Building an expert-system based conversational agent to provide personalised resources about neurological disorders

by

Tanvir Sajed

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Researchers developing artificially intelligent conversational agents (aka, chatbots) seek effective ways to provide personal assistance to users with various needs. We have implemented a web-based conversational agent that recommends resources to help clients (caregivers of patients suffering from various neurodevelopmental disorders — either family members or health-care professionals). Our agent first asks the client a conditional sequence of relevant questions about the patient's medical symptoms to fill out a patient profile, then recommends an appropriate website from a curated knowledge base of resources, previously validated by experts. We implement this agent as an expert-system (ES), using an inference engine to make recommendations based on a knowledge base of information about resources and the information acquired about the patient. This ES technology means the designer can easily update information about the resources, without redesigning the question-answering process. We anticipate that our framework can be used in different applications that require a conversational agent to ask personalised questions to identify the best recommendations for users.

Preface

The research conducted for this thesis forms part of a University of Alberta collaboration between Dr. Francois Bolduc from the Faculty of Medicine and Dentistry, Dr. Osmar Zaïane and Dr. Russell Greiner from the Department of Computing Science. Chapters 1, 3, and 4 will be revised for publication as a journal paper in the field of expert systems applications.

Acknowledgements

I sincerely express my deepest gratitude to my supervisors, Dr. Osmar Zaïane and Dr. Russell Greiner for their guidance and support throughout my Master's degree. They have supported me with valuable advice, insights and lessons that I acknowledge and carry-forward for the rest of my life. I am grateful for their kindness in my difficult situations, trust in my research and work, their constructive feedback on my thesis, and their genuine care for my success. I sincerely hope that my contributions have helped propagate their research in their respective fields.

I am also grateful to my co-supervisor, Dr. Francois Bolduc, for providing support and funding for the medical project that constitutes part of my thesis. I thank him for continuously providing me advice for the project, for his constructive feedback on my thesis, and helping the success of the project become a reality.

Finally, I am grateful to my friends and family for providing support during my difficult times and being there for me in my successes and failures.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Conversational agents (aka chatbots) are software systems that engage in dialogue exchanges with humans, often in natural language [69]. Many people use conversational agents like Google Home and Amazon Alexa for help with recommendations about shopping and general knowledge, setting reminders on their phone, accessing current information about weather, news and so on [50]. With the help of cutting-edge automated speech recognition algorithms, the tools analyse the users' speech and convert it into words; they then use sophisticated Natural Language tools to better interpret the information request and provide relevant responses [39]. They may use open-source knowledge bases such as Wikipedia to get the information needed to generate useful responses to user queries. These tools combine a variety of algorithms, including information retrieval systems [7], knowledge-based systems [32], and end-to-end machine-learned systems [75] to provide a useful service to users — a service that includes providing answers and recommending resources. However, they are unable to replicate longer, human-like conversations, as when friends engage in chitchat, and are only restricted to short task-oriented dialogues [44].

A chatbot system must address multiple tasks, each with a unique problem definition. For example, a recommendation chatbot that suggests buying the best laptops must address at least two tasks. One task is understanding the requirements of the user with regards to laptop specifications, i.e., whether the user wants a high-end laptop or a cheap laptop. This task involves natural language understanding [2] as the chatbot attempts to perceive the user's intent from spoken or written natural language input. Another task, known as natural language generation [71], is generating a written or spoken response appropriate to the user's intent. These tasks are fundamentally different with respect to their objective function. Researchers have used various approaches including rule-based systems, information retrieval systems, machine learning, and/or reinforcement learning in order to implement a complete conversational agent [13, 78, 89].

The first chatbot ELIZA was built by a team at MIT in 1966 [81, 96] to serve the role of a virtual psychotherapist. It receives user input, in written natural language, and breaks it down by rule-based keyword matching. Using another set of rules triggered by keywords determined from the input, ELIZA generates a response in written natural language. ELIZA has influenced research and development of multiple chatbot frameworks like ALICE and ELIZABETH [1, 79, 80] using open-source dataset corpora of human dialogue that can be used to train various models of chatbot. Using FAQs listed at the website of the School of Computer Science at the University of Leeds, researchers retrained the ALICE chatbot system to create FAQChat [78]. They claim that FAQChat is a viable alternative to Google as a tool to access open-source FAQ databases and generate answers in written natural language to written user questions. Like FAQChat and ELIZA, the chatbot proposed in our thesis is also rule-based with textual input and output. Rule-based chatbot frameworks, like ELIZABETH, allow knowledge from an expert to be hard-coded as rules. Alternatively, the rules may be learned using machine learning techniques from previous conversational data that include input from users and valid responses annotated by domain experts. Prior to the deep learning boom in the 2010s, research was restricted to rule-based chatbot systems due to the lack of large datasets and effective machine learning techniques. Recent research into chatbots led to open-ended systems that answer conversational questions [86], answer text comprehension questions [54], or engage in versatile conversations with users [89], all trained using variations of sequence-to-sequence deep-learning models [87] on very large collections of question and answers [68, 70].

Many chatbot applications do not have a large dataset of relevant training examples. In that case, developers resort to expert designed chatbots and template-based chatbots. A template-based chatbot attempts to fill out a template of information by asking the user questions followed by incorporating the response from the user in a template of answers. For example, a chatbot-doctor may ask a subject questions regarding medical symptoms. The answers are compiled together by a set of rules that provide responses to the subject in the form of a diagnosis. However, the application can be made to be more complex and dynamic if questions asked sequentially depend on the responses by the user. The order of questions asked may change depending on the sequence of answers provided by the subject. In order to incorporate this functionality, an old technology called expert-systems may be used to build the chatbot.

Expert-systems (ES) [36, 95] are programs that implement decision making using an inference engine that reaches decisions or conclusions given the facts about a particular instance. They were very popular in the 1970s and 80s, especially after the appearance of eMYCIN [12, 53], the first backward-chaining expert-system shell; they provided a means for non-programmers to easily use them. An example of a recent expert-system-based chatbot is a web-based pharmacist built by an undergraduate student at Ashesi University in Ghana [17]. Their chatbot answers questions in the domain of pharmacy, e.g., questions on information about drugs and prescriptions. It implements an expert-system as a web application like the chatbot proposed in our thesis. Most existing chatbot frameworks are either learning-based or rule-based without a generic inference engine, unlike the generic inference engine provided by an expert-system, and not compatible with expert-systems. The difficulty and cost of knowledge acquisition from domain experts to define the rules and ontologies, the knowledge base requires, has hindered this technology. With the advent of machine learning aiming at automatically learning rules from data when large labelled datasets are available, researchers are turning to this approach to build machine-learned systems, as this avoids the cost of knowledge acquisition from domain experts. However, access to large labelled data to automatically learn these rules is not always possible, making expert designed systems and expert-systems a viable solution to problems with availability of domain experts and limited data.

