, with the confidence higher than the particular confidence threshold. We observe and analyze the number of corrects, incorrects and unknowns for the different setups having different training data sizes. After the analysis of percentage of corrects, incorrects and unknowns for different data set sizes with increase in confidence thresholds, we will analyze how to pick an optimal confidence threshold for different setups. To pick up an optimal confidence threshold, we will mainly focus on number of incorrects and unknowns as incorrectly identified requests can produce incorrect responses, and for unknown requests, chatbot does not have a response to send to the user. We want to calculate penalties for incorrects and unknowns in each set up. For this, we assign a cost for them in 3 strategies:

1. **Incorrects are more expensive than unknowns**: We assign a cost for incorrect as 5 and unknown as 1.
2. **Unknowns are more expensive than incorrects**: We assign a cost for unknown as 5 and incorrect as 1.
3. **Both are expensive**: We assign a cost for both as 5.

We will then calculate the cost for all 3 strategies for the number of incorrects and unknowns in all training data set setups. We calculate the cost as:

Cost= Cost of incorrect(number of incorrects) + cost of unknown(number of unknowns)

Another goal of our experiment is to analyze if the type of intent affects the intent classification. For that, we separate out the intent-wise requests from test data set and we then prepared 9 different test data set including requests of 1 intent in 1 data set. We keep the same training data setups and we test these 9 test data sets against the same trained RASA NLU models in previous setup. We analyze the number of corrects, incorrects and unknowns by drawing the classification matrix for each setup.

With these setups, we want to analyze what is the effect of different training data set sizes on the intent classification of a trained NLU model and also how the number of corrects, incorrects and unknowns differ for different types of intents for different confidence thresholds. From this analysis, we aim to find out the best suitable threshold value for different set ups which we discuss in next section in detail. And this confidence threshold value can be used as a reliability estimator of a intent classification by trained NLU model.

5.2.2 Results

We first draw the classification matrix for all setups for different training data set sizes which is already discussed in Section 5.1.1. All the classification matrices can be found in Appendix A.1. Figure 5.2 shows the intent classification results. Percentage of corrects, incorrects and unknowns in each setup for thresholds 0.1 to 0.9 are represented in a stacked bar graph for each data set size.

We can see in the Figure 5.2 that the overall percentage of corrects increases significantly as we increase the training set size, while the overall percentage for unknowns and incorrects drops significantly with increase in training data set size. For smaller data sets, percentage of corrects for individual set up drops significantly with the increase in confidence threshold from 0.1 to 0.9, and it drops gradually for larger data sets. Percentage of incorrects for individual setups drops significantly for smaller data set sizes with increase in confidence threshold 0.1 to 0.9, where it drops gradually for larger data set size. Percentage of unknowns for individual setups grows significantly with the increase in confidence threshold for smaller data set size, while it grows gradually for larger data set size.

Figure 5.3 shows the cost vs confidence threshold value for each training data set. The possible optimal threshold is shown for individual set ups. We select the minimum value of threshold for all the strategies. Optimal threshold values for strategy 1 are marked with red ticks, strategy 2 are marked with green ticks for strategy 3, they are marked with blue ticks. We observe that the minimum threshold value for strategy 2, is always 0.1 for all set ups.

Figure 5.4 shows the trend of optimal thresholds for different strategies, for all data set sizes. The optimal threshold values remain same for lower training data set sizes when incorrects are more expensive, and they start increasing after training set size increases from 117 to 234 and so on. The thresholds at higher training data set sizes, for 1871, 2700 and 3742, remain constant. When unknowns are more expensive, confidence thresholds are same for all data set sizes. For strategy 3, where incorrects and unknowns both are expensive, confidence thresholds remain constant till data set size 117, it grows for data set size 234, and again remains constant till data set size 1871. The optimal threshold grows for data set size 2700 and 3742.

We now analyze how the intent classification works for different domains and intents separately. After we tested each intent separately against the RASA NLU models for all data set size set ups, we draw the classification matrix and observe the number of corrects, incorrects and unknowns. Figures 5.5, 5.6 and 5.7 represent the percentages of corrects, incorrects and unknowns for data set sizes small- 30, medium- 468 and large- 3742 for separate intents for different confidence thresholds. We analyze type of intents informational, transactional and transactional change for the 3 domains.

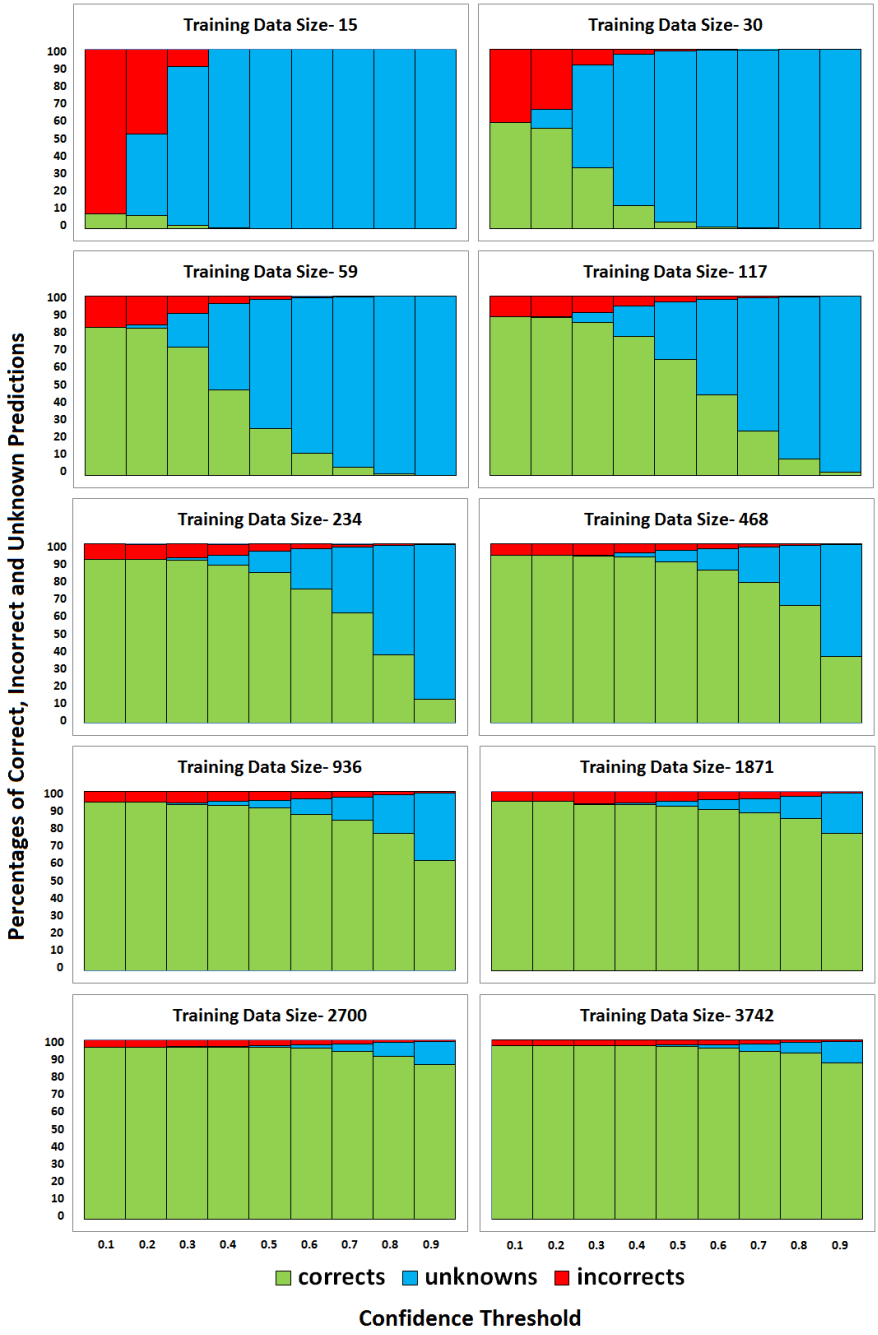


Figure 5.2: Intent Classification Results per Training Data Set Sizes

Small training data set

We observe that travel domain has more number of corrects than the meeting and software domain. Intents in travel domain have less number of incorrects where meeting domain has more number of incorrects and software domain has even more number of incorrects. This shows that the travel domain was more distinguishable than other 2 domains. Total number of incorrects are more in transactional change intent for travel and meeting domain. Number of unknowns are quite high for all intents in all domains. Each intent has different

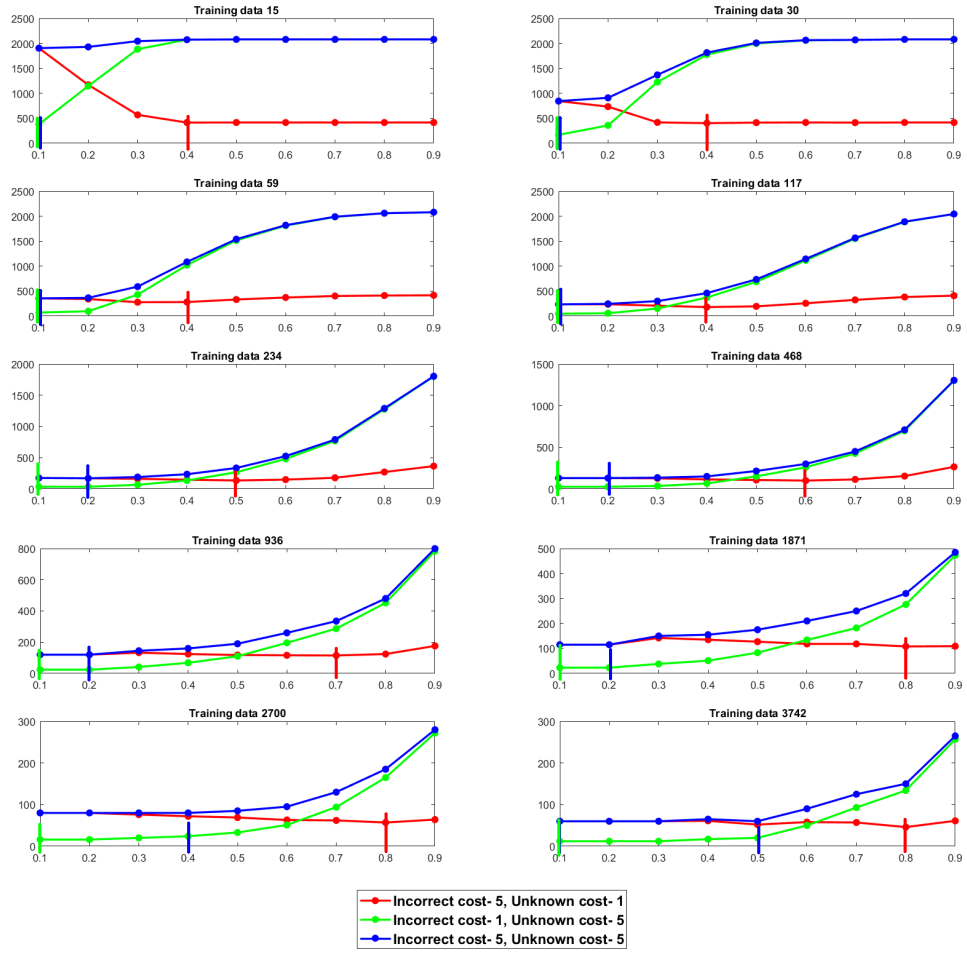


Figure 5.3: Total Cost vs Confidence Thresholds per Training Data Set Sizes

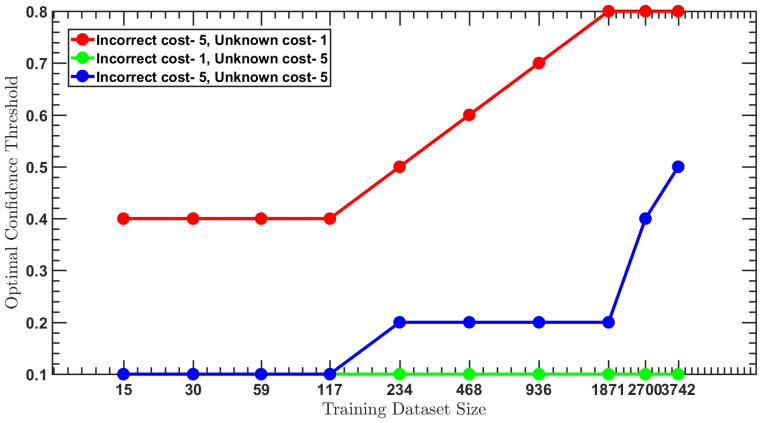


Figure 5.4: Optimal Thresholds Trends for Different Cost Strategies per Training Data Set Sizes

results for the same threshold value. for example if we want to choose optimal confidence

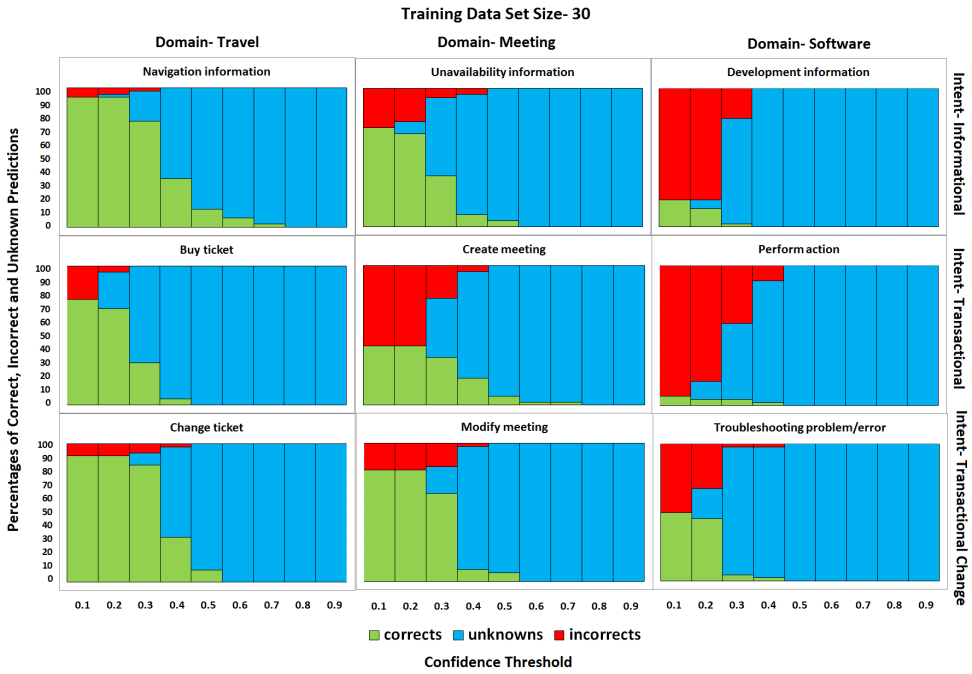


Figure 5.5: Specific Intent Classification Analysis for All Domain-Intent Pairs for Small Data Set Size

threshold suggested by Figure 5.4, considering the incorrects are more expensive, we will choose 0.4 as optimal threshold for data set size 30. The number of corrects, incorrects and unknowns are different for different intents, and they represent significant change.