In this thesis, we research techniques of implementing a conversational agent using an expert-system framework. In order to implement a chatbot framework, we explored a medical application that attempts to provide relevant resources for parents or health care providers of subjects suffering from various neurological disorders and exhibiting challenging behaviours like physical aggression [16]. Online resources and technology-based interventions associated with neurological disorders are scarce and not organised in one single application [29]. We are developed an expert-system-based chatbot framework so that caregivers can interact with a conversational agent and find appropriate, personalised resources for subjects with neurological disorders.

1.1 Motivation

It is estimated that one billion people worldwide are suffering from various forms of neurodevelopmental disorder (NDD), which includes Autism, ADHD, and Intellectual Disability disorders [63]. NDD affects around 13% of the world population [63]. Local and international resources to help subjects with NDD are inadequate and unequally distributed [63]. These disorders are a serious concern especially in children who are unable to help themselves due to the inability to understand their symptoms. Caregivers are responsible to intervene and alleviate a situation that involves a child exhibiting physical aggression towards others due to a neurodevelopmental disorder, including Autism [64, 83]. These disorders lead to challenging behaviours such as physical aggression and self-harm when triggered by anxiety, depression, etc. Challenging behaviours can be managed by an intervention from a parent or a caregiver [52] and hence, it is important for the caregivers to be aware of effective interventions.

Online resources related to NDD are accessible by any web browser. However, there are no resources that offer advice on interventions or actions to take for specific cases of NDD and challenging behaviours. A chatbot that analyses specific subject characteristics and recommends a useful resource available locally or internationally can help parents and caregivers provide efficient service to children suffering from NDD. Our research into NDD and conversational agents led to the conclusion that a conversational agent that provides personalised recommendations for subjects of NDD does not exist.

Building and maintaining a chatbot to recommend resources is challenging as online resources are not static. The webpages are continuously changing; new webpages appear and others disappear or become obsolete. It is important that the chatbot need not be redesigned and the internal rules of the rule base need not be changed for each batch of new resources, or periodically when webpages are updated. We need to dissociate the expertise of recommending resources about NDD from the resources themselves, in order to have the resources maintained and updated in an external database by an individual without any programming skills.

Due to the lack of large datasets of existing dialogue between subjects and health practitioners recommending resources for NDD, we decided to implement an expert-system-based chatbot. Since the NDD resources are dynamic and continuously changing, we created a knowledge base of resources that is maintainable by an individual without any programming skills. We dissociated the information of the resources from the knowledge base of the expert-system so that the it does not end up with rules specific to resources. It also allowed others to modify resource attributes in the knowledge base without changing the rules. We needed to build an automated system like a chatbot, which can converse with the subject or the caregiver and offer resource recommendations since the availability of human support may not be enough to handle the large number of NDD subjects. In order to address these issues, we propose a flexible expert-system chatbot framework that responds to dynamic webpages without having to change the internal rules of the expert-system.

1.2 Thesis statements

Thesis 1. *We can use the proposed expert-system model to build a conversational agent that recommends online resources*

In order to demonstrate that we can build a conversational agent that recommends online resources by using the proposed expert-system model, we chose multiple resources from the resource database as test cases. We created an imaginary user with characteristics that match the attributes of the example resources. If the chatbot recommends these resources under the appropriate subject profile, it shows that the system matches the model it is built from. However, the system does not guarantee that the flow of dialog is efficient since we did not have the resources to conduct a research study on it.

Thesis 2. *The mechanism used by the expert-system-based chatbot is not affected by changes to the resources to recommend if these happen to change, disappear or new ones are created in the resource database as long as we are using the same characteristics of the resources*

In order to support the claim that the expert-system framework can easily add information about recommendations, we added the attributes of a relevant website to the system via the administration console, a web portal used by administrators to modify characteristics of resources. We tested whether the website was recommended given the right combination of subject facts – i.e. challenging behaviour, condition, the age of children, and where they lived. If the website is recommended via the user interface of the system, we see that information about the website was effectively incorporated without changing the rules of the expert-system.

When modifying information about an existing recommendation, we updated the website attributes from the administration console. We tested whether the same website was recommended under the right combination of website attributes. We performed similar steps for deleting information about a website from the resource database. We chose a website that was previously recommended by the expert-system. After removing the attributes of the website from the system via the administration console, we checked whether the system had stopped recommending the website under the right combination of subject facts.

1.3 Contributions

We have developed an expert-system-based chatbot framework that offers recommendations to caregivers of children suffering from various forms of NDDs. The recommendations are websites or resources vetted by a domain expert who is responsible for tagging each resources with appropriate attributes like Autism as its "condition", physical aggression as its "challenging behaviour", and anxiety as its "trigger". We have developed a front-end that acts as a messaging interface to the user. The interface asks appropriate questions to establish the subject profile, provides an appropriate recommendation matching the subject profile, and afterwards, allows the user to rate the recommendation. Moreover, our framework can be used by health practitioners seeking generic information about challenging behaviours and conditions associated with NDD. All the conversations between the user and the chatbot can be saved (with user consent) at a secured database, along with the user's feedback on the quality of website defined as a numerical rating between 1 and 5 (1 being a very bad resource and 5 being an outstanding resource). We anticipate the saved data can later be used to build a machine learning framework that ranks resources based on user feedback and provides better personalised recommendations over time.

In order for the system to recommend websites about interventions, an expert has collected a sample of useful websites. These websites are tagged with the relevant challenging behaviours, neurodevelopmental conditions and other relevant characteristics related to subjects. For example, if the website presents content about self-harm in children suffering from Autism, triggered by anxiety, the website will be tagged with "self-harm" for challenging behaviour, "anxiety" for trigger, and "autism" for condition. Some other resources (in addition to websites) collected by the expert are phone numbers that may be called in case of emergency, or support when a child is demonstrating a challenging behaviour.

The back-end is equipped with a knowledge base of expert-verified websites and their characteristics. Experts created some hand-made generic rules based on combinations of subject facts, and the system uses an inference engine to make decisions based on these rules and subject facts. The website is equipped with an administration console where an administrator can add, edit, or delete information about recommendations.

In the field of conversational agents [69], there are few surveys describing the scope of the field. We have performed an extensive review of Conversational Artificial Intelligence that we have covered in Chapter 2. We attempted to define the different aspects and features of conversational agents and describe how various methods fit the defined aspects. Generally, a chatbot needs to identify the context of a conversation, make use of the input from a user and implement a model to generate a valid response to the user's input that may be a question, answer or a comment.