Medium training data set

Number of incorrects and unknowns are significantly dropped for medium data set size. but the trend of having more incorrects for transactional change intent is same for the medium data set. When we choose an optimal threshold as 0.6 according to the Figure 5.4 considering incorrects are more expensive, we observe that the number of corrects does not change significantly.

Large training data set

For large training data set, NLU model performed well in classification for all the intents. Number of corrects are very high even if for high confidence thresholds. By choosing optimal confidence threshold as 0.8 according to the Figure 5.4 considering incorrects are more expensive, we observe that the informational intents do not have incorrects at threshold 0.8. Transcational intents for travel and meeting also do not contain incorrects in them. Transactional change intents of travel and software domain contain incorrects in them and the requests in those intents can be analyzed and improved to achieve better classification results.

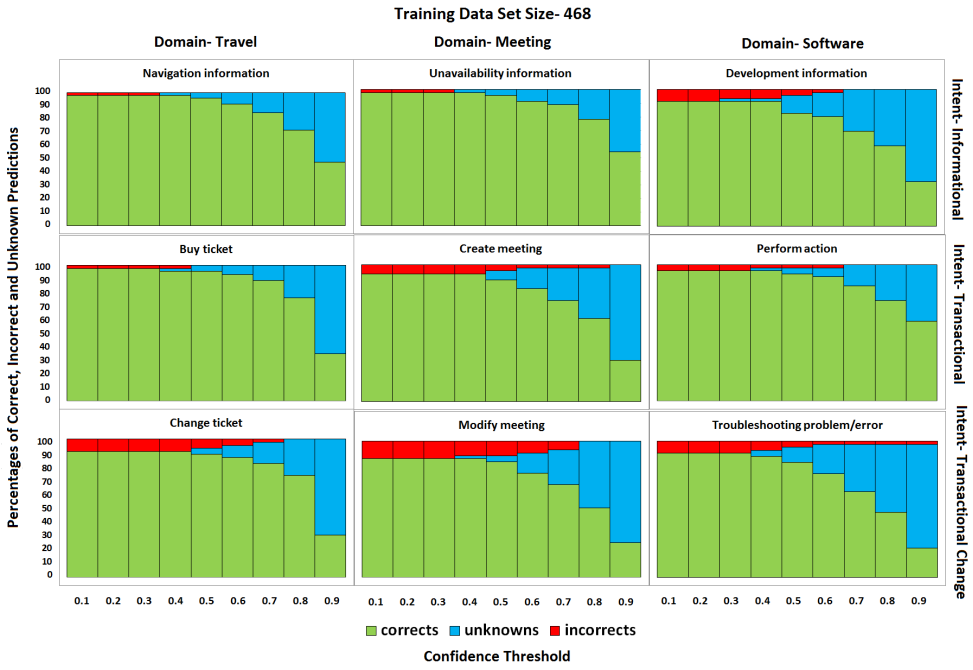


Figure 5.6: Specific Intent Classification Analysis for All Domain-Intent Pairs for Medium Data Set Size

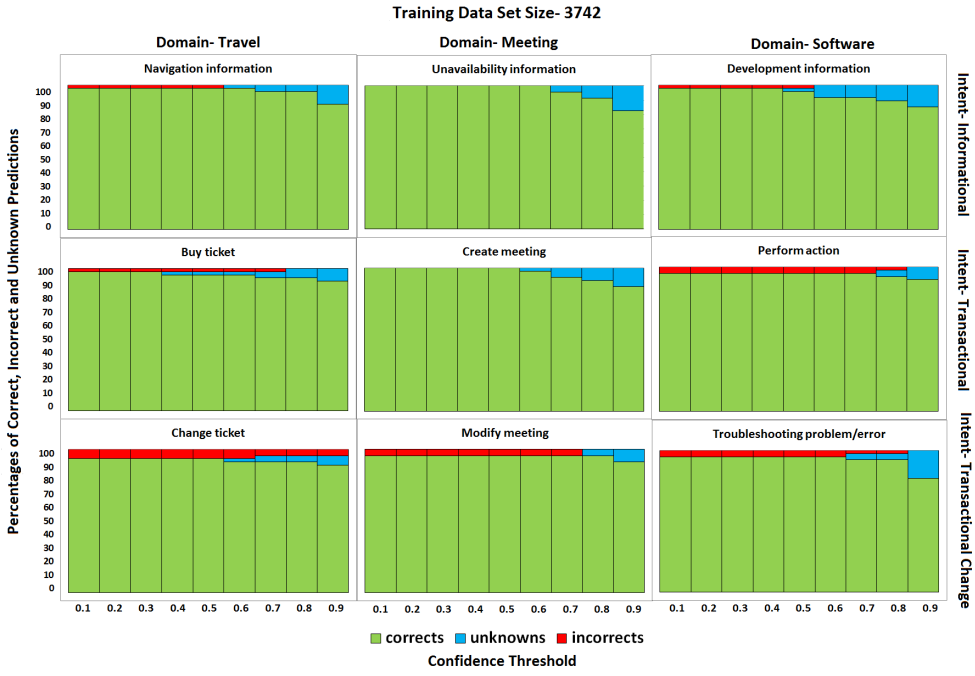


Figure 5.7: Specific Intent Classification Analysis for All Domain-Intent Pairs for Large Data Set Size

5.3 Discussion

We observed from our results that the intent classification of NLU model significantly changes with the training data set size. When the data set size is doubled, number of cor-

rects increase significantly.
Incorrect have arithmetic progression with increase in confidence threshold values for all the set ups while, unknowns have the geometric progression with increase in confidence thresholds for all the setups. Intent classification varies for different types of domains and intents. The requests in intents like buy ticket and change ticket or create meeting and modify meeting are closer to each other as they have same entities present in them. We observed that the, NLU model often confuses while predicting requests for such intents, even if for large data set sizes. So the optimal confidence threshold value may change for each intent and an appropriate threshold can be chosen differently for different intents with the cost analysis for each intent separately.

Optimal confidence threshold can be chosen based on the priority of cost. When incorrects are more expensive than unknowns, smaller values should be chosen for small data sets. When unknowns are more expensive, optimal threshold should be selected as smallest to save the cost. When both are equally expensive, smallest confidence threshold should be chosen for all data sets. We observe a specific pattern in finding out optimal threshold values based on costs and choosing a optimal threshold value having minimum cost in each strategy. Optimal threshold value remains same as 0.4 when incorrects are more expensive for the data set size below 100, it grows linearly till data set size 100 is reached and it again remains constant as 0.8 for the data sets more than 1000. It would be interesting to see, if the optimal threshold would reach 1.0 and for which amount of data set. When unknowns are more expensive, optimal confidence threshold should be chosen as minimum, to have less number of unknowns for all the data set sizes. When incorrects and unknowns both are expensive, optimal threshold trend has a similar pattern of having same optimal threshold value for data sets size below than 100, it becomes double when data set size is doubled after 100, and again remains constant till the data set size 1000 is reached. It again doubles after the data set size 1000 and grows for larger data set sizes. These trends would be useful in understanding how to choose, which multiple of data set size while increasing the data set size for better classification performance based on the confidence threshold value.

Our approach of estimating reliability of a automatic NLU model based on the confidence threshold provided insights about the 2 dimensions which should be considered while choosing an optimal threshold value. First dimension is training data set size, which affects the classification significantly. Second is the type of intents for which NLU model is trained. The simulations of different cost strategies provided insights about choosing optimal threshold based on the penalties associated with number of incorrects and unknowns.

Chapter 6

Discussion

In this chapter, we discuss our work and possible ways to improve or approaches. We then identify limitations and threats to validity in the approaches we designed in RQ3 and RQ4.

6.1 Discussion

In order to address issues related to natural language understanding issues in chatbots, we studied approaches to involve human computation in chatbots. We analyzed the challenges, opportunities and state of the art in this field. To deepen our knowledge and complement the findings in literature study, we designed a hybrid chatbot system and implemented it as a proof of concept.

We designed an approach where high quality data is collected for training NLU model of a chatbot. We chose human computation for collecting this training data, as we believed the advantage to collect training data from real humans would provide the set of real examples that actual humans could ask to a chatbot. Also It is crucial to make sure that training data is diverse and asking different people from different countries can help to reach such diversity. Starting with understanding the need of information, our approach follows pipeline where we first prepare seed input data for brainstorming task based on the information need, then we have brainstorming task to collect data, validation task to validate collected data. The output of our pipeline is high quality data. In our approach requester of as task generates the seed data. It would be interesting if the seed data preparation could also a designed as a separate task on human computation platform. This task would be performed by the more experienced workers as this is a quite tedious work and the requester can ask crowd workers to prepare seed data for the brainstorming task. They may get more diverse seed data examples with the help of workers. We could collect high quality data in our experiment for all 9 intents in a single day, so it was quite faster.

While designing and implementing hybrid chatbot system, we focused more on natural language understanding module of a chatbot and we observed that the trained automatic NLU models do not predict the meaning of user requests reliably. Hence we designed and approach to collect high quality training data and train the NLU model of a chatbot with this data. Reliability of NLU model is estimated with the confidence threshold and this threshold could be found out based on costs assigned to the number of incorrects and unknown requests. These costs are the penalties associated with incorrectly identified requests and the requests which are identified as non-reliable because of the lower confidence value of prediction than the threshold and hence a human computation involvement is needed. The The threshold values vary with the training data set size and types of intents from 0.4 to 0.8. The choice of the threshold may also vary for different domains and based on different types of application, for which a chatbot is built. For the real time applications such as cur-

rent travel/traffic information, chatbot would not afford to have more number of unknowns as it will then need to involve human computation component and the response would be sent back to the user with some delay. For the domains like scheduling meeting, it is fine to experience some delay in responses and hence confidence threshold having the strategy with more expensive unknowns could be considered.

Lets consider a case where some developer is designing a new chatbot for 10 intents. With our approach, he/she can collect 500 high quality request examples for each intent with approximately 365 USD in total and train the NLU model. Following our methodology for reliability estimation, this person can make an educated decision and define the confidence threshold of 0.4, assuring the prediction accuracy of 95%, which will grow over time if user requests with low confidence predictions are routed to workers on the human computation platform.

6.2 Threats to Validity

In this section, we discuss the limitations in our hybrid chatbot system. We also discuss the threats to validity for our approaches we designed for high quality data collection and estimating reliability of intent classification of automatic NLU model.

Hybrid chatbot system

We mainly focused on natural language understanding of user requests and provided a design for NLU model having humans in the loop. We did not focus on the process after the meaning of request is understood till the appropriate response is generated and provided to the user. Though we provided a proof of concept of our hybrid chatbot design by implementing a transportation chatbot, it was not evaluated with live user requests.

High quality data collection

We designed a pipeline to collect high quality training data with the help of human computation. We collected the possible chatbot user requests for specific domains with the brainstorming task by providing seed data examples and validated those requests with validation task on CrowdFlower. For the brainstorming task, we provided the seed examples based on the fixed templates, and we restricted workers to provide requests matching to those templates. These templates potentially could limit the diversity in the imagination of workers while providing the requests. We introduced diversity based on these templates and provided guidance towards this diversity to the workers. But we acknowledge that this guidance provided some bias to them and workers used their skills and imagination less while providing the examples of user requests. We could have achieved more diversity in requests if they could have been also asked to provide requests based on their imaginations and not only based on the request templates.

Our validation task provided us high accuracy, though the task design has a scope of improvement. We applied a same validation task design for all intents. We observed that the workers often were confused while validating the requests semantically. The prominent confusion we observed in validation requests for 'Modify meeting' intent. Modify meeting intent had lots of requests which were actually related to 'Create meeting' intent, those were semantically wrong. But the workers marked them correct and could not provide right judgments for those requests.

We provided different agreements strategies of validation judgments, and showed the accuracies they achieve in validating the requests collected from brainstorming task. We also designed a re-iteration process of brainstorming + validation task for the requests

which were identified as incorrect by validation task, after considering particular agreement strategy. We could achieve high accuracies for all intents for brainstorming + validation task and we did not re-iterate the process for the incorrectly identified requests. It could have been interesting to re-iterate through the process for incorrectly identified requests and observe the accuracies for those iterations. We collected thousands of user requests with our experiment for different intents in different domains, it would be of great interest to collect user requests for larger amounts than thousands and evaluate the preforms of our approach.

Reliability estimation of intent classification

We provided an approach which to pick an optimal confidence threshold value for estimating reliability of intent classification of an automatic NLU model. We started with training the automatic NLU model with high quality data collected from data collection approach. We divided it into train and test set and trained RASA NLU models with different data set sizes, for testing the models with test data. We trained the models with high quality data and tested them with high quality data. It could have been interesting to test the trained models with noisy data. Testing these trained models by real users would have given an idea about how they predict the real user requests which might be noisy, incomplete or irrelevant.

Though we tested our reliability estimation of intent classification for different data set sizes and different types of intents for 9 intents. Changing the number of intents might introduce variance in intent classification and also the optimal confidence threshold value may vary according to the number of intents.

Chapter 7

Conclusion and Future work

7.1 Conclusion

In this thesis work, we research about efficiently involving human computation in chatbots for better understanding user requests. To solve this main research question we started with understanding the architecture of chatbots, the challenges that characterize the training and execution of chatbots, and the opportunities and limitations in using human computation applied to support the chatbot operations. We studied details of these areas thoroughly in our literature survey. An extensive literature study provided us insights how the hybrid chatbot systems with humans in the loop work and also provided us guidelines to design such hybrid chatbot system.