To summarise, in this thesis our contributions are as follows :

1. *We have developed an expert-system-based framework for a conversational agent that recommends useful resources (including websites) to caregivers of subjects suffering from NDD and challenging behaviours.*

2. *We have conducted a review into the scope and features of conversational agents developed in academia and industry.*

3. *We have developed a user-friendly interface in the form of a website through which the chatbot can converse with the user and provide recommendations.*

4. *We have developed a database of relevant resources and websites for NDD subjects and labelled them with corresponding challenging behaviours, conditions, types of intervention, locations, etc.*

5. *We have developed a user-friendly admin console for administrators of the chatbot to easily add new, modify, or delete information related to re-*

sources.

6. *We have conducted a usability heuristic evaluation [59] of the interface to the conversational agent and performed a detailed analysis on the heuristics.*

1.4 Outline of thesis

In Chapter 2, we provide a detailed review of conversational agents in the literature and the industry. We define concepts such as the context, the input to the system, the response provided by the system; we then compare various conversational agents against these concepts. Then, we explain various methods used to build conversational agents and discuss different evaluation protocols for chatbots.

In Chapter 3, we define expert-systems and provide details of the method used to build the Coaching Assistant for Medical Information (CAMI). We outline different frameworks and software libraries used by our chatbot. We also define the schema of the resource database used to store specific characteristics about resources. The chapter defines the rule base used by the expert-system for inference. We also provide definitions for subject facts such as age, gender, challenging behaviour, location, etc.

In Chapter 4, we discuss the results of the heuristic evaluation of the interface of our chatbot and the steps taken to rectify some of the issues presented by the evaluation. We describe the results of the experiments to show that our expert-system framework matches the proposed system and is flexible with regards to adding, modifying, and deleting information about resources.

In Chapter 5, we conclude the thesis and discuss future improvements to the expert-system framework. We discuss how our system can be used for any recommendation based chatbot systems with defined website attributes and subject facts. The rules can be easily modified in the code and adapted to fit a particular application. We believe our expert-system framework is not just limited to applications in the medical domain, but in many different domains given that the scope of the application is properly defined.

Chapter 2

Conversational Agents

In this chapter, we discuss the current state-of-the-art research in conversational agents — especially Question Answering Systems [4, 35, 84] and Dialogue Systems [5, 51, 101] — then define some crucial aspects of conversation, and describe how different algorithms and methods have tried to tackle the problem of emulating a human conversation. We provide examples of the defined terms in Figure 2.1, which shows a conversation between a human and an AI agent. In the following sections, we define context, input, output, and model, the four fundamental dimensions of every conversational agent.

Definition 2.1. *The **input** comes from the user in the form of sentences, which are analysed by the conversational agent.*

Definition 2.2. *The **output** is the response created by the conversational agent given the input and the task.*

Definition 2.3. *The **method** is the process of generating an output by considering both the input and the task.*

Additionally, we introduce definitions of other terms used in this chapter :

Definition 2.4. *An **utterance** is a message sent from the user to the chatbot or vice versa.*

Definition 2.5. *An **interaction** is a single exchange of communication between the user and the chatbot agent that includes one of more utterances from the user and one or more utterances from the agent.*

Definition 2.6. *A* ***conversation*** *or a* ***dialogue*** *is a collection of one or more interactions between the user and the chatbot agent.*

Taking Figure 2.1 as reference, an example of input is "hello" and an example of output is "could you please let me know what are the operating systems you are using? ". An utterance can be both an input and an output. An interaction, a single exchange of communication, is the combined input and output — e.g. the first two lines of the excerpt. Here, the conversation is composed of multiple interactions between the user and the machine. The entire exchange can be termed either a conversation or a dialogue. A conversation is basically the exchange of interactions in a single session.

2.1 Conversation

A conversation is defined as a series of interactions between the agent and the user, where the user and the agent have passed information back and forth. The user may ask a question after which the agent may answer that question. Similarly, the user may answer a question after it is asked by the agent. A conversation may have one or many interactions between the user and the agent.

Example of an interaction (the simplest conversation):

User : Who is the prime minister of Canada?
Agent : Justin Trudeau

Example of an interaction with two utterances from the agent:

User : Who is the president of Canada?
Agent : Did you mean the prime minister of Canada?
Agent : Canada does not have any president. The prime minister of Canada is Justin Trudeau.

Machine: *hi*
Human: *hello*
Machine: *could you please let me know what are the operating systems you are using ?*
Human: *i am using linux*
Machine: *ok it was in the ssl vpn on which you stated it was broken*
Machine: *please check if you are logged in to corp network*
Machine: *what is the error that you are running please*
Human: *i am seeing an error related to vpn*
Machine: *what is the error message that you are getting when connecting to vpn using network connect ?*
Human: *connection refused or something like that*
Machine: *may i know the version of network connect you connect ?*
Human: *i am not sure i know that*
Machine: *is the network connect prompting for any user certificate*
Machine: *?*
Human: *yes*
Machine: *are you at home or at work at the moment ?*
Machine: *office ?*
Human: *i am at home*
Machine: *try this*
Machine: *goto <URL> page and enter your name and password*
Human: *done*
Machine: *try logging in with <NAME> and check if its working fine now*
Human: *yes , now it works !*
Machine: *great*
Machine: *anything else that i can help*
Human: *no , that was very useful*
Machine: *you are other than <NAME> :)*
Machine: *have a nice night*

(The *<URL>* indeed contains information about vpn access.)

Figure 2.1: An example excerpt of a conversation between a human and an agent [89].

A conversation always includes at least one interaction between the user and the agent. The example below includes more than one interaction between the user and the agent.

Example of a longer conversation:

User : Which country is Edmonton located in?
Agent : Canada
User : Which province in Canada is Edmonton located in?
Agent : Alberta

2.2 Context

Context is defined as the knowledge that is used by the agent to predict responses to a question asked by the user. Context may include information from the previous part of the current dialogue during the current conversation. It may also include the dialogue from previous conversations between the agent and the user. Context may originate externally from useful websites in the world wide web, such as Wikipedia. Typically, there are three types of context namely previous conversations, current conversation and external information.

External context can be specific in the sense that the source of knowledge is only included in a database. An example of a database is DBPedia [6], a collection of structured information from Wikipedia. DBPedia contains information from many different domains of knowledge, so an application using DBPedia as context may be open-ended. Sometimes, context may only be restricted to a group of documents. For example, a chatbot application may be responsible for providing information about ADHD. An expert may have collected a group of documents, including webpages, fact sheets, PDF documents, etc., only related to ADHD. The context for that application is then restricted to the information from the defined group of documents.