For designing a chatbot system having humans in the loop, we first identified the components of our hybrid chatbot system, and designed the flow of them mainly for two phases-design phase and run time phase. We presented how human computation can be used at both design time and run time. To prove the feasibility of our envisioned system, we implemented a hybrid chatbot system for a transportation domain.

After we design our system we focused on an efficient involvement of humans in hybrid chatbot systems. For this, we first focused on effectively involving humans in a design time phase. We designed an approach to collect high quality training data with the help of human computation to train automatic NLU model. We showed that the approach can achieve high accuracy in collecting and validating high quality training data on human computation platform.

After collecting high quality training data, we trained the NLU model of a chatbot system with that data and analyzed the predictions of user request understanding. We designed an approach to estimate the reliability of predictions of the NLU model, based on the confidence threshold. The approach is then tested for different data set sizes and the penalties for unreliable predictions are calculated. The trend of intent classification for different training data set sizes and different types of intents were observed and analyzed. The validity of approach and results were discussed with the CTO of RASA NLU, and they are appreciated as relevant and beneficial. They are looking forward for a collaboration and future work with us.

7.2 Future work

For the limitations discussed in Chapter 6 and also apart from those limitations, we would also like to extend this research in certain aspects. We discuss them in this section.

Improving validation task

One of the limitations we discussed in Chapter 6, is about improving the validation task for validating user requests collected from brainstorming task. We will focus more on improving validation task, where the performance was poor in identifying semantically incorrect requests. The task design would be improved by providing knowledge to the workers about how to identify if the request is semantically correct or not. We will design test questions for testing the understanding of workers about the semantical correctness of the requests would be designed to ensure the quality of validation judgments.

Providing re-iteration process for high quality data collection approach

The procedure of re-iteration of brainstorming task and validation task can be designed. The work in identifying the re-iterations based on the quantity requirement of high quality training data can be done. Designing this re-iteration procedure would provide an idea about how much cost is needed for a particular quantity of high quality data with high accuracy and how long will it take to collect the required amount of high quality data.

Retraining

To reduce the number of times we go to the human computation component, retraining methodology could be designed. We have designed the retraining module in our hybrid chatbot system, we did not deliver it in this thesis. Machine component could be trained with each previously identified user requests by human computation component. This would help in serving similar types of incoming user requests.

Based on this idea of retraining, we performed a short retraining experiment for data set 59, keeping the confidence threshold values as 0.3 and then 0.5. We retrained automatic NLU model with a request predicted by human computation component at run time. The preliminary results of our short experiments were quite interesting and as they reduced number of the non-reliable requests to half. Thus it reduced the number of times a request sent to human computation component. It would be interesting to extend this experiment to different data set sizes and observe the retraining effect for different confidence threshold values.

Estimating reliability of entity extraction

We provided an approach to estimate the reliability of intent classification of automatic NLU module. An approach to estimate the reliability of entity extraction by the automatic NLU module would be an interesting extension of reliability estimation. Many automatic tools do not provide confidence values for entity extraction, so first the approach to identify the confidence value for entity extraction would be designed and then estimating the reliability of this confidence would be investigated.

Bibliography

[1] Sameera A Abdul-Kader and John Woods. Survey on chatbot design techniques in speech conversation systems. *Int. J. Adv. Comput. Sci. Appl.(IJACSA)*, 6(7), 2015.

[2] Eman Saad AlHagbani and Muhammad Badruddin Khan. Challenges facing the development of the arabic chatbot. In *First International Workshop on Pattern Recognition*, pages 100110Y–100110Y. International Society for Optics and Photonics, 2016.

[3] Mohammad Allahbakhsh, Boualem Benatallah, Aleksandar Ignjatovic, Hamid Reza Motahari-Nezhad, Elisa Bertino, and Schahram Dustdar. Quality control in crowdsourcing systems: Issues and directions. *IEEE Internet Computing*, 17(2):76–81, 2013.

[4] Michael S Bernstein, Joel Brandt, Robert C Miller, and David R Karger. Crowds in two seconds: Enabling realtime crowd-powered interfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 33–42. ACM, 2011.

[5] Michael S Bernstein, David R Karger, Robert C Miller, and Joel Brandt. Analytic methods for optimizing realtime crowdsourcing. *arXiv preprint arXiv:1204.2995*, 2012.

[6] Jeffrey P Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samual White, et al. Vizwiz: nearly real-time answers to visual questions. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 333–342. ACM, 2010.

[7] Jon Chamberlain, Udo Kruschwitz, and Massimo Poesio. Methods for engaging and evaluating users of human computation systems. In *Handbook of Human Computation*, pages 679–694. Springer, 2013.

[8] Jenny J Chen, Natala J Menezes, Adam D Bradley, and TA North. Opportunities for crowdsourcing research on amazon mechanical turk. *Interfaces*, 5(3), 2011.

[9] Yun-Nung Chen, Jeffrey P Bigham, et al. Real-time on-demand crowd-powered entity extraction. *arXiv preprint arXiv:1704.03627*, 2017.

[10] Lydia B Chilton, Clayton T Sims, Max Goldman, Greg Little, and Robert C Miller. Seaweed: A web application for designing economic games. In *Proceedings of the ACM SIGKDD workshop on human computation*, pages 34–35. ACM, 2009.

[11] Justin Cranshaw, Emad Elwany, Todd Newman, Rafal Kocielnik, Bowen Yu, Sandeep Soni, Jaime Teevan, and Andrés Monroy-Hernández. Calendar. help: Designing a

workflow-based scheduling agent with humans in the loop. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 2382–2393. ACM, 2017.

[12] OV Deryugina. Chatterbots. *Scientific and Technical Information Processing*, 37(2):143–147, 2010.

[13] Anhai Doan, Raghu Ramakrishnan, and Alon Y Halevy. Crowdsourcing systems on the world-wide web. *Communications of the ACM*, 54(4):86–96, 2011.

[14] Steven Dow, Anand Kulkarni, Scott Klemmer, and Björn Hartmann. Shepherding the crowd yields better work. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1013–1022. ACM, 2012.

[15] T. Ekeinhor-Komi, J. L. Bouraoui, R. Laroche, and F. LefÃšvre. Towards a virtual personal assistant based on a user-defined portfolio of multi-domain vocal applications. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 106–113, Dec 2016.

[16] Richard D Evans, James X Gao, Sara Mahdikhah, Mourad Messaadia, and David Baudry. A review of crowdsourcing literature related to the manufacturing industry. *Journal of Advanced Management Science*, 4(3):224–321, 2015.

[17] Ailbhe Finnerty, Pavel Kucherbaev, Stefano Tranquillini, and Gregorio Convertino. Keep it simple: Reward and task design in crowdsourcing. In *Proceedings of the Biannual Conference of the Italian Chapter of SIGCHI*, page 14. ACM, 2013.

[18] Michael Fischer and Monica Lam. From books to bots: Using medical literature to create a chat bot. In *Proceedings of the First Workshop on IoT-enabled Healthcare and Wellness Technologies and Systems*, pages 23–28. ACM, 2016.

[19] Cinzia Cappiello Boualem Benatallah Florian Daniel, Pavel Kucherbaev and Mohammad Allahbakhsh. *Quality Control in Crowdsourcing: A Survey of Quality Attributes,Assessment Techniques and Assurance Actions*, 2017. in submission to ACM Computing Surveys journal ACM.

[20] David Griol, Zoraida Callejas, Ramón López-Cózar, and Giuseppe Riccardi. A domain-independent statistical methodology for dialog management in spoken dialog systems. *Computer Speech & Language*, 28(3):743–768, 2014.

[21] David Griol, Lluís F Hurtado, Encarna Segarra, and Emilio Sanchis. A statistical approach to spoken dialog systems design and evaluation. *Speech Communication*, 50(8):666–682, 2008.

[22] Jessi Hempel. Facebook launches m, its bold answer to siri and cortana, 2015.

[23] Chang Hu, Benjamin B Bederson, Philip Resnik, and Yakov Kronrod. Monotrans2: A new human computation system to support monolingual translation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1133–1136. ACM, 2011.

[24] Gaoping Huang and Alexander J Quinn. Bluesky: Crowd-powered uniform sampling of idea spaces. In *Proceedings of the 2017 ACM SIGCHI Conference on Creativity and Cognition*, pages 119–130. ACM, 2017.

[25] Ting-Hao Kenneth Huang, Amos Azaria, and Jeffrey P Bigham. Instructablecrowd: Creating if-then rules via conversations with the crowd. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 1555–1562. ACM, 2016.

[26] Ting-Hao Kenneth Huang, Walter S Lasecki, and Jeffrey P Bigham. Guardian: A crowd-powered spoken dialog system for web apis. In *Third AAAI conference on human computation and crowdsourcing*, 2015.

[27] Oana Inel, Khalid Khamkham, Tatiana Cristea, Anca Dumitrache, Arne Rutjes, Jelle van der Ploeg, Lukasz Romaszko, Lora Aroyo, and Robert-Jan Sips. Crowdtruth: Machine-human computation framework for harnessing disagreement in gathering annotated data. In *International Semantic Web Conference*, pages 486–504. Springer, 2014.

[28] BI Intelligence. Messaging apps are now bigger than social networks. *Business Insider, September*, 20, 2016.

[29] Panagiotis G Ipeirotis. Analyzing the amazon mechanical turk marketplace. *XRDS: Crossroads, The ACM Magazine for Students*, 17(2):16–21, 2010.

[30] PG Ipeirotis, R Chandrasekar, and P Bennett. Report on the human computation workshop. *HCOMP 2010*, 2010.

[31] Zongcheng Ji, Zhengdong Lu, and Hang Li. An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988*, 2014.

[32] Roman Khazankin, Daniel Schall, and Schahram Dustdar. Predicting qos in scheduled crowdsourcing. In *International Conference on Advanced Information Systems Engineering*, pages 460–472. Springer, 2012.

[33] Aniket Kittur, Ed H Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 453–456. ACM, 2008.

[34] Aniket Kittur, Boris Smus, Susheel Khamkar, and Robert E Kraut. Crowdforge: Crowdsourcing complex work. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 43–52. ACM, 2011.

[35] Frank Kleemann, G Günter Voß, and Kerstin Rieder. Un (der) paid innovators: The commercial utiliza-tion of consumer work through crowdsourcing. *Science, technology & innovation studies*, 4(1):PP–5, 2008.

[36] Pavel Kucherbaev, Florian Daniel, Maurizio Marchese, Fabio Casati, and Brian Reavey. Toward effective tasks navigation in crowdsourcing. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, pages 401–404. ACM, 2014.

[37] Anand Kulkarni, Matthew Can, and Björn Hartmann. Collaboratively crowdsourcing workflows with turkomatic. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1003–1012. ACM, 2012.

[38] Walter Lasecki, Christopher Miller, Adam Sadilek, Andrew Abumoussa, Donato Borrello, Raja Kushalnagar, and Jeffrey Bigham. Real-time captioning by groups of non-experts. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pages 23–34. ACM, 2012.

[39] Walter S Lasecki. Real-time conversational crowd assistants. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 2725–2730. ACM, 2013.

[40] Walter S Lasecki, Kyle I Murray, Samuel White, Robert C Miller, and Jeffrey P Bigham. Real-time crowd control of existing interfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 23–32. ACM, 2011.

[41] Walter S Lasecki, Phyo Thiha, Yu Zhong, Erin Brady, and Jeffrey P Bigham. Answering visual questions with conversational crowd assistants. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility*, page 18. ACM, 2013.

[42] Walter S. Lasecki, Rachel Wesley, Jeffrey Nichols, Anand Kulkarni, James F. Allen, and Jeffrey P. Bigham. Chorus: A crowd-powered conversational assistant. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 151–162, New York, NY, USA, 2013. ACM.

[43] Edith Law and Luis Von Ahn. Input-agreement: a new mechanism for collecting data using human computation games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1197–1206. ACM, 2009.

[44] James Lester, Karl Branting, and Bradford Mott. Conversational agents. *The Practical Handbook of Internet Computing*, pages 220–240, 2004.

[45] Wenmin Li, Jiawei Han, and Jian Pei. Cmar: Accurate and efficient classification based on multiple class-association rules. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 369–376. IEEE, 2001.

[46] Greg Little, Lydia B Chilton, Max Goldman, and Robert C Miller. Turkit: human computation algorithms on mechanical turk. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 57–66. ACM, 2010.

[47] Yang Liu, E. Shriberg, A. Stolcke, D. Hillard, M. Ostendorf, and M. Harper. Enriching speech recognition with automatic detection of sentence boundaries and disfluencies. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5):1526–1540, Sept 2006.

[48] Daniel Marcu. Machine translation techniques, May 12 2009. US Patent 7,533,013.

[49] Dominic W Massaro, Ying Liu, Trevor H Chen, and Charles Perfetti. A multilingual embodied conversational agent for tutoring speech and language learning. In *INTERSPEECH*, 2006.

[50] Michael McTear, Zoraida Callejas, and David Griol. *The conversational interface*. Springer, 2016.

[51] Patrick Minder, Sven Seuken, Abraham Bernstein, and Mengia Zollinger. Crowdmanager-combinatorial allocation and pricing of crowdsourcing tasks with time constraints. In *Workshop on Social Computing and User Generated Content in conjunction with ACM Conference on Electronic Commerce (ACM-EC 2012)*, 2012.

[52] Adam Miner, Amanda Chow, Sarah Adler, Ilia Zaitsev, Paul Tero, Alison Darcy, and Andreas Paepcke. Conversational agents and mental health: Theory-informed assessment of language and affect. In *Proceedings of the Fourth International Conference on Human Agent Interaction*, pages 123–130. ACM, 2016.

[53] Bill Murdock. *How to select a threshold for acting using confidence scores*, june 2016. `https://developer.ibm.com/watson/blog/2016/06/23/how-to-select-a-threshold-for-acting-using-confidence-scores/`.

[54] Ibrahim Naji. *Text Classification Threshold Performance Graph*, jan 2013. `http://thinknook.com/text-classification-threshold-performance-graph-2013-01-20/`.

[55] Tim Paek and David Maxwell Chickering. The markov assumption in spoken dialogue management. In *6th SIGDIAL Workshop on Discourse and Dialogue*, 2005.

[56] Geert-Jan Houben Pavel Kucherbaev, Alessandro Bozzon. *Human Aided Bots*, 2017. in submission to to IEEE Internet Computing journal.

[57] Silvia Quarteroni and Suresh Manandhar. Designing an interactive open-domain question answering system. *Natural Language Engineering*, 15(01):73–95, 2009.

[58] Alexander J Quinn and Benjamin B Bederson. A taxonomy of distributed human computation. *Human-Computer Interaction Lab Tech Report, University of Maryland*, 2009.

[59] Alexander J Quinn and Benjamin B Bederson. Human computation: a survey and taxonomy of a growing field. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1403–1412. ACM, 2011.

[60] Nicole M Radziwill and Morgan C Benton. Evaluating quality of chatbots and intelligent conversational agents. *arXiv preprint arXiv:1704.04579*, 2017.

[61] Bahareh Rahmanian and Joseph G Davis. User interface design for crowdsourcing systems. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, pages 405–408. ACM, 2014.

[62] Elliot Salisbury, Sebastian Stein, and Sarvapali Ramchurn. Real-time opinion aggregation methods for crowd robotics. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 841–849. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[63] Gopal Ravi Sankar, Jean Greyling, Dieter Vogts, and Mathys C du Plessis. Models towards a hybrid conversational agent for contact centres. In *Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology*, pages 200–209. ACM, 2008.

[64] Denis Savenkov, Scott Weitzner, and Eugene Agichtein. Crowdsourcing for (almost) real-time question answering. In *Workshop on Human-Computer Question Answering, NAACL*, 2016.

[65] Daniel Schall, Florian Skopik, and Schahram Dustdar. Expert discovery and interactions in mixed service-oriented systems. *IEEE Transactions on services computing*, 5(2):233–245, 2012.

[66] John R Searle. A taxonomy of illocutionary acts. 1975.

[67] Nitin Seemakurty, Jonathan Chu, Luis Von Ahn, and Anthony Tomasic. Word sense disambiguation via human computation. In *Proceedings of the acm sigkdd workshop on human computation*, pages 60–63. ACM, 2010.

[68] Jerry O Talton, Daniel Gibson, Lingfeng Yang, Pat Hanrahan, and Vladlen Koltun. Exploratory modeling with collaborative design spaces. *ACM Transactions on Graphics-TOG*, 28(5):167, 2009.

[69] David R Traum and Staffan Larsson. The information state approach to dialogue management. In *Current and new directions in discourse and dialogue*, pages 325–353. Springer, 2003.

[70] Miroslav Vodolán and Filip Jurčíček. Data collection for interactive learning through the dialog. *arXiv preprint arXiv:1603.09631*, 2016.

[71] Luis Von Ahn. Human computation. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 1–2. IEEE, 2008.

[72] Luis Von Ahn, Manuel Blum, Nicholas J Hopper, and John Langford. Captcha: Using hard ai problems for security. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 294–311. Springer, 2003.

[73] Luis Von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326. ACM, 2004.

[74] Luis Von Ahn and Laura Dabbish. Designing games with a purpose. *Communications of the ACM*, 51(8):58–67, 2008.

[75] Marilyn Walker, Amanda Stent, François Mairesse, and Rashmi Prasad. Individual and domain adaptation in sentence planning for dialogue. *J. Artif. Int. Res.*, 30(1):413–456, November 2007.

[76] Richard Wallace. The elements of aiml style. *Alice AI Foundation*, 2003.

[77] Richard S Wallace. *Be Your Own Botmaster: The Step By Step Guide to Creating, Hosting and Selling Your Own AI Chat Bot On Pandorabots*. ALICE AI foundations, Incorporated, 2003.

[78] Joseph Weizenbaum. Elizaâa computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.

[79] Dean Withey. *Where to get Chatbot Training Data (and what it is)*, june 2017. `https://blog.ubisend.com/optimise-chatbots/chatbot-training-data`.

[80] Lixiu Yu, Aniket Kittur, and Robert E Kraut. Distributed analogical idea generation: inventing with crowds. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 1245–1254. ACM, 2014.

[81] Lixiu Yu, Aniket Kittur, and Robert E Kraut. Searching for analogical ideas with crowds. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 1225–1234. ACM, 2014.

[82] T. Zhao, K. Lee, and M. Eskenazi. Dialport: Connecting the spoken dialog research community to real user data. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 83–90, Dec 2016.

Appendix A

Figures and Tables

A.1 Classification Matrices

1. Training data set size 15

	0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
corrects	35	30	7	1	0	0	0	0	0
incorrects	381	196	40	0	0	0	0	0	0
unknowns	0	190	369	415	416	416	416	416	416

Table A.1: Classification Matrix - Training Data Set Size 15

2. Training data set size 30

	0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
corrects	247	234	142	53	15	3	1	0	0
incorrects	169	138	36	10	3	1	0	0	0
unknowns	0	44	238	353	399	412	414	416	416

Table A.2: Classification Matrix - Training Data Set Size 30

3. Training data set size 59

	0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
corrects	345	343	298	199	108	51	18	4	0
incorrects	71	67	40	16	6	2	1	0	0
unknowns	0	6	78	201	302	362	397	412	416

Table A.3: Classification Matrix - Training Data Set Size 59

4. Training data set size 117

	0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
corrects	369	368	356	324	269	188	103	38	7
incorrects	47	46	37	22	12	7	3	1	0
unknowns	0	2	23	70	135	222	310	377	409

Table A.4: Classification Matrix - Training Data Set Size 117

5. Training data set size 234

	0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
corrects	381	381	378	368	349	311	257	159	55
incorrects	35	34	31	25	17	11	5	3	1
unknowns	0	0	7	22	50	94	153	255	360

Table A.5: Classification Matrix - Training Data Set Size 234

6. Training data set size 468

	0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
corrects	390	390	389	386	374	356	326	274	155
incorrects	26	26	25	21	16	10	6	3	1
unknowns	0	0	2	9	27	50	84	139	260

Table A.6: Classification Matrix - Training Data Set Size 468

7. Training data set size 936

	0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
corrects	392	392	387	384	378	364	349	320	257
incorrects	24	24	26	23	20	16	12	7	4
unknowns	0	0	3	9	18	36	55	89	156

Table A.7: Classification Matrix - Training Data Set Size 936

8. Training data set size 1871

	0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
corrects	393	393	387	386	382	375	367	353	320
incorrects	23	23	27	25	23	18	16	10	3
unknowns	0	0	2	5	11	23	33	53	94

Table A.8: Classification Matrix - Training Data Set Size 1871

9. Training data set size 2700

	0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
corrects	400	400	400	400	399	397	390	379	360
incorrects	16	16	15	14	13	11	9	5	2
unknowns	0	0	1	2	4	8	17	32	54

Table A.9: Classification Matrix - Training Data Set Size 2700

10. Training data set size 3742

	0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
corrects	404	404	404	403	402	398	391	386	363
incorrects	12	12	12	12	10	10	8	4	2
unknowns	0	0	0	1	4	8	17	26	51

Table A.10: Classification Matrix - Training Data Set Size 3742

A.2 Templates- Input Seed Data Files of all Brainstorming Tasks

1. Travel- Buy ticket

Request Template	Sequence Order
I want to buy a ticket to {YOUR DESTINATION} from {YOUR SOURCE}, {YOUR TYPE OF TRAVEL} on {DATE}	First- Destination, Second- Source, Third- Type of Travel, Fourth- Date
I want to buy a ticket to {YOUR DESTINATION} from {YOUR SOURCE}, on {DATE}, for {YOUR TYPE OF TRAVEL}	First- Destination, Second- Source, Third- Date, Fourth- Type of Travel
I want to buy a ticket to {YOUR DESTINATION} for {YOUR TYPE OF TRAVEL} from {YOUR SOURCE}, on {DATE}	First- Destination, Second- Type of Travel, Third- Source, Fourth- Date
I want to buy a ticket to {YOUR DESTINATION} for {YOUR TYPE OF TRAVEL} on {DATE} from {YOUR SOURCE}	First- Destination, Second- Type of Travel, Third- Date, Fourth- Source
I want to buy a ticket to {YOUR DESTINATION} on {DATE} for {YOUR TYPE OF TRAVEL} from {YOUR SOURCE}	First- Destination, Second- Date, Third- Type of Travel, Fourth- Source
I want to buy a ticket from {YOUR SOURCE} to {YOUR DESTINATION} for {YOUR TYPE OF TRAVEL} on {DATE}	First- Source, Second-Destination, Third- Type of Travel, Fourth- Date
I want to buy a ticket from {YOUR SOURCE} to {YOUR DESTINATION} on {DATE} for {YOUR TYPE OF TRAVEL}	First- Source, Second-Destination, Third- Date, Fourth- Type of Travel
I want to buy a ticket from {YOUR SOURCE} for {YOUR TYPE OF TRAVEL} to {YOUR DESTINATION} on {DATE}	First- Source, Second- Type of Travel, Third- Destination, Fourth- Date
I want to buy a ticket from {YOUR SOURCE} for {YOUR TYPE OF TRAVEL} on {DATE} to {YOUR DESTINATION}	First- Source, Second- Type of Travel, Third- Date, Fourth- Destination
I want to buy a ticket from {YOUR SOURCE} on {DATE} for {YOUR TYPE OF TRAVEL} to {YOUR DESTINATION}	First- Source, Second- Date, Third- Type of Travel, Fourth- Destination
I want to buy a ticket from {YOUR SOURCE} on {DATE} to {YOUR DESTINATION} for {YOUR TYPE OF TRAVEL}	First- Source, Second- Date, Third- Destination , Fourth- Type of Travel
I want to buy a ticket for {YOUR TYPE OF TRAVEL} from {YOUR SOURCE} to {YOUR DESTINATION} on {DATE}	First- Type of Travel, Second- Source, Third- Destination , Fourth- Date
I want to buy a ticket for {YOUR TYPE OF TRAVEL} from {YOUR SOURCE} on {DATE} to {YOUR DESTINATION}	First- Type of Travel, Second- Source, Third- Date , Fourth- Destination
I want to buy a ticket for {YOUR TYPE OF TRAVEL} to {YOUR DESTINATION} from {YOUR SOURCE} on {DATE}	First- Type of Travel, Second- Destination, Third- Source , Fourth- Date
I want to buy a ticket for {YOUR TYPE OF TRAVEL} to {YOUR DESTINATION} on {DATE} from {YOUR SOURCE}	First- Type of Travel, Second- Destination, Third- Date , Fourth- Source
I want to buy a ticket for {YOUR TYPE OF TRAVEL} on {DATE} to {YOUR DESTINATION} from {YOUR SOURCE}	First- Type of Travel, Second- Date, Third- Destination , Fourth- Source
I want to buy a ticket for {YOUR TYPE OF TRAVEL} on {DATE} from {YOUR SOURCE} to {YOUR DESTINATION}	First- Type of Travel, Second- Date, Third- Source , Fourth- Destination
I want to buy a ticket on {DATE} for {YOUR TYPE OF TRAVEL} from {YOUR SOURCE} to {YOUR DESTINATION}	First- Date, Second- Type of Travel, Third- Source , Fourth- Destination
I want to buy a ticket on {DATE} for {YOUR TYPE OF TRAVEL} to {YOUR DESTINATION} from {YOUR SOURCE}	First- Date, Second- Type of Travel, Third- Destination , Fourth- Source
I want to buy a ticket on {DATE} from {YOUR SOURCE} for {YOUR TYPE OF TRAVEL} to {YOUR DESTINATION}	First- Date, Second- Source , Third- Type of Travel , Fourth- Destination
I want to buy a ticket on {DATE} from {YOUR SOURCE} to {YOUR DESTINATION} for {YOUR TYPE OF TRAVEL}	First- Date, Second- Source , Third- Destination , Fourth- Type of Travel
I want to buy a ticket on {DATE} to {YOUR DESTINATION} from {YOUR SOURCE} for {YOUR TYPE OF TRAVEL}	First- Date, Second- Destination , Third- Source , Fourth- Type of Travel
I want to buy a ticket on {DATE} to {YOUR DESTINATION} for {YOUR TYPE OF TRAVEL} from {YOUR SOURCE}	First- Date, Second- Destination , Third- Type of Travel , Fourth- Source
I want to buy a ticket from {YOUR SOURCE}, {YOUR TYPE OF TRAVEL} on {DATE}	First- Source, Second- Type of Travel, Third- Date
I want to buy a ticket from {YOUR SOURCE}, on {DATE}, for {YOUR TYPE OF TRAVEL}	First- Source, Second- Date, Third- Type of Travel
I want to buy a ticket for {YOUR TYPE OF TRAVEL} from {YOUR SOURCE}, on {DATE}	First- Type of Travel, Second- Source, Third- Date
I want to buy a ticket for {YOUR TYPE OF TRAVEL} on {DATE} from {YOUR SOURCE}	First- Type of Travel, Second- Date, Third- Source
I want to buy a ticket on {DATE} from {YOUR SOURCE} for {YOUR TYPE OF TRAVEL}	First- Date, Second- Source, Third- Type of Travel
I want to buy a ticket on {DATE} for {YOUR TYPE OF TRAVEL} from {YOUR SOURCE}	First- Date, Second- Type of Travel, Third- Source
I want to buy a ticket to {YOUR DESTINATION} for {YOUR TYPE OF TRAVEL} on {DATE}	First-Destination, Second- Type of Travel, Third- Date
I want to buy a ticket to {YOUR DESTINATION} on {DATE} for {YOUR TYPE OF TRAVEL}	First-Destination, Second- Date, Third- Type of Travel
I want to buy a ticket for {YOUR TYPE OF TRAVEL} to {YOUR DESTINATION} on {DATE}	First- Type of Travel, Second- Destination, Third- Date
I want to buy a ticket for {YOUR TYPE OF TRAVEL} on {DATE} to {YOUR DESTINATION}	First- Type of Travel, Second- Date, Third- Destination
I want to buy a ticket on {DATE} for {YOUR TYPE OF TRAVEL} to {YOUR DESTINATION}	First- Date, Second- Type of Travel, Third- Destination
I want to buy a ticket on {DATE} to {YOUR DESTINATION} for {YOUR TYPE OF TRAVEL}	First- Date, Second- Destination , Third- Type of Travel
I want to buy a ticket from {YOUR SOURCE} to {YOUR DESTINATION} on {DATE}	First- Source, Second- Destination , Third- Date
I want to buy a ticket from {YOUR SOURCE} on {DATE} to {YOUR DESTINATION}	First- Source, Second- Date , Third- Destination
I want to buy a ticket to {YOUR DESTINATION} from {YOUR SOURCE} on {DATE}	First- Destination, Second- Source , Third- Date
I want to buy a ticket to {YOUR DESTINATION} on {DATE} from {YOUR SOURCE}	First- Destination, Second- Date , Third- Source
I want to buy a ticket on {DATE} to {YOUR DESTINATION} from {YOUR SOURCE}	First- Date, Second- Destination , Third- Source
I want to buy a ticket on {DATE} from {YOUR SOURCE} to {YOUR DESTINATION}	First- Date, Second- Source , Third- Destination
I want to buy a ticket for {YOUR TYPE OF TRAVEL} from {YOUR SOURCE} to {YOUR DESTINATION}	First- Type of Travel, Second- Source , Third- Destination
I want to buy a ticket for {YOUR TYPE OF TRAVEL} to {YOUR DESTINATION} from {YOUR SOURCE}	First- Type of Travel, Second- Destination , Third- Source
I want to buy a ticket from {YOUR SOURCE} for {YOUR TYPE OF TRAVEL} to {YOUR DESTINATION}	First- Source , Second- Type of Travel , Third- Destination
I want to buy a ticket from {YOUR SOURCE} to {YOUR DESTINATION} for {YOUR TYPE OF TRAVEL}	First- Source , Second- Destination , Third- Type of Travel
I want to buy a ticket to {YOUR DESTINATION} from {YOUR SOURCE} for {YOUR TYPE OF TRAVEL}	First- Destination , Second- Source , Third- Type of Travel
I want to buy a ticket to {YOUR DESTINATION} for {YOUR TYPE OF TRAVEL} from {YOUR SOURCE}	First- Destination , Second- Type of Travel , Third- Source
I want to buy a ticket {YOUR TYPE OF TRAVEL} on {DATE}	First- Type of Travel, Second- Date
I want to buy a ticket on {DATE}, for {YOUR TYPE OF TRAVEL}	First- Date, Second- Type of Travel
I want to buy a ticket from {YOUR SOURCE}, on {DATE}	First- Source, Second- Date
I want to buy a ticket on {DATE} from {YOUR SOURCE}	First- Date, Second- Source
I want to buy a ticket from {YOUR SOURCE} for {YOUR TYPE OF TRAVEL}	First- Source, Second- Type of Travel
I want to buy a ticket for {YOUR TYPE OF TRAVEL} from {YOUR SOURCE}	First- Type of Travel, Second- Source
I want to buy a ticket from {YOUR SOURCE}, to {YOUR DESTINATION}	First- Source, Second- Destination
I want to buy a ticket to {YOUR DESTINATION} from {YOUR SOURCE}	First- Destination, Second- Source
I want to buy a ticket to {YOUR DESTINATION} for {YOUR TYPE OF TRAVEL}	First- Destination, Second- Type of Travel
I want to buy a ticket for {YOUR TYPE OF TRAVEL} to {YOUR DESTINATION}	First- Type of Travel, Second- Destination
I want to buy a ticket to {YOUR DESTINATION} on {DATE}	First- Destination, Second- Date
I want to buy a ticket on {DATE} to {YOUR DESTINATION}	First- Date, Second- Destination
I want to buy a ticket to {YOUR DESTINATION}	Required Entity- Destination
I want to buy a ticket from {YOUR SOURCE}	Required Entity- Source
I want to buy a ticket for {YOUR TYPE OF TRAVEL}	Required Entity- Type of Travel
I want to buy a ticket on {DATE}	Required Entity- Date