2.2.1 Domain

The domain is a feature of the application and its context that puts a limit to the context available to answer a particular question. If a chatbot includes context that covers information that is not restricted to a specific constrained domain, the application is open-domain. One can ask any question and expect a correct answer. The Google search engine [11] is an example of an open domain application where users can ask questions from different domains.

Closed-Domain

If the application has a limited number of documents or databases it considers context, and its scope is limited to answering questions within the domain of these documents and databases, the application is closed-domain. A chatbot is task-oriented and closed-domain if it only focuses on a particular task. Consider the task of booking a restaurant reservation. The answers to the questions are specific to the domain of restaurant booking. The agent picks the best answers/questions according to the context. If you ask the agent "Who is the president of United States?", it will not correctly answer the question since it falls outside the task of restaurant booking. It is not expected to answer any question outside its domain of expertise and the task it handles.

2.3 Input

Input is the data put forth by the user to communicate with the agent. It can be written as text, for example when writing questions in the Google search engine box. Input maybe auditory, for example, in the cases of Google Home and Amazon Alexa. With spoken input, there needs to be a microphone to record the speech of the user and send the signal to the agent. We categorize input into three different types: a question, an answer, or a comment. The input may be visual, such as images sent by the user via the chat interface.

Question

Typically in Question Answering Systems, the user only asks a question. Examples include “What is the current year”, “Who won the football world cup”, etc. The question does not necessarily follow with a question mark. The agent needs to analyse and recognise the question from the input. Understanding short text input or conversations is a difficult problem in itself and has been explored more in the literature compared to understanding longer text conversations [37, 77].

Answer

With a conversational agent such as a Dialogue System, the user is not restricted to asking a question. They are free to answer any questions asked by the agent. The agent may ask a relevant question in order to determine the situation of the user, for example asking about symptoms in the case of a patient’s medical diagnosis. The user provides an input to the agent as an answer to the question asked by the agent.

Comment

A comment may be anything that is not a question or an answer. For example, a user can start with an introductory greeting in the form of “hi” or “hello”. This initiates the conversation with an agent. In the case of an input comment, the agent does not need to answer any specific question. The agent can start the communication casually, like a normal conversation between people. The comment can be a goodbye such as “Great to talk to you. Bye”. It maybe used to compliment the service of an agent like “Thank you that helped a lot. You provide really good advice”. Alternatively, it can be used to provide negative feedback about the quality of the service, for example “You are an idiot”. Comments are used in chitchat conversations and they are explored in the paper by Vinyals et al. [89] which framed their conversational agent as a neural conversational model that uses a neural network architecture.

2.4 Output

Output is the data put forth by the agent to communicate with the user. It may take the form of written text when conversing within a mobile application or a web interface with text-based forms. It may take the form of spoken output in the case of devices with a speaker such as Google Home or Amazon Alexa. We categorise output as an answer to a question asked by the user, a question asked by the agent, or a comment that includes everything outside the realms of a question or an answer — any response from the agent to communicate with the user. The output is generally categorized into long text or short text. Most conversational agents respond in short text since responding in long text, an active topic of research, is more challenging [30, 76].

Question

An agent may resort to asking the user a question in order to understand their situation better. For example, in medical diagnosis, an agent needs to ask a series of questions in order to find out whether the patient is suffering from a variety of symptoms. It can be simple "yes" or "no" question, multiple choice questions, or questions that can be answered in free text.

Answer

Typical Question Answering Systems require the agent to only answer questions. The agent considers the context of the application in order to answer a question asked by the user.

Comment

Similar to the case of the user, an agent is also free to comment during the conversation with the user, if allowed in the context of the application. The agent may take the initiative and start with "hello" or "good day" and wait for the user's reply. The agent may also conclude the conversation with a "goodbye, have a good day".

2.5 Methods

The method of a chatbot is defined as a processing engine that analyses the context and the input together to provide an output. Typically, the method represents an ensemble of different models tuned for different tasks, for example, understanding and formulating a question or generating a response. Machine learning techniques — for example, deep learning — are applied on training data to build a learned model that is used for inference during response generation with real users [85]. Reinforcement learning may also be used for response generation [47]; a well defined state and action space is required for the agent. Information retrieval uses probabilistic techniques to infer an answer after the question is matched with documents from the knowledge base, which may represent the world wide web for open-ended applications [32].

2.5.1 Neural Conversational Model

Vinyals et al. [89] described the task of a conversational agent in detail using a neural conversational model. Their research combined natural language understanding and natural language generation in one simple neural network, a sequence-to-sequence model. Sequence-to-sequence models are recurrent neural networks used to predict a sequence of words given a previous sequence of words. Typically, a softmax or sigmoid final layer follows the final hidden layer into a vector of word embeddings, and the predicted word is the one with the highest probability. Words are predicted one at a time resulting in a sequence of words.

During training, a language model updates its weights by minimizing the differences between one-hot encoded actual words and their corresponding logits of the predicted words. However, these models do not work efficiently in the context of conversational AI and question answering. The hidden layers are not sophisticated enough to learn the context and facts from questions or conversations. Recent works using deep learning tend to have recurrent neural networks as a baseline over more sophisticated architectures like memory networks [85]. Figure 2.2 describes the sequence-to-sequence model implemented

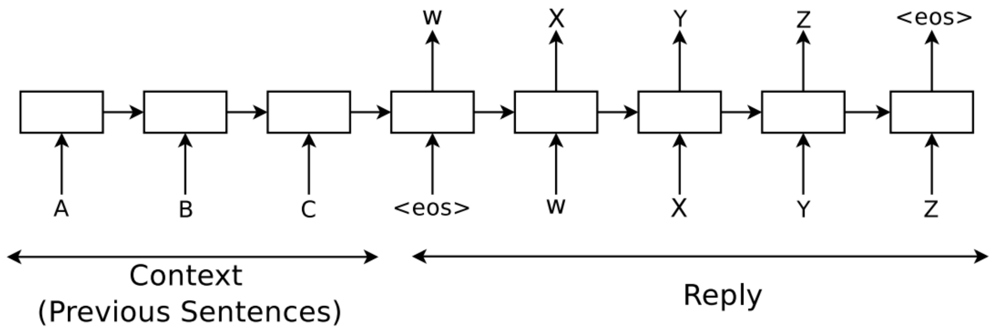


Figure 2.2: The sequence-to-sequence model used by Vinyals et al. [89]. Here, A, B and C are previous sentences considered as the context for the model. W, X, Y and Z are words from the sentence uttered by the user. These words interact with the model, during inference, to produce W, X, Y, Z as machine-predicted sequence of words as output.

by the authors and Figure 2.1 shows an example excerpt of a conversation between a human and the machine (agent).

The authors trained their model twice in two different experiments, one on a closed domain IT helpdesk troubleshooting dataset and another on an open domain movie-transcript dataset. The performance of their model was evaluated by human judges on a test dataset of 200 examples. Each of these test examples was evaluated by four different judges and if at least three judges agreed that the answer was of high quality, that test example was a success. They also ran CleverBot [26] on all of these 200 test examples and had the human judges evaluate CleverBot's responses. Their model was preferred over CleverBot on 97 of these test examples, while CleverBot was preferred over their model on 60 of the 200 test examples.

Comparison with CAMI

Our chatbot, CAMI, is architecturally very different from the neural conversation model. CAMI does not implement a machine learned framework but a hand-coded expert-system. CAMI queries a resource database to retrieve a relevant resource per patient facts. Unlike CAMI, there is no explicit database in the neural conversation model. Its context can be very open-ended, but

CAMI's is only restricted to a specific medical domain, NDD. CAMI and the neural conversation model are similar in the sense that both models can ask questions in the form of written natural language. CAMI also provides multiple choice questions to the user while the neural conversation model do not ask questions in a multiple choice format.

2.5.2 Memory Networks

Memory networks [10, 85, 98] are a class of end-to-end neural networks that produce responses from context and input. These networks can directly be applied in a question answering problem where the input is the question from the user and the output is an answer with the highest probability found from the final softmax classification layer of a known set of answers — a set of options rather than a sequence of words. The architecture has four major components. The first component encodes the input in a vector space of feature representation layer with learnable weights. The second component is the generalizable memory component that can be written to and read from. Input representations are saved to this component and updated with new data. The idea is to generalize, learn the most common input representations, and understand the context from the input. The third component is the output feature map layer which encodes an output feature representation vector, given the input and the current memory state.

End-to-end memory networks are architectures where all four components of memory networks are joined end-to-end in a multilayer neural network shown in Figure 2.3 [85]. The input sentences interact with a hidden layer of weights to create an embedding vector. The input sentences act like context for a given problem. The question, q, interacts with the input embedding in a layer where the inner product of the question and the embedding is calculated. The next layer is a softmax attention layer where the inner product is compared against the input embedding, and the most relevant sentences in the context are considered against the question. Some sentences from the context will have more weight than others when compared with the question. The weighted sum of the inputs is generated as output from the memory layer.

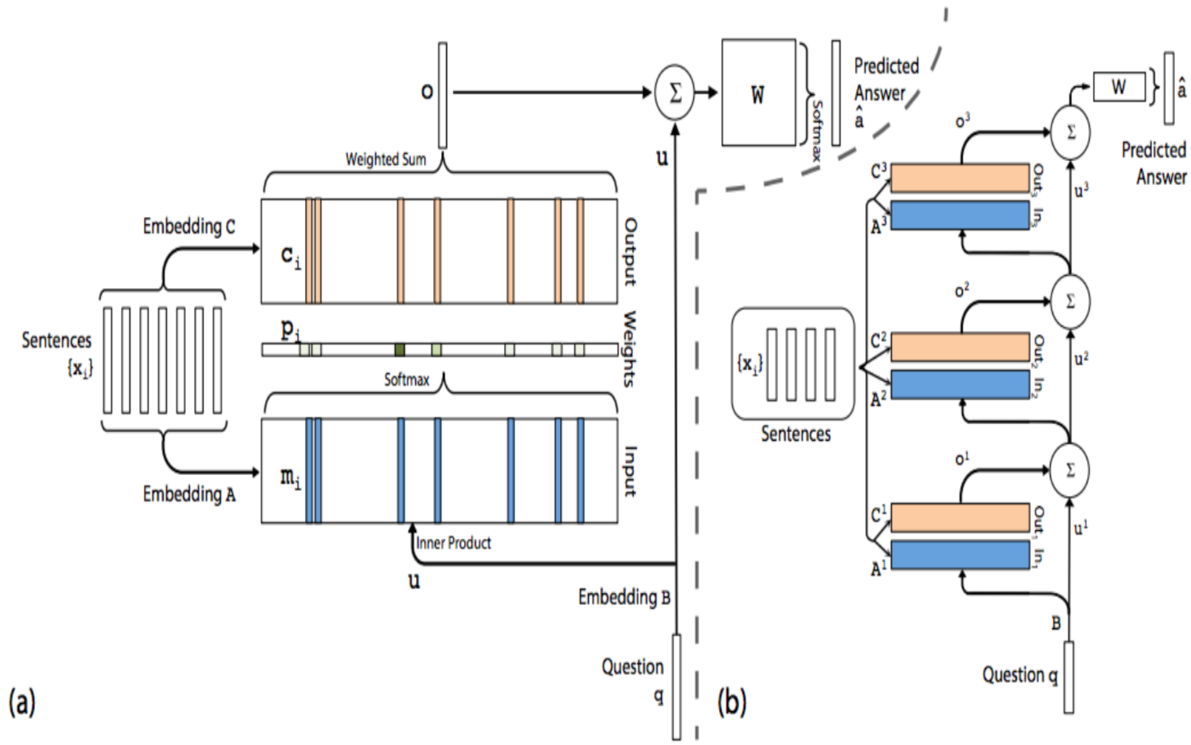


Figure 2.3: The architecture of a memory network as described by Sukhbaatar et al. [89]. Embeddings for question q interacts with embeddings for sentences x to create weighted sum o. The dot product of embedding of questions q and o is used in the final softmax layer with weights W to predict answer a from a list of known choices.

The output feature vector and the question vector interact in another layer to produce the predicted answer. This layer is connected to the final softmax classification layer of all the possible answers from the problem. Different papers introduced different variations of memory networks [49, 103] and submitted the results of experiments conducted on the Facebook BAbI datasets [97, 99].

In order to evaluate the models, the authors used accuracy as a performance measure calculating the percentage of answers the model got right. The answers need to be in exactly the right sequence of words when compared to the ground truth. The evaluation measure is known as exact match (EM), where the sequence of predicted words need to match the sequence of words for the ground truth.

Comparison with CAMI

CAMI is architecturally very different from memory networks. CAMI does not implement a machine learning framework but rather an expert-system for inference. CAMI uses a resource database to make queries in order to retrieve a relevant resource as per patient facts. Unlike CAMI, there is no database in memory networks. Their context can be very open-ended depending on the dataset used for learning, while CAMI's is only restricted to the medical domain of NDD. CAMI and memory networks are similar in the sense that both models can ask questions in the form of written natural language. However, CAMI also provides multiple choice questions to the user.