Figure A.1: Request Template for Travel- Buy ticket intent

2. Travel- Change ticket

Request Template	Sequence Order
I want to change my ticket to {YOUR DESTINATION} from {YOUR SOURCE} on {DATE}	First- Destination, Second- Source, Third- Date
I want to change my ticket to {YOUR DESTINATION} on {DATE} from {SOURCE}	First- Destination, Second- Date, Third- Source
I want to change my ticket from {YOUR SOURCE} to {YOUR DESTINATION} on {DATE}	First- Source, Second- Destination, Third- Date
I want to change my ticket from {YOUR SOURCE} on {DATE} to {YOUR DESTINATION}	First- Source, Second-Date, Third- Destination
I want to change my ticket on {DATE} from {YOUR SOURCE} to {YOUR DESTINATION}	First- Date, Second- Source, Third- Destination
I want to change my ticket on {DATE} to {YOUR DESTINATION} from {YOUR SOURCE}	First- Date, Second- Destination, Third- Source
I want to change my ticket to {YOUR DESTINATION} from {YOUR SOURCE}	First- Destination, Second- Source
I want to change my ticket to {YOUR DESTINATION} on {DATE}	First- Destination, Second- Date
I want to change my ticket from {YOUR SOURCE} to {YOUR DESTINATION}	First- Source, Second- Destination
I want to change my ticket from {YOUR SOURCE} on {DATE}	First- Source, Second- Date
I want to change my ticket on {DATE} from {YOUR SOURCE}	First- Date, Second- Source
I want to change my ticket on {DATE} routes to travel to {YOUR DESTINATION}	First- Date, Second- Destination
I want to change my ticket to {YOUR DESTINATION}	Required Travel Entity- Destination
I want to change my ticket from {YOUR SOURCE}	Required Travel Entity- Source
I want to change my ticket on {DATE}	Required Travel Entity- Date

Figure A.2: Request Template for Travel- Change ticket intent

3. Meeting- Unavailability information

Request Template	Sequence Order
I cannot attend meeting at {ORIGINAL TIME}, I can attend it at {YOUR PROPOSED TIME} at {PLACE}	First- Original Time, Second- Proposed Time, Third- Place
I cannot attend meeting at {ORIGINAL TIME}, I can attend it at {PLACE} at {YOUR PROPOSED TIME}	First- Original Time, Second- Place, Third- Proposed Time
I can attend meeting at {YOUR PROPOSED TIME}, as I cannot attend it at {ORIGINAL TIME}, at {PLACE}	First- Proposed Time, Second- Original Time, Third- Place
I can attend meeting at {YOUR PROPOSED TIME} at {PLACE}, as I cannot attend it at {ORIGINAL TIME}	First- Proposed Time, Second-Place, Third- Original Time
I can attend meeting at {PLACE} at {YOUR PROPOSED TIME}, as I cannot attend it at {ORIGINAL TIME}	First- Place, Second- Proposed Time, Third- Original Time
I cannot attend meeting at {PLACE} at {ORIGINAL TIME}, I can attend it at {YOUR PROPOSED TIME}	First- Place, Second- Original Time, Third- Proposed Time
I cannot attend meeting at {ORIGINAL TIME}, I can attend it at {YOUR PROPOSED TIME}	First- Original Time, Second- Proposed Time
I cannot attend meeting at {ORIGINAL TIME} at {PLACE}	First- Original Time, Second- Place
I can attend meeting at {YOUR PROPOSED TIME}, as I cannot attend it at {ORIGINAL TIME}	First- Proposed Time, Second- Original Time
I can attend meeting at {YOUR PROPOSED TIME} at { PLACE}	First- Proposed Time, Second- Place
I can attend meeting at {PLACE} at {YOUR PROPOSED TIME}	First- Place, Second- Proposed Time
I cannot attend meeting at {PLACE} at {ORIGINAL TIME}	First- Place, Second- Original Time
I cannot attend meeting at {ORIGINAL TIME}	Required Travel Entity- Original Time
I can attend meeting at {YOUR PROPOSED TIME}	Required Travel Entity- Proposed Time
I can attend meeting at {PLACE}	Required Travel Entity- Place

Figure A.3: Request Template for Meeting- Unavailability information intent

4. Meeting- Create meeting

Request Template	Sequence Order
I want to create a meeting for {YOUR NAMES/NUMBER OF PARTICIPANTS} starting from {YOUR START TIME OF A MEETING}, of {YOUR DURATION OF A MEETING} at {YOUR PLACE OF A MEETING}	First- Names/Number of participants, Second- Start time of the meeting, Third- Duration of the meeting, Fourth- Place of the meeting
I want to create a meeting for {YOUR NAMES/NUMBER OF PARTICIPANTS} starting from {YOUR START TIME OF A MEETING}, at {YOUR PLACE OF A MEETING}, for {YOUR DURATION OF A MEETING}	First- Names/Number of participants, Second- Start time of the meeting, Third- Place of the meeting, Fourth- Duration of the meeting
I want to create a meeting for {YOUR NAMES/NUMBER OF PARTICIPANTS} for {YOUR DURATION OF A MEETING} starting from {YOUR START TIME OF A MEETING}, at {YOUR PLACE OF A MEETING}	First- Names/Number of participants, Second- Duration of the meeting, Third- Start time of the meeting, Fourth- Place of the meeting
I want to create a meeting for {YOUR NAMES/NUMBER OF PARTICIPANTS} for {YOUR DURATION OF A MEETING} at {YOUR PLACE OF A MEETING} starting from {YOUR START TIME OF A MEETING}	First- Names/Number of participants, Second- Duration of the meeting, Third- Place of the meeting, Fourth- Start time of the meeting
I want to create a meeting for {YOUR NAMES/NUMBER OF PARTICIPANTS} at {YOUR PLACE OF A MEETING} starting from {YOUR START TIME OF A MEETING} for {YOUR DURATION OF A MEETING}	First- Names/Number of participants, Second- Place of the meeting, Third- Start time of the meeting, Fourth- Duration of the meeting
I want to create a meeting for {YOUR NAMES/NUMBER OF PARTICIPANTS} at {YOUR PLACE OF A MEETING} for {YOUR DURATION OF A MEETING} starting from {YOUR START TIME OF A MEETING}	First- Names/Number of participants, Second- Place of the meeting, Third- Duration of the meeting, Fourth- Start time of the meeting
I want to create a meeting starting from {YOUR START TIME OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS} for {YOUR DURATION OF A MEETING} at {YOUR PLACE OF A MEETING}	First- Start time of the meeting, Second-Names/Number of participants, Third- Duration of the meeting, Fourth- Place of the meeting
I want to create a meeting starting from {YOUR START TIME OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS} at {YOUR PLACE OF A MEETING} for {YOUR DURATION OF A MEETING}	First- Start time of the meeting, Second-Names/Number of participants, Third- Place of the meeting, Fourth- Duration of the meeting
I want to create a meeting starting from {YOUR START TIME OF A MEETING} for {YOUR DURATION OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS} at {YOUR PLACE OF A MEETING}	First- Start time of the meeting, Second- Duration of the meeting, Third- Names/Number of participants, Fourth- Place of the meeting
I want to create a meeting starting from {YOUR START TIME OF A MEETING} for {YOUR DURATION OF A MEETING} at {YOUR PLACE OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS}	First- Start time of the meeting, Second- Duration of the meeting, Third- Place of the meeting, Fourth- Names/Number of participants
I want to create a meeting starting from {YOUR START TIME OF A MEETING} at {YOUR PLACE OF A MEETING} for {YOUR DURATION OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS}	First- Start time of the meeting, Second- Place of the meeting, Third- Duration of the meeting, Fourth- Names/Number of participants
I want to create a meeting starting from {YOUR START TIME OF A MEETING} at {YOUR PLACE OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS} for {YOUR DURATION OF A MEETING}	First- Start time of the meeting, Second- Place of the meeting, Third- Names/Number of participants , Fourth- Duration of the meeting
I want to create a meeting for {YOUR DURATION OF A MEETING} starting from {YOUR START TIME OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS} at {YOUR PLACE OF A MEETING}	First- Duration of the meeting, Second- Start time of the meeting, Third- Names/Number of participants , Fourth- Place of the meeting
I want to create a meeting for {YOUR DURATION OF A MEETING} starting from {YOUR START TIME OF A MEETING} at {YOUR PLACE OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS}	First- Duration of the meeting, Second- Start time of the meeting, Third- Place of the meeting , Fourth- Names/Number of participants
I want to create a meeting for {YOUR DURATION OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS} starting from {YOUR START TIME OF A MEETING} at {YOUR PLACE OF A MEETING}	First- Duration of the meeting, Second- Names/Number of participants, Third- Start time of the meeting , Fourth- Place of the meeting
I want to create a meeting for {YOUR DURATION OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS} at {YOUR PLACE OF A MEETING} starting from {YOUR START TIME OF A MEETING}	First- Duration of the meeting, Second- Names/Number of participants, Third- Place of the meeting , Fourth- Start time of the meeting
I want to create a meeting for {YOUR DURATION OF A MEETING} at {YOUR PLACE OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS} starting from {YOUR START TIME OF A MEETING}	First- Duration of the meeting, Second- Place of the meeting, Third- Names/Number of participants , Fourth- Start time of the meeting
I want to create a meeting for {YOUR DURATION OF A MEETING} at {YOUR PLACE OF A MEETING} starting from {YOUR START TIME OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS}	First- Duration of the meeting, Second- Place of the meeting, Third- Start time of the meeting , Fourth- Names/Number of participants
I want to create a meeting at {YOUR PLACE OF A MEETING} for {YOUR DURATION OF A MEETING} starting from {YOUR START TIME OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS}	First- Place of the meeting, Second- Duration of the meeting, Third- Start time of the meeting , Fourth- Names/Number of participants
I want to create a meeting at {YOUR PLACE OF A MEETING} for {YOUR DURATION OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS} starting from {YOUR START TIME OF A MEETING}	First- Place of the meeting, Second- Duration of the meeting, Third- Names/Number of participants , Fourth- Start time of the meeting
I want to create a meeting at {YOUR PLACE OF A MEETING}starting from {YOUR START TIME OF A MEETING} for {YOUR DURATION OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS}	First- Place of the meeting, Second- Start time of the meeting , Third- Duration of the meeting , Fourth- Names/Number of participants
I want to create a meeting at {YOUR PLACE OF A MEETING} starting from {YOUR START TIME OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS} for {YOUR DURATION OF A MEETING}	First- Place of the meeting, Second- Start time of the meeting , Third- Names/Number of participants , Fourth- Duration of the meeting
I want to create a meeting at {YOUR PLACE OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS} starting from {YOUR START TIME OF A MEETING} for {YOUR DURATION OF A MEETING}	First- Place of the meeting, Second- Names/Number of participants , Third- Start time of the meeting , Fourth- Duration of the meeting
I want to create a meeting at {YOUR PLACE OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS} for {YOUR DURATION OF A MEETING} starting from {YOUR START TIME OF A MEETING}	First- Place of the meeting, Second- Names/Number of participants , Third- Duration of the meeting , Fourth- Start time of the meeting
I want to create a meeting starting from {YOUR START TIME OF A MEETING}, {YOUR DURATION OF A MEETING} at {YOUR PLACE OF A MEETING}	First- Start time of the meeting, Second- Duration of the meeting, Third- Place of the meeting
I want to create a meeting starting from {YOUR START TIME OF A MEETING}, at {YOUR PLACE OF A MEETING}, for {YOUR DURATION OF A MEETING}	First- Start time of the meeting, Second- Place of the meeting, Third- Duration of the meeting
I want to create a meeting for {YOUR DURATION OF A MEETING} starting from {YOUR START TIME OF A MEETING}, at {YOUR PLACE OF A MEETING}	First- Duration of the meeting, Second- Start time of the meeting, Third- Place of the meeting
I want to create a meeting for {YOUR DURATION OF A MEETING} at {YOUR PLACE OF A MEETING} starting from {YOUR START TIME OF A MEETING}	First- Duration of the meeting, Second- Place of the meeting, Third- Start time of the meeting
I want to create a meeting at {YOUR PLACE OF A MEETING} starting from {YOUR START TIME OF A MEETING} for {YOUR DURATION OF A MEETING}	First- Place of the meeting, Second- Start time of the meeting, Third- Duration of the meeting
I want to create a meeting at {YOUR PLACE OF A MEETING} for {YOUR DURATION OF A MEETING} starting from {YOUR START TIME OF A MEETING}	First- Place of the meeting, Second- Duration of the meeting, Third- Start time of the meeting
I want to create a meeting for {YOUR NAMES/NUMBER OF PARTICIPANTS} for {YOUR DURATION OF A MEETING} at {YOUR PLACE OF A MEETING}	First-Names/Number of participants, Second- Duration of the meeting, Third- Place of the meeting
I want to create a meeting for {YOUR NAMES/NUMBER OF PARTICIPANTS} at {YOUR PLACE OF A MEETING} for {YOUR DURATION OF A MEETING}	First-Names/Number of participants, Second- Place of the meeting, Third- Duration of the meeting
I want to create a meeting for {YOUR DURATION OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS} at {YOUR PLACE OF A MEETING}	First- Duration of the meeting, Second- Names/Number of participants, Third- Place of the meeting
I want to create a meeting for {YOUR DURATION OF A MEETING} at {YOUR PLACE OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS}	First- Duration of the meeting, Second- Place of the meeting, Third- Names/Number of participants
I want to create a meeting at {YOUR PLACE OF A MEETING} for {YOUR DURATION OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS}	First- Place of the meeting, Second- Duration of the meeting, Third- Names/Number of participants
I want to create a meeting at {YOUR PLACE OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS} for {YOUR DURATION OF A MEETING}	First- Place of the meeting, Second- Names/Number of participants , Third- Duration of the meeting
I want to create a meeting starting from {YOUR START TIME OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS} at {YOUR PLACE OF A MEETING}	First- Start time of the meeting, Second- Names/Number of participants , Third- Place of the meeting
I want to create a meeting starting from {YOUR START TIME OF A MEETING} at {YOUR PLACE OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS}	First- Start time of the meeting, Second- Place of the meeting , Third- Names/Number of participants
I want to create a meeting for {YOUR NAMES/NUMBER OF PARTICIPANTS} from {YOUR START TIME OF A MEETING} at {YOUR PLACE OF A MEETING}	First- Names/Number of participants, Second- Start time of the meeting , Third- Place of the meeting
I want to create a meeting for {YOUR NAMES/NUMBER OF PARTICIPANTS} at {YOUR PLACE OF A MEETING} starting from {YOUR START TIME OF A MEETING}	First- Names/Number of participants, Second- Place of the meeting , Third- Start time of the meeting
I want to create a meeting at {YOUR PLACE OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS} starting from {YOUR START TIME OF A MEETING}	First- Place of the meeting, Second- Names/Number of participants , Third- Start time of the meeting
I want to create a meeting at {YOUR PLACE OF A MEETING} starting from {YOUR START TIME OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS}	First- Place of the meeting, Second- Start time of the meeting , Third- Names/Number of participants
I want to create a meeting for {YOUR DURATION OF A MEETING} starting from {YOUR START TIME OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS}	First- Duration of the meeting, Second- Start time of the meeting , Third- Names/Number of participants
I want to create a meeting for {YOUR DURATION OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS} starting from {YOUR START TIME OF A MEETING}	First- Duration of the meeting, Second- Names/Number of participants , Third- Start time of the meeting
I want to create a meeting starting from {YOUR START TIME OF A MEETING} for {YOUR DURATION OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS}	First- Start time of the meeting , Second- Duration of the meeting , Third- Names/Number of participants
I want to create a meeting starting from {YOUR START TIME OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS} for {YOUR DURATION OF A MEETING}	First- Start time of the meeting , Second- Names/Number of participants , Third- Duration of the meeting
I want to create a meeting for {YOUR NAMES/NUMBER OF PARTICIPANTS} starting from {YOUR START TIME OF A MEETING} for {YOUR DURATION OF A MEETING}	First- Names/Number of participants , Second- Start time of the meeting , Third- Duration of the meeting
I want to create a meeting for {YOUR NAMES/NUMBER OF PARTICIPANTS} for {YOUR DURATION OF A MEETING} starting from {YOUR START TIME OF A MEETING}	First- Names/Number of participants , Second- Duration of the meeting , Third- Start time of the meeting
I want to create a meeting for {YOUR DURATION OF A MEETING} at {YOUR PLACE OF A MEETING}	First- Duration of the meeting, Second- Place of the meeting
I want to create a meeting at {YOUR PLACE OF A MEETING}, for {YOUR DURATION OF A MEETING}	First- Place of the meeting, Second- Duration of the meeting
I want to create a meeting starting from {YOUR START TIME OF A MEETING}, at {YOUR PLACE OF A MEETING}	First- Start time of the meeting, Second- Place of the meeting
I want to create a meeting at {YOUR PLACE OF A MEETING} starting from {YOUR START TIME OF A MEETING}	First- Place of the meeting, Second- Start time of the meeting
I want to create a meeting starting from {YOUR START TIME OF A MEETING} for {YOUR DURATION OF A MEETING}	First- Start time of the meeting, Second- Duration of the meeting
I want to create a meeting for {YOUR DURATION OF A MEETING} starting from {YOUR START TIME OF A MEETING}	First- Duration of the meeting, Second- Start time of the meeting
I want to create a meeting starting from {YOUR START TIME OF A MEETING}, for {YOUR NAMES/NUMBER OF PARTICIPANTS}	First- Start time of the meeting, Second- Names/Number of participants
I want to create a meeting for {YOUR NAMES/NUMBER OF PARTICIPANTS} starting from {YOUR START TIME OF A MEETING}	First- Names/Number of participants, Second- Start time of the meeting
I want to create a meeting for {YOUR NAMES/NUMBER OF PARTICIPANTS} for {YOUR DURATION OF A MEETING}	First- Names/Number of participants, Second- Duration of the meeting
I want to create a meeting for {YOUR DURATION OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS}	First- Duration of the meeting, Second- Names/Number of participants
I want to create a meeting for {YOUR NAMES/NUMBER OF PARTICIPANTS} at {YOUR PLACE OF A MEETING}	First- Names/Number of participants, Second- Place of the meeting
I want to create a meeting at {YOUR PLACE OF A MEETING} for {YOUR NAMES/NUMBER OF PARTICIPANTS}	First- Place of the meeting, Second- Names/Number of participants
I want to create a meeting for {YOUR NAMES/NUMBER OF PARTICIPANTS}	Required Entity- Names/Number of participants
I want to create a meeting starting from {YOUR START TIME OF A MEETING}	Required Entity- Start time of the meeting
I want to create a meeting for {YOUR DURATION OF A MEETING}	Required Entity- Duration of the meeting
I want to create a meeting at {YOUR PLACE OF A MEETING}	Required Entity- Place of the meeting