2.5.3 Information Retrieval

Information retrieval [7, 37, 72] is a class of algorithms that attempt to gather useful information from a data source by various means. The data source may be structured as in a database like MySQL, PostgreSQL, etc. It may also be unstructured, as in Wikipedia, where the text is the main source of information.

In a structured source, the database is defined by tables and columns that categorize data. For example, data about a user who has provided their height, weight, gender, and age is categorized into columns of height, weight, gender, and age. Consequently, it is very easy to execute an SQL query to a database in order to retrieve information about a user. Other structured information can be stored in XML or JSON format, and there have been many attempts to parse information from JSON and XML and retrieve relevant information [40].

However, in an unstructured source, the data is not classified into tables and columns. It may be free text like Wikipedia where the information about a user, for example a scientist, is written like "Albert Einstein was born on $\langle date, year \rangle$" . To retrieve such information, one needs to use natural language processing algorithms to extract named entities like the name of the person and the date when they were born [61]. Popular information retrieval algorithms, for instance the Google search engine [11], uses keyword matching to match words and sections of text between user queries and website content. The websites matched with order of confidence levels are returned back to the user with the first suggestion as the most confident suggestion. But the source of information does not always need to be free text. It may be encoded as a PDF file or an image that has information in pixels. In order to use a PDF file, we need to convert a PDF into free text and use natural language processing algorithms to extract information.

In the case of images, researchers need to implement image processing algorithms to extract information [27]. Researchers have implemented algorithms to extract shape, colour, texture, and other information available from images. A standard research problem is the classification of images into entities like humans, animals, books, laptops, etc. With recent advances in deep learning, the number of successful breakthroughs in image classification has grown exponentially, especially with deep convolutional networks on Imagenet dataset [42]. Researchers have also implemented speech processing algorithms for information retrieval from audio files [66], for converting audio information into subject-specific text-based vocabulary.

Yan et al. [104] implemented DocChat, an information retrieval system as part of a chatbot that can leverage information from unstructured documents. The chatbot retrieved sentences from these documents and ranked them in order of similarity with the user query — an utterance from the user that may be a question or a comment. It compared similarity between the query and the candidate sentences using multiple features like inverse document frequency of each word in a sentence. After calculating similarity between the query and candidate sentences, the chatbot ranked all the sentences in order of similarity. From the ranked list of sentences, it provided the sentence with the highest similarity as a response to the user query.

In order to evaluate DocChat, the authors used WikiQA dataset, and a list of [question, document, and answer] tuples. Each sentence from the document was either labelled with 0 if it was an inappropriate answer for the question or 1 if it was an appropriate answer. The task was to predict the correct answer labelled 1 and predict all other sentences as 0, for that specific question. They used evaluation metrics such as Mean average precision (MAP) and Mean reciprocal ranking (MRR) to provide a performance measure for their algorithm.

Another example of an information-retrieval-system-based chatbot is FAQChat [78], which uses keyword matching to fetch information from its knowledge base to answer user queries. The authors compared the performance of FAQChat with Google search engine. They built an interface where users can submit questions and receive answers from either Google or FAQChat. They chose their own formulated questions from a range of topics, and received two answers, one from Google and another from FAQChat. The user would then choose one of the answers as the best answer for the question. On conducting the experiment, the authors found that the mean number of users who chose answers from FAQChat over Google were higher.

Comparison with CAMI

The information retrieval chatbots described here are similar to CAMI as both use an external database to fetch answers to user queries. However, CAMI specifically uses an expert-system for inference unlike the chatbots described in this subsection. Their context can be open-ended or specific to a domain but CAMI's is only restricted to the medical domain of NDD. DocChat can ask questions in the form of written natural language like CAMI, while FAQChat cannot ask questions. Both FAQChat and DocChat can answer questions in text while CAMI is only able to retrieve resources as the only form of answer. CAMI is the only chatbot to provide multiple choice questions to the user.

2.5.4 Active Question Answering

With advances in deep reinforcement learning [56] and after the recent invention of Deep Q networks [55], scientists realised these techniques can also be applied to problems that involve decision making. Question answering and conversational AI are complex problems that involve decision making by the agent. It involves learning the best policy — the best question to ask, or the best answers to offer given the state of the agent and the question asked by the user. One can formulate this into a reinforcement learning problem where we can define state of the agent as the question asked by the user and the history of conversation, actions the agent can take such as asking a question or answering a question, and the rewards from the user depending on the action the agent took. Evaluation may be qualitative or quantitative. With respect to qualitative evaluation, human acceptance of an answer may be geared towards certain qualitative attributes — whether an agent's answer is good enough to be correct, or whether the answer makes logical sense but unrelated to the question [62]. Assessment may vary between users depending on their individual biases like education, expertise of the subject, etc. Learning from qualitative data is challenging and currently, it is an active research area [14].

C. Buck et al. [13] formulated the problem of question answering into a reinforcement learning task. The user interacts with an AI agent that communicates with a black box environment, external to the agent. The user provides a question to the AI agent that reformulates it into multiple questions. The agent then forwards the questions to the environment that provides an answer to each of these questions. The answer is matched against the original question to produce F1 score by comparing tokens between the answer and original question. The F1 score is the reward for that answer [21]. From the list of answers, the one with the highest F1 score is chosen as the best answer and ultimately the answer to the original question provided by the user. The authors of the paper built a complex architecture that involves deep learning techniques to influence the policy gradient of the AI agent.

The task of reformulating a question into multiple similar questions is done by a sequence-to-sequence model. The authors describe it as similar to the task of machine translation, which translates a text in one language to another [8]. Active Question Answering (AQA), introduced in C. Buck et al. [13], is a task to reproduce questions similar to the original question in the same language and elicit the best answer. The environment provides an answer to each of these questions and a reward once matched with the original question.

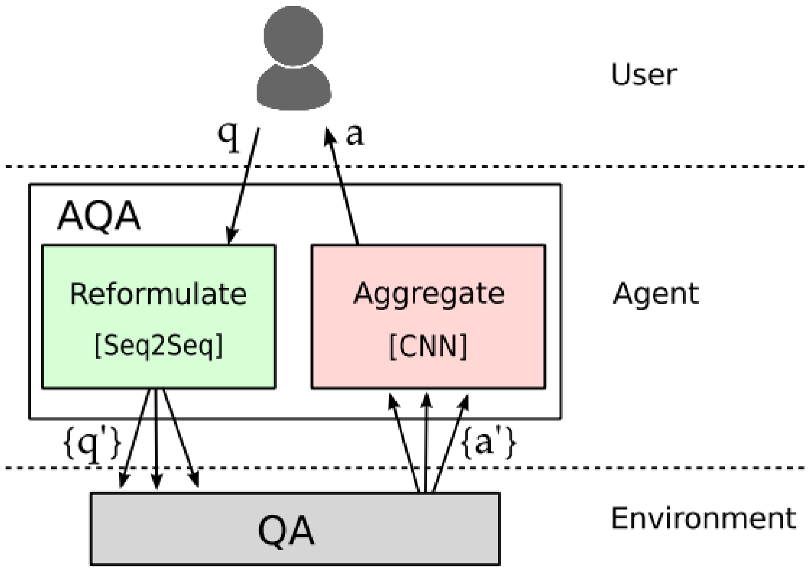


Figure 2.4: The architecture of the model, copied from C. Buck et al. [13], consists of the Reformulate network, sequence-to-sequence model, and the Aggretate network, a CNN model. Question q from the user is reformulated into multiple questions, answers to which are retrieved from the Environment. The best answer a is chosen by the Aggregate network and returned back to the user.

The authors considered this a reward function where the provided question is a state; the action is the task of producing similar questions to the original question; and the reward as the combined function of the action and the original question. The agent learns over time an optimal policy — generating the best questions given the original question — via reinforcement learning.

The task of choosing the best answer is done by a convolutional neural network that matches the question with the best answer based on the best F1 score. Eventually, the main aspect of the architecture is training with policy gradient involved in the sequence-to-sequence network of the question reformulation task. The agent learns, over time, how to train the reformulation network better to produce the best questions given the original question, and choose the best possible answer from the environment.

The authors trained their architecture on the Jeopardy! question and answer dataset called SearchQA [21]. Their model was evaluated on exact matches of answers (EM) and F1 scores. Their model beat the state-of-the-art model BiDAF, which is a deep neural network built for question answering [74]. An example of an original question is "Michael Caine & Steve Martin teamed up as Lawrence & Freddy, a couple of these, the title of a 1988 film". This question was reformulated by AQA into "What is name is name where name is name michael caine steve martin teamed lawrence freddy and title 1988 film key 2000 ?". The correct answer is "dirty rotten scoundrels".

Comparison with CAMI

CAMI is architecturally very different from AQA. CAMI does not implement a combined machine learning and reinforcement learning framework but rather an expert-system for inference. CAMI uses a resource database to make queries in order to retrieve a relevant resource given the characteristics of the user. Unlike CAMI, there is no explicit database for AQA. Its context can be very open-ended, but CAMI's is restricted only to the medical domain of NDD. CAMI can ask questions to user queries unlike AQA, which only answers questions (although it can reformulate questions in order to choose the best answer).

2.5.5 Knowledge-Based Systems

Knowledge-based systems [22] typically use structured information as the data source. This class of models attempts to process the input and make a query to a structured database to retrieve the answer. It is also possible that multiple answers can be retrieved and used to fill out an answer template. Knowledge-based systems can be used in conjunction with other systems, like information retrieval systems, in an ensemble system to make a method more effective and approach different situations.

In the Webclopedia architecture [32], a knowledge-based Question Answering System developed by Hermjakob et al., the authors explained several steps in order to produce an answer from a question from the user. During question parsing, they extracted useful information from the question that is relevant to the answer. Using the query creator engine, they extracted and combined useful words in order to create a meaningful query. Using the query and information retrieval techniques, they retrieved about 1000 important documents. The system then scored every sentences in these documents and used these scores to rank segments of sentences. They parsed top segments from these documents and matched them against answers stored in a database. Finally, they ranked and extracted possible answers and formatted them to be shown to the user.

The authors used the NIST TREC QA datasets [91, 92], about 1 million newspaper texts distributed by the NIST TREC, to select their answers to questions. They also used WordNet to formulate their questions. In order to evaluate their responses to the questions in the test set, they used Mean Reciprocal rank to provide a score for all their responses and rank them. If the correct response is found in the list of ranked answers at position n, the reciprocal rank for the question would be $1/n$. If the correct answer is not found in the ranked list of predicted answers, the reciprocal rank for that question will be 0. They then calculated the mean reciprocal rank over all the reciprocal ranks of the questions in the test set. Webclopedia tied for 2nd place in 2000 in the NIST TREC QA competition.

Comparison with CAMI

CAMI and Webclopedia are similar in their use of an external database to fetch answers to user queries. The answer that CAMI provides is a link to a resource whereas Webclopedia's answers are strings of text. CAMI specifically uses an expert-system for inference while Webclopedia uses information retrieval techniques to choose answers to user questions. Webclopedia is open-ended, but CAMI is restricted to the medical domain of NDD. Webclopedia replies in the form of written natural language like CAMI, but it cannot ask questions. CAMI can also ask multiple choice questions to the user.

2.5.6 Industry Chatbots

Google Home and Amazon Alexa

Google Home and Amazon Alexa [39, 50] are virtual assistants that help people, in general, complete a variety of everyday tasks like reminders on cooking, cleaning, doctor's appointments, etc. They are conversational agents, specifically Dialogue Systems, that converse in spoken language with the user [24, 45]. Their algorithms have been implemented in mobile phones too, not just in their own devices, using application programming interfaces connecting the software system with other devices. The Google assistant in phones accepts both text and speech as input for conversations with the user. For simplicity, we discuss only the Google Home and Amazon Alexa, the devices that only engage in spoken conversation with the user, in this thesis.

Google took their search engine to the next level with a Dialogue System capable of a short conversation consisting of multiple interactions with the user. By storing dialogue states of user and chatbot, Google Home can track how the current conversation with a user develops. This capability is also available with Amazon Alexa. They both have the ability to record conversations, with user permission, that provides previous conversation context for current conversations. Previous conversation contexts provide prior knowledge about the user that allows the devices to be more personalised and able to respond with better answers.

Very little is known about the exact architecture behind Google Home and Amazon Alexa. However, they are both open-ended chatbots that can perform various tasks for the user in addition to retrieving resources for the user. CAMI also provides resources for the user but its domain is restricted to the medical field of NDD. Like CAMI, they can both ask questions. However, Google Home and Amazon Alexa can only interact with the user via a spoken conversation unlike CAMI, which only interacts with texts.

GYANT

GYANT [90, 94] is a virtual assistant that offers medical advice to patients suffering from general medical conditions like fever, sore throat, etc. Like CAMI, it is meant to be used by medical professionals to support patient healthcare. Users are asked questions via the GYANT user interface about medical symptoms and they can either answer via text or by clicking one of the multiple choice options provided by the chatbot. The chatbot asks questions to complete a patient profile of medical symptoms and actions and interventions that the user has already taken on behalf of the patient. GYANT is very similar to CAMI in that it asks questions in order to fill out a patient profile. However, compared to CAMI, it covers a broader range of medical illness, is more versatile and engaging in its approach, asks more detailed questions, and can also answer some off-topic questions. It also lets the user write answers and ask any questions via a textbox at any time, while CAMI only lets the user answer in text for specific questions. With regards to medical advice, GYANT does not provide links to recommended websites. It also does not cover the range of medical conditions that CAMI covers with regards to NDD.