Figure A.4: Request Template for Meeting- Create meeting intent

5. Meeting- Modify meeting

Request Template	Sequence Order
I want to change/modify a meeting for {IF A CHANGE IN NAMES/NUMBER OF PARTICPANTS} starting from {IF A CHANGE IN START TIME OF A MEETING}, of {IF A CHANGE IN DURATION OF A MEETING} at {IF A CHANGE IN PLACE OF A MEETING}	First- Names/Number of participants, Second- Start time of the meeting, Third- Duration of the meeting, Fourth- Place of the meeting
I want to change/modify a meeting for {IF A CHANGE IN NAMES/NUMBER OF PARTICPANTS} starting from {IF A CHANGE IN START TIME OF A MEETING}, at {IF A CHANGE IN PLACE OF A MEETING}, for {IF A CHANGE IN DURATION OF A MEETING}	First- Names/Number of participants, Second- Start time of the meeting, Third- Place of the meeting, Fourth- Duration of the meeting
I want to change/modify a meeting for {IF A CHANGE IN NAMES/NUMBER OF PARTICPANTS} for {IF A CHANGE IN DURATION OF A MEETING} starting from {IF A CHANGE IN START TIME OF A MEETING}, at {IF A CHANGE IN PLACE OF A MEETING}	First- Names/Number of participants, Second- Duration of the meeting, Third- Start time of the meeting, Fourth- Place of the meeting
I want to change/modify a meeting for {IF A CHANGE IN NAMES/NUMBER OF PARTICPANTS} for {IF A CHANGE IN DURATION OF A MEETING} at {IF A CHANGE IN PLACE OF A MEETING} starting from {IF A CHANGE IN START TIME OF A MEETING}	First- Names/Number of participants, Second- Duration of the meeting, Third- Place of the meeting, Fourth- Start time of the meeting
I want to change/modify a meeting for {IF A CHANGE IN NAMES/NUMBER OF PARTICPANTS} at {IF A CHANGE IN PLACE OF A MEETING} starting from {IF A CHANGE IN START TIME OF A MEETING} for {IF A CHANGE IN DURATION OF A MEETING}	First- Names/Number of participants, Second- Place of the meeting, Third- Start time of the meeting, Fourth- Duration of the meeting
I want to change/modify a meeting for {IF A CHANGE IN NAMES/NUMBER OF PARTICPANTS} at {IF A CHANGE IN PLACE OF A MEETING} for {IF A CHANGE IN DURATION OF A MEETING} starting from {IF A CHANGE IN START TIME OF A MEETING}	First- Names/Number of participants, Second- Place of the meeting, Third- Duration of the meeting, Fourth- Start time of the meeting
I want to change/modify a meeting starting from {IF A CHANGE IN START TIME OF A MEETING} for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS} for {IF A CHANGE IN DURATION OF A MEETING} at {IF A CHANGE IN PLACE OF A MEETING}	First- Start time of the meeting, Second-Names/Number of participants, Third- Duration of the meeting, Fourth- Place of the meeting
I want to change/modify a meeting starting from {IF A CHANGE IN START TIME OF A MEETING} for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS} at {IF A CHANGE IN PLACE OF A MEETING} for {IF A CHANGE IN DURATION OF A MEETING}	First- Start time of the meeting, Second-Names/Number of participants, Third- Place of the meeting, Fourth- Duration of the meeting
I want to change/modify a meeting starting from {IF A CHANGE IN START TIME OF A MEETING} for {IF A CHANGE IN DURATION OF A MEETING} for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS} at {IF A CHANGE IN PLACE OF A MEETING}	First- Start time of the meeting, Second- Duration of the meeting, Third- Names/Number of participants, Fourth- Place of the meeting
I want to change/modify a meeting starting from {IF A CHANGE IN START TIME OF A MEETING} for {IF A CHANGE IN DURATION OF A MEETING} at {IF A CHANGE IN PLACE OF A MEETING} for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS}	First- Start time of the meeting, Second- Duration of the meeting, Third- Place of the meeting, Fourth- Names/Number of participants
I want to change/modify a meeting starting from {IF A CHANGE IN START TIME OF A MEETING} at {IF A CHANGE IN PLACE OF A MEETING} for {IF A CHANGE IN DURATION OF A MEETING} for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS}	First- Start time of the meeting, Second- Place of the meeting, Third- Duration of the meeting, Fourth- Names/Number of participants
I want to change/modify a meeting starting from {IF A CHANGE IN START TIME OF A MEETING} at {IF A CHANGE IN PLACE OF A MEETING} for {IF A CHANGE IN NAMES/NUMBER OF PARTICPANTS} for {IF A CHANGE IN DURATION OF A MEETING}	First- Start time of the meeting, Second- Place of the meeting, Third- Names/Number of participants , Fourth- Duration of the meeting
I want to change/modify a meeting for {IF A CHANGE IN DURATION OF A MEETING} starting from {IF A CHANGE IN START TIME OF A MEETING} for {IF A CHANGE IN NAMES/NUMBER OF PARTICPANTS} at {IF A CHANGE IN PLACE OF A MEETING}	First- Duration of the meeting, Second- Start time of the meeting, Third- Names/Number of participants , Fourth- Place of the meeting
I want to change/modify a meeting for {IF A CHANGE IN DURATION OF A MEETING} starting from {IF A CHANGE IN START TIME OF A MEETING} at {IF A CHANGE IN PLACE OF A MEETING} for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS}	First- Duration of the meeting, Second- Start time of the meeting, Third- Place of the meeting , Fourth- Names/Number of participants
I want to change/modify a meeting for {IF A CHANGE IN DURATION OF A MEETING} for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS} starting from {IF A CHANGE IN START TIME OF A MEETING} at {IF A CHANGE IN PLACE OF A MEETING}	First- Duration of the meeting, Second- Names/Number of participants, Third- Start time of the meeting , Fourth- Place of the meeting
I want to change/modify a meeting for {IF A CHANGE IN DURATION OF A MEETING} for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS} at {IF A CHANGE IN PLACE OF A MEETING} starting from {IF A CHANGE IN START TIME OF A MEETING}	First- Duration of the meeting, Second- Names/Number of participants, Third- Place of the meeting , Fourth- Start time of the meeting
I want to change/modify a meeting for {IF A CHANGE IN DURATION OF A MEETING} at {IF A CHANGE IN PLACE OF A MEETING} for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS} starting from {IF A CHANGE IN START TIME OF A MEETING}	First- Duration of the meeting, Second- Place of the meeting, Third- Names/Number of participants , Fourth- Start time of the meeting
I want to change/modify a meeting for {IF A CHANGE IN DURATION OF A MEETING} at {IF A CHANGE IN PLACE OF A MEETING} starting from {IF A CHANGE IN START TIME OF A MEETING} for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS}	First- Duration of the meeting, Second- Place of the meeting, Third- Start time of the meeting , Fourth- Names/Number of participants
I want to change/modify a meeting at {IF A CHANGE IN PLACE OF A MEETING} for {IF A CHANGE IN DURATION OF A MEETING} starting from {IF A CHANGE IN START TIME OF A MEETING} for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS}	First- Place of the meeting, Second- Duration of the meeting, Third- Start time of the meeting , Fourth- Names/Number of participants
I want to change/modify a meeting at {IF A CHANGE IN PLACE OF A MEETING} for {IF A CHANGE IN DURATION OF A MEETING} for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS} starting from {IF A CHANGE IN START TIME OF A MEETING}	First- Place of the meeting, Second- Duration of the meeting, Third- Names/Number of participants , Fourth- Start time of the meeting
I want to change/modify a meeting at {IF A CHANGE IN PLACE OF A MEETING}starting from {IF A CHANGE IN START TIME OF A MEETING} for {IF A CHANGE IN DURATION OF A MEETING} for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS}	First- Place of the meeting, Second- Start time of the meeting , Third- Duration of the meeting , Fourth- Names/Number of participants
I want to change/modify a meeting at {IF A CHANGE IN PLACE OF A MEETING} starting from {IF A CHANGE IN START TIME OF A MEETING} for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS} for {IF A CHANGE IN DURATION OF A MEETING}	First- Place of the meeting, Second- Start time of the meeting , Third- Names/Number of participants , Fourth- Duration of the meeting
I want to change/modify a meeting at {IF A CHANGE IN PLACE OF A MEETING} for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS} starting from {IF A CHANGE IN START TIME OF A MEETING} for {IF A CHANGE IN DURATION OF A MEETING}	First- Place of the meeting, Second- Names/Number of participants , Third- Start time of the meeting , Fourth- Duration of the meeting
I want to change/modify a meeting at {IF A CHANGE IN PLACE OF A MEETING} for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS} for {IF A CHANGE IN DURATION OF A MEETING} starting from {IF A CHANGE IN START TIME OF A MEETING}	First- Place of the meeting, Second- Names/Number of participants , Third- Duration of the meeting , Fourth- Start time of the meeting
I want to change/modify a meeting starting from {IF A CHANGE IN START TIME OF A MEETING}, {IF A CHANGE IN DURATION OF A MEETING} at {IF A CHANGE IN PLACE OF A MEETING}	First- Start time of the meeting, Second- Duration of the meeting, Third- Place of the meeting
I want to change/modify a meeting starting from {IF A CHANGE IN START TIME OF A MEETING}, at {IF A CHANGE IN PLACE OF A MEETING}, for {IF A CHANGE IN DURATION OF A MEETING}	First- Start time of the meeting, Second- Place of the meeting, Third- Duration of the meeting
I want to change/modify a meeting for {IF A CHANGE IN DURATION OF A MEETING} starting from {IF A CHANGE IN START TIME OF A MEETING}, at {IF A CHANGE IN PLACE OF A MEETING}	First- Duration of the meeting, Second- Start time of the meeting, Third- Place of the meeting
I want to change/modify a meeting for {IF A CHANGE IN DURATION OF A MEETING} at {IF A CHANGE IN PLACE OF A MEETING} starting from {IF A CHANGE IN START TIME OF A MEETING}	First- Duration of the meeting, Second- Place of the meeting, Third- Start time of the meeting
I want to change/modify a meeting at {IF A CHANGE IN PLACE OF A MEETING} starting from {IF A CHANGE IN START TIME OF A MEETING} for {IF A CHANGE IN DURATION OF A MEETING}	First- Place of the meeting, Second- Start time of the meeting, Third- Duration of the meeting
I want to change/modify a meeting at {IF A CHANGE IN PLACE OF A MEETING} for {IF A CHANGE IN DURATION OF A MEETING} starting from {IF A CHANGE IN START TIME OF A MEETING}	First- Place of the meeting, Second- Duration of the meeting, Third- Start time of the meeting
I want to change/modify a meeting for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS} for {IF A CHANGE IN DURATION OF A MEETING} at {IF A CHANGE IN PLACE OF A MEETING}	First-Names/Number of participants, Second- Duration of the meeting, Third- Place of the meeting
I want to change/modify a meeting for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS} at {IF A CHANGE IN PLACE OF A MEETING} for {IF A CHANGE IN DURATION OF A MEETING}	First-Names/Number of participants, Second- Place of the meeting, Third- Duration of the meeting
I want to change/modify a meeting for {IF A CHANGE IN DURATION OF A MEETING} for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS} at {IF A CHANGE IN PLACE OF A MEETING}	First- Duration of the meeting, Second- Names/Number of participants, Third- Place of the meeting
I want to change/modify a meeting at {IF A CHANGE IN PLACE OF A MEETING} at {IF A CHANGE IN PLACE OF A MEETING} for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS}	First- Place of the meeting, Second- Duration of the meeting, Third- Names/Number of participants
I want to change/modify a meeting for {IF A CHANGE IN DURATION OF A MEETING} at {IF A CHANGE IN PLACE OF A MEETING} starting from {IF A CHANGE IN START TIME OF A MEETING}	First- Duration of the meeting, Second- Place of the meeting, Third- Start time of the meeting
I want to change/modify a meeting at {IF A CHANGE IN PLACE OF A MEETING} starting from {IF A CHANGE IN START TIME OF A MEETING} for {IF A CHANGE IN DURATION OF A MEETING}	First- Place of the meeting, Second- Start time of the meeting, Third- Duration of the meeting
I want to change/modify a meeting at {IF A CHANGE IN PLACE OF A MEETING} for {IF A CHANGE IN DURATION OF A MEETING} starting from {IF A CHANGE IN START TIME OF A MEETING}	First- Place of the meeting, Second- Duration of the meeting, Third- Start time of the meeting
I want to change/modify a meeting for {IF A CHANGE IN DURATION OF A MEETING} at {IF A CHANGE IN PLACE OF A MEETING} starting from {IF A CHANGE IN START TIME OF A MEETING}	First- Duration of the meeting, Second- Place of the meeting, Third- Start time of the meeting
I want to change/modify a meeting for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS} for {IF A CHANGE IN DURATION OF A MEETING} at {IF A CHANGE IN PLACE OF A MEETING}	First-Names/Number of participants, Second- Duration of the meeting, Third- Place of the meeting
I want to change/modify a meeting for {IF A CHANGE IN NAMES/NUMBER OF PARTICPANTS} at {IF A CHANGE IN PLACE OF A MEETING} for {IF A CHANGE IN DURATION OF A MEETING}	First-Names/Number of participants, Second- Place of the meeting, Third- Duration of the meeting
I want to change/modify a meeting for {IF A CHANGE IN DURATION OF A MEETING} for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS} at {IF A CHANGE IN PLACE OF A MEETING}	First- Duration of the meeting, Second- Names/Number of participants, Third- Place of the meeting
I want to change/modify a meeting for {IF A CHANGE IN DURATION OF A MEETING} at {IF A CHANGE IN PLACE OF A MEETING} for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS}	First- Duration of the meeting, Second- Place of the meeting, Third- Names/Number of participants
I want to change/modify a meeting at {IF A CHANGE IN PLACE OF A MEETING} for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS}	First- Place of the meeting, Second- Names/Number of participants, Third- Duration of the meeting
I want to change/modify a meeting for {IF A CHANGE IN NAMES/NUMBER OF PARTICIPANTS}	Required Entity- Names/Number of participants
I want to change/modify a meeting starting from {IF A CHANGE IN START TIME OF A MEETING}	Required Entity- Start time of the meeting
I want to change/modify a meeting for {IF A CHANGE IN DURATION OF A MEETING}	Required Entity- Duration of the meeting
I want to change/modify a meeting at {IF A CHANGE IN PLACE OF A MEETING}	Required Entity- Place of the meeting

Figure A.5: Request Template for Meeting- Modify meeting intent

6. Software- Development information

Request Template	Sequence Order
How to develop a code in {LANGUAGE}, on {OPERATING SYSTEM} with {PACKAGE/LIBRARY/TOOL}?	First- Programming Language Second- Operating System, Third- Package/Library/Tool
How to develop a code in {LANGUAGE}, with {PACKAGE/LIBRARY/TOOL} on {OPERATING SYSTEM}?	First- Programming Language, Second- Package/Library/Tool, Third- Operating System
How to develop a code on {OPERATING SYSTEM}, in {LANGUAGE}, with {PACKAGE/LIBRARY/TOOL}?	First- Operating System, Second- Programming Language, Third- Package/Library/Tool
How to develop a code on {OPERATING SYSTEM} with {PACKAGE/LIBRARY/TOOL}, in {LANGUAGE}?	First- Operating System, Second-Package/Library/Tool, Third- Programming Language
How to develop a code with {PACKAGE/LIBRARY/TOOL} on {OPERATING SYSTEM}, in {LANGUAGE}?	First- Package/Library/Tool, Second- Operating System, Third- Programming Language
How to develop a code with {PACKAGE/LIBRARY/TOOL} in {LANGUAGE}, on {OPERATING SYSTEM}?	First- Package/Library/Tool, Second- Programming Language, Third- Operating System
How to develop a code in {LANGUAGE}, on {OPERATING SYSTEM}?	First- Programming Language, Second- Operating System
How to develop a code in {LANGUAGE} with {PACKAGE/LIBRARY/TOOL}?	First- Programming Language, Second- Package/Library/Tool
How to develop a code on {OPERATING SYSTEM}, in {LANGUAGE}?	First- Operating System, Second- Programming Language
How to develop a code on {OPERATING SYSTEM} with {PACKAGE/LIBRARY/TOOL}?	First- Operating System, Second- Package/Library/Tool
How to develop a code with {PACKAGE/LIBRARY/TOOL} on {OPERATING SYSTEM}?	First- Package/Library/Tool, Second- Operating System
How to develop a code with {PACKAGE/LIBRARY/TOOL} in {LANGUAGE}?	First- Package/Library/Tool, Second- Programming Language
How to develop a code in {LANGUAGE}?	Required Travel Entity- Programming Language
How to develop a code on {OPERATING SYSTEM}?	Required Travel Entity- Operating System
How to develop a code with {PACKAGE/LIBRARY/TOOL}?	Required Travel Entity- Package/Library/Tool

Figure A.6: Request Template for Software- Development information intent

7. Software- Perform action

Request Template	Sequence Order
I want to {INSTALL/DEPLOY/UPLOAD/DOWNLOAD ACTION} software on {OPERATING SYSTEM} which is of {SIZE}	First- Install/Deploy/Upload/Download Action Second- Operating System, Third- Size
I want to {INSTALL/DEPLOY/UPLOAD/DOWNLOAD ACTION} software, which is of {SIZE} on {OPERATING SYSTEM}	First- Install/Deploy/Upload/Download Action, Second- Size, Third- Operating System
On {OPERATING SYSTEM}, I want to {INSTALL/DEPLOY/UPLOAD/DOWNLOAD ACTION} software , with {SIZE}	First- Operating System, Second- Install/Deploy/Upload/Download Action, Third- Size
On {OPERATING SYSTEM}, with {SIZE}, I want to {INSTALL/DEPLOY/UPLOAD/DOWNLOAD ACTION} software	First- Operating System, Second-Size, Third- Install/Deploy/Upload/Download Action
Software of {SIZE} on {OPERATING SYSTEM}, I want to {INSTALL/DEPLOY/UPLOAD/DOWNLOAD ACTION}	First- Size, Second- Operating System, Third- Install/Deploy/Upload/Download Action
Software of {SIZE}, I want to {INSTALL/DEPLOY/UPLOAD/DOWNLOAD ACTION}, on {OPERATING SYSTEM}	First- Size, Second- Install/Deploy/Upload/Download Action, Third- Operating System
I want to {INSTALL/DEPLOY/UPLOAD/DOWNLOAD ACTION} software, on {OPERATING SYSTEM}	First- Install/Deploy/Upload/Download Action, Second- Operating System
I want to {INSTALL/DEPLOY/UPLOAD/DOWNLOAD ACTION} software which is of {SIZE}	First- Install/Deploy/Upload/Download Action, Second- Size
On {OPERATING SYSTEM}, I want to {INSTALL/DEPLOY/UPLOAD/DOWNLOAD ACTION} software	First- Operating System, Second- Install/Deploy/Upload/Download Action
On {OPERATING SYSTEM} with {SIZE}	First- Operating System, Second- Size
Software of {SIZE} on {OPERATING SYSTEM}	First- Size, Second- Operating System
Software of {SIZE}, I want to {INSTALL/DEPLOY/UPLOAD/DOWNLOAD ACTION}	First- Size, Second- Install/Deploy/Upload/Download Action
I want to {INSTALL/DEPLOY/UPLOAD/DOWNLOAD ACTION} software	Required Travel Entity- Install/Deploy/Upload/Download Action
Software on {OPERATING SYSTEM}	Required Travel Entity- Operating System
Software of {SIZE}	Required Travel Entity- Size

Figure A.7: Request Template for Software- Perform action intent

8. Software- Troubleshooting problem/error

Request Template	Sequence Order
I have an error {YOUR ERROR/PROBLEM} on {YOUR OPERATING SYSTEM} , for {YOUR PROGRAMMING LANGUAGE} with {PACKAGE/LIBRARY/TOOL USED}	First- Error/Problem, Second- Operating system, Third- Programming Language, Fourth- Package/Library/Tool Used
I have an error {YOUR ERROR/PROBLEM} on {YOUR OPERATING SYSTEM} , with {PACKAGE/LIBRARY/TOOL USED}, for {YOUR PROGRAMMING LANGUAGE}	First- Error/Problem, Second- Operating system, Third- Package/Library/Tool Used, Fourth- Programming Language
I have an error {YOUR ERROR/PROBLEM} for {YOUR PROGRAMMING LANGUAGE} on {YOUR OPERATING SYSTEM} , with {PACKAGE/LIBRARY/TOOL USED}	First- Error/Problem, Second- Programming Language, Third- Operating system, Fourth- Package/Library/Tool Used
I have an error {YOUR ERROR/PROBLEM} for {YOUR PROGRAMMING LANGUAGE} with {PACKAGE/LIBRARY/TOOL USED} on {YOUR OPERATING SYSTEM}	First- Error/Problem, Second- Programming Language, Third- Package/Library/Tool Used, Fourth- Operating system
I have an error {YOUR ERROR/PROBLEM} with {PACKAGE/LIBRARY/TOOL USED} on {YOUR OPERATING SYSTEM} for {YOUR PROGRAMMING LANGUAGE}	First- Error/Problem, Second- Package/Library/Tool Used, Third- Operating system, Fourth- Programming Language
I have an error {YOUR ERROR/PROBLEM} with {PACKAGE/LIBRARY/TOOL USED} for {YOUR PROGRAMMING LANGUAGE} on {YOUR OPERATING SYSTEM}	First- Error/Problem, Second- Package/Library/Tool Used, Third- Programming Language, Fourth- Operating system
I have an error on {YOUR OPERATING SYSTEM} for {YOUR ERROR/PROBLEM} for {YOUR PROGRAMMING LANGUAGE} with {PACKAGE/LIBRARY/TOOL USED}	First- Operating system, Second-Error/Problem, Third- Programming Language, Fourth- Package/Library/Tool Used
I have an error on {YOUR OPERATING SYSTEM} for {YOUR ERROR/PROBLEM} with {PACKAGE/LIBRARY/TOOL USED} for {YOUR PROGRAMMING LANGUAGE}	First- Operating system, Second-Error/Problem, Third- Package/Library/Tool Used, Fourth- Programming Language
I have an error on {YOUR OPERATING SYSTEM} for {YOUR PROGRAMMING LANGUAGE} for {YOUR ERROR/PROBLEM} with {PACKAGE/LIBRARY/TOOL USED}	First- Operating system, Second- Programming Language, Third- Error/Problem, Fourth- Package/Library/Tool Used
I have an error on {YOUR OPERATING SYSTEM} for {YOUR PROGRAMMING LANGUAGE} with {PACKAGE/LIBRARY/TOOL USED} for {YOUR ERROR/PROBLEM}	First- Operating system, Second- Programming Language, Third- Package/Library/Tool Used, Fourth- Error/Problem
I have an error on {YOUR OPERATING SYSTEM} with {PACKAGE/LIBRARY/TOOL USED} for {YOUR PROGRAMMING LANGUAGE} for {YOUR ERROR/PROBLEM}	First- Operating system, Second- Package/Library/Tool Used, Third- Programming Language, Fourth- Error/Problem
I have an error on {YOUR OPERATING SYSTEM} with {PACKAGE/LIBRARY/TOOL USED} for {YOUR ERROR/PROBLEM} for {YOUR PROGRAMMING LANGUAGE}	First- Operating system, Second- Package/Library/Tool Used, Third- Error/Problem , Fourth- Programming Language
I have an error for {YOUR PROGRAMMING LANGUAGE} on {YOUR OPERATING SYSTEM} for {YOUR ERROR/PROBLEM} with {PACKAGE/LIBRARY/TOOL USED}	First- Programming Language, Second- Operating system, Third- Error/Problem , Fourth- Package/Library/Tool Used
I have an error for {YOUR PROGRAMMING LANGUAGE} on {YOUR OPERATING SYSTEM} with {PACKAGE/LIBRARY/TOOL USED} for {YOUR ERROR/PROBLEM}	First- Programming Language, Second- Operating system, Third- Package/Library/Tool Used , Fourth- Error/Problem
I have an error for {YOUR PROGRAMMING LANGUAGE} with {PACKAGE/LIBRARY/TOOL USED} for {YOUR ERROR/PROBLEM} on {YOUR OPERATING SYSTEM}	First- Programming Language, Second- Error/Problem, Third- Operating system , Fourth- Package/Library/Tool Used
I have an error for {YOUR PROGRAMMING LANGUAGE} with {PACKAGE/LIBRARY/TOOL USED} for {YOUR ERROR/PROBLEM} on {YOUR OPERATING SYSTEM}	First- Programming Language, Second- Error/Problem, Third- Package/Library/Tool Used , Fourth- Operating system
I have an error for {YOUR PROGRAMMING LANGUAGE} with {PACKAGE/LIBRARY/TOOL USED} on {YOUR OPERATING SYSTEM} for {YOUR ERROR/PROBLEM}	First- Programming Language, Second- Package/Library/Tool Used, Third- Error/Problem , Fourth- Operating system
I have an error with {PACKAGE/LIBRARY/TOOL USED} for {YOUR PROGRAMMING LANGUAGE} on {YOUR OPERATING SYSTEM} for {YOUR ERROR/PROBLEM}	First- Package/Library/Tool Used, Second- Programming Language, Third- Operating system , Fourth- Error/Problem
I have an error with {PACKAGE/LIBRARY/TOOL USED} for {YOUR ERROR/PROBLEM} on {YOUR OPERATING SYSTEM} for {YOUR PROGRAMMING LANGUAGE}	First- Package/Library/Tool Used, Second- Programming Language, Third- Error/Problem , Fourth- Operating system
I have an error with {PACKAGE/LIBRARY/TOOL USED}on {YOUR OPERATING SYSTEM} for {YOUR PROGRAMMING LANGUAGE} for {YOUR ERROR/PROBLEM}	First- Package/Library/Tool Used, Second- Operating system , Third- Programming Language , Fourth- Error/Problem
I have an error with {PACKAGE/LIBRARY/TOOL USED} on {YOUR OPERATING SYSTEM} for {YOUR ERROR/PROBLEM} for {YOUR PROGRAMMING LANGUAGE}	First- Package/Library/Tool Used, Second- Operating system , Third- Error/Problem , Fourth- Programming Language
I have an error with {PACKAGE/LIBRARY/TOOL USED} for {YOUR ERROR/PROBLEM} on {YOUR OPERATING SYSTEM} for {YOUR PROGRAMMING LANGUAGE}	First- Package/Library/Tool Used, Second- Error/Problem , Third- Operating system , Fourth- Programming Language
I have an error with {PACKAGE/LIBRARY/TOOL USED} for {YOUR ERROR/PROBLEM} on {YOUR OPERATING SYSTEM} for {YOUR PROGRAMMING LANGUAGE} on {YOUR OPERATING SYSTEM}	First- Package/Library/Tool Used, Second- Error/Problem , Third- Programming Language , Fourth- Operating system
I have an error on {YOUR OPERATING SYSTEM} , {YOUR PROGRAMMING LANGUAGE} with {PACKAGE/LIBRARY/TOOL USED}	First- Operating system, Second- Programming Language, Third- Package/Library/Tool Used
I have an error on {YOUR OPERATING SYSTEM} , with {PACKAGE/LIBRARY/TOOL USED}, for {YOUR PROGRAMMING LANGUAGE}	First- Operating system, Second- Package/Library/Tool Used, Third- Programming Language
I have an error for {YOUR PROGRAMMING LANGUAGE} on {YOUR OPERATING SYSTEM} , with {PACKAGE/LIBRARY/TOOL USED}	First- Programming Language, Second- Operating system, Third- Package/Library/Tool Used
I have an error for {YOUR PROGRAMMING LANGUAGE} with {PACKAGE/LIBRARY/TOOL USED} on {YOUR OPERATING SYSTEM}	First- Programming Language, Second- Package/Library/Tool Used, Third- Operating system
I have an error with {PACKAGE/LIBRARY/TOOL USED} on {YOUR OPERATING SYSTEM} for {YOUR PROGRAMMING LANGUAGE}	First- Package/Library/Tool Used, Second- Operating system, Third- Programming Language
I have an error with {PACKAGE/LIBRARY/TOOL USED} for {YOUR ERROR/PROBLEM} on {YOUR OPERATING SYSTEM} for {YOUR PROGRAMMING LANGUAGE}	First- Package/Library/Tool Used, Second- Programming Language, Third- Operating system
I have an error for {YOUR ERROR/PROBLEM} for {YOUR PROGRAMMING LANGUAGE} with {PACKAGE/LIBRARY/TOOL USED}	First-Error/Problem, Second- Programming Language, Third- Package/Library/Tool Used
I have an error {YOUR ERROR/PROBLEM} with {PACKAGE/LIBRARY/TOOL USED} for {YOUR PROGRAMMING LANGUAGE}	First-Error/Problem, Second- Package/Library/Tool Used, Third- Programming Language
I have an error for {YOUR PROGRAMMING LANGUAGE} for {YOUR ERROR/PROBLEM} with {PACKAGE/LIBRARY/TOOL USED}	First- Programming Language, Second- Error/Problem, Third- Package/Library/Tool Used
I have an error for {YOUR PROGRAMMING LANGUAGE} with {PACKAGE/LIBRARY/TOOL USED} for {YOUR ERROR/PROBLEM}	First- Programming Language, Second- Package/Library/Tool Used, Third- Error/Problem
I have an error with {PACKAGE/LIBRARY/TOOL USED} for {YOUR ERROR/PROBLEM} on {YOUR OPERATING SYSTEM}	First- Package/Library/Tool Used, Second- Error/Problem , Third- Operating system
I have an error with {PACKAGE/LIBRARY/TOOL USED} on {YOUR OPERATING SYSTEM} for {YOUR ERROR/PROBLEM}	First- Package/Library/Tool Used, Second- Operating system , Third- Error/Problem
I have an error for {YOUR PROGRAMMING LANGUAGE} on {YOUR OPERATING SYSTEM} for {YOUR ERROR/PROBLEM}	First- Programming Language, Second- Operating system , Third- Error/Problem
I have an error for {YOUR PROGRAMMING LANGUAGE} for {YOUR ERROR/PROBLEM} on {YOUR OPERATING SYSTEM}	First- Programming Language, Second- Error/Problem , Third- Operating system
I have an error for {YOUR PROGRAMMING LANGUAGE} for {YOUR PROGRAMMING LANGUAGE} for {YOUR ERROR/PROBLEM}	First- Operating system , Second- Programming Language , Third- Error/Problem
I have an error for {YOUR PROGRAMMING LANGUAGE} for {YOUR ERROR/PROBLEM} for {YOUR PROGRAMMING LANGUAGE}	First- Operating system , Second- Error/Problem , Third- Programming Language
I have an error {YOUR ERROR/PROBLEM} on {YOUR OPERATING SYSTEM} for {YOUR PROGRAMMING LANGUAGE}	First- Error/Problem , Second- Operating system , Third- Programming Language
I have an error {YOUR ERROR/PROBLEM} for {YOUR PROGRAMMING LANGUAGE} on {YOUR OPERATING SYSTEM}	First- Error/Problem , Second- Programming Language , Third- Operating system
I have an error for {YOUR PROGRAMMING LANGUAGE} with {PACKAGE/LIBRARY/TOOL USED}	First- Programming Language, Second- Package/Library/Tool Used
I have an error with {PACKAGE/LIBRARY/TOOL USED}, for {YOUR PROGRAMMING LANGUAGE}	First- Package/Library/Tool Used, Second- Programming Language
I have an error on {YOUR OPERATING SYSTEM} , with {PACKAGE/LIBRARY/TOOL USED}	First- Operating system, Second- Package/Library/Tool Used
I have an error with {PACKAGE/LIBRARY/TOOL USED} on {YOUR OPERATING SYSTEM}	First- Package/Library/Tool Used, Second- Operating system
I have an error on {YOUR OPERATING SYSTEM} for {YOUR PROGRAMMING LANGUAGE}	First- Operating system, Second- Programming Language
I have an error for {YOUR PROGRAMMING LANGUAGE} on {YOUR OPERATING SYSTEM}	First- Programming Language, Second- Operating system
I have an error on {YOUR OPERATING SYSTEM} , for {YOUR ERROR/PROBLEM}	First- Operating system, Second- Error/Problem
I have an error {YOUR ERROR/PROBLEM} on {YOUR OPERATING SYSTEM}	First- Error/Problem, Second- Operating system
I have an error {YOUR ERROR/PROBLEM} for {YOUR PROGRAMMING LANGUAGE}	First- Error/Problem, Second- Programming Language
I have an error for {YOUR PROGRAMMING LANGUAGE} for {YOUR ERROR/PROBLEM}	First- Programming Language, Second- Error/Problem
I have an error {YOUR ERROR/PROBLEM} with {PACKAGE/LIBRARY/TOOL USED}	First- Error/Problem, Second- Package/Library/Tool Used
I have an error with {PACKAGE/LIBRARY/TOOL USED} for {YOUR ERROR/PROBLEM}	First- Package/Library/Tool Used, Second- Error/Problem
I have an error {YOUR ERROR/PROBLEM}	Required Entity- Error/Problem
I have an error on {YOUR OPERATING SYSTEM}	Required Entity- Operating system
I have an error for {YOUR PROGRAMMING LANGUAGE}	Required Entity- Programming Language
I have an error with {PACKAGE/LIBRARY/TOOL USED}	Required Entity- Package/Library/Tool Used

Figure A.8: Request Template for Software- Troubleshooting problem/error intent

A.3 Screen Shots

A.3.1 Entity Extraction Task

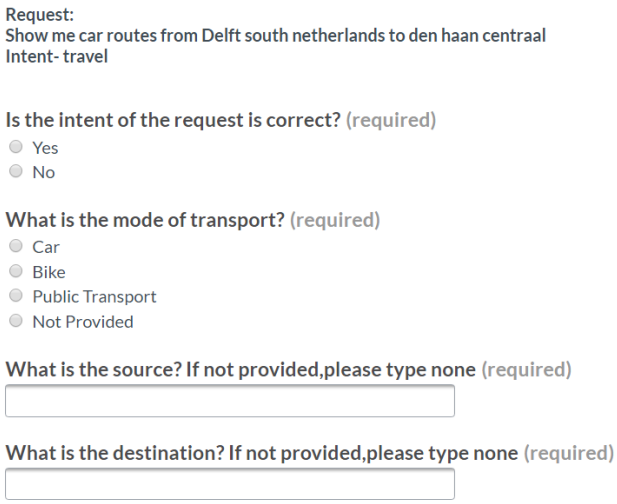


Figure A.9: Screen shot of Entity Extraction Task on CrowdFlower