Buoy Health

Buoy Health [18, 41] is a medical chatbot that asks a patient or a health care professional about symptoms of illness and offers medical advice. Like CAMI, users are asked questions about symptoms while the chatbot completes a patient profile that it will later use to determine what advice to offer. The user interface is easy to use. The chatbot covers a wide range of common illnesses and asks a variety of questions to compile a detailed profile of the patient — much more detailed than CAMI. It also has a search engine for symptoms where users can search for relevant symptoms and ask specific questions about those symptoms to better understand them. Users answer multiple choice questions, but the chatbot does not allow users to write in free text and they are limited to the topic of conversation only, much like CAMI. With regards to medical advice, Buoy Health's chatbot does not provide website recommendations, unlike CAMI. It also does not cover the range of medical conditions that CAMI covers with regards to NDD.

2.6 Evaluation

Evaluating conversational agents is a challenging problem. For example, if the answer to "Who is the current Prime Minister of Canada" is "Justin Trudeau", one may write the answer in several different ways. An alternate answer may be, "It's Justin Trudeau for sure". Technically both answers are correct. However, it is difficult for an automated system to keep track of all the correct versions of an answer to a possible question. In this example, an algorithm matches the name "Justin" to an answer. However, if someone writes "It's not Justin, it's O'Toole", evaluating such an answer is also difficult. It is clearly a wrong answer. However, complex automated methods need to be implemented to analyse the word "not" beside "Justin" and give a score of zero, meaning an incorrect answer. Evaluating answers like "I do not know" is also challenging since it is neither right or wrong. Common measures of accuracy will not work in these cases where the answer predicted by the model is neither right or wrong.

The challenges to evaluating a chatbot lead us to human evaluators. There have been multiple studies where experts have been called on to evaluate an answer predicted by chatbots for domain-specific question answering. An expert decides whether the answer is good enough to be correct, and, based on human judgement, the evaluation of a conversational agent is performed. A measure of accuracy is used to evaluate the performance of the model based on the percentage of right answers in the entire test set. Human evaluation has been heavily implemented in literature for evaluating conversational agents in many different systems like SASSI [34], Trains [23], and PARADISE [93].

There are also some automated methods for evaluating a conversational agent. The Facebook BAbI tasks [99] provide a simple accuracy estimate of an answer provided by an agent. If the answer provided by an agent is correct to the word, meaning every word is correct and in the same sequence, the answer is deemed as a right answer and an exact match (EM). In any other circumstance, the answer is wrong. It is a strict measure of correctness. In summary, any classification metric may be used for evaluating a chatbot with the correct answer being accurate to the sentence with words that are positional independent. Any deviations from the correct answer, even when the answer seems similarly worded, is considered incorrect.

In order to find an evaluation metric to give credit to variations of correct answers, researchers at IBM developed an automated evaluation method called BLEU [65]. The method compares n-gram matches of the actual answer with the predicted answer. The matches are positional independent. The higher the matches between the predicted answer and the true answer, the higher its BLEU score. This method has primarily been used for machine translation. However, it has also been used for evaluation of conversational agents [88]. Meteor [9] is another automated evaluation method that focuses on comparing words between the actual answer and the predicted answer while also considering synonyms of words. The authors of Meteor claim they have improved it to identify the weakness of BLEU. While BLEU and Meteor are better than simply looking at a string as a correct answer, they do not address the difficulties of evaluating a conversational agent since differently worded sentences with limited n-gram matches may still carry similar meaning. Lie et al. [48] summarise reasons why the aforementioned metrics do not work well when evaluating open-ended conversational systems.

Jiwei et al. [46] developed neural persona conversational models in order to handle different personalities and speech patterns during response generation. They used perplexity and BLEU metrics in order to evaluate their models. The perplexity of a distribution is the weighted geometric average of the inverse of the probabilities. It is used to measure the quality of a language model with regards to the distribution of words in a generated response. Response generation models use perplexity as a qualitative measure of its performance. Typically, perplexity is not the sole evaluation metric of a conversational agent since it does not compare the generation with an actual answer.

In recent works, multiple evaluation metrics have been used to evaluate conversational agents. Venkatesh et al. [88] described a series of different computational metrics and later combined them for an evaluation model. The authors used the Alexa Prize dataset [3] to perform an evaluation of different models trained on that dataset. They used Alexa user ratings on millions of existing conversations between Amazon Alexa and users to train their evaluation model and mimic a human evaluator during the process. Their target is to research and develop upon an automated evaluation system for training and testing open-ended conversational agents.

2.7 Comparison of Conversational Agents

In this section we compare a few conversational agents introduced in this thesis against the dimensions of context, input and output. Table 2.1 compares different conversational agents under the features of the input dimension. Table 2.2 compares different conversational agents under the features of the output dimension. As shown in the two tables, these conversational agents only provide textual input and output. They, however, do not process visual and auditory input and output. Some industry chatbots, for example Amazon Alexa, do have the capability to process auditory input and output; they are not included in the tables due to the lack of academic papers confirming all the features discussed in the tables.

Tables 2.4 compare different conversational agents against multiple features of the context dimension. Similarly, in the case for the input and the output dimensions, the conversational agents shown in these tables only provide textual context.

2.8 Expert-system-based Chatbots

We looked at expert-system-based chatbots and found a few implementations in the literature. One such implementation came from Ashesi's undergraduate work [17] that combined expert-systems with a pharmaceutical chatbot. We noticed how they used pyknow to implement an expert-system using python.

Input	Visual	Textual	Auditory	Question	Answer	Comment
CAMI	✗	✓	✗	✗	✓	✗
Neural Conversational Model [89]	✗	✓	✗	✓	✓	✓
Memory Networks QA [85]	✗	✓	✗	✓	✗	✗
Memory Networks Dialogue [10]	✗	✓	✗	✓	✓	✓
Docchat [104]	✗	✓	✗	✓	✓	✓
AQA [13]	✗	✓	✗	✓	✗	✗
Webclopedia	✗	✓	✗	✓	✗	✗

Table 2.1: A few conversational agents categorized under the dimension of input. The table describes the format of the input available. For example, is the chatbot letting the user write words or speak to it or both? It also describes whether the user is able to question the chatbot, answer or provide a comment like “Thank you”.

We used experta [60], a modern fork of pyknow, to implement our expert-system since pyknow is now deprecated.

Aquabot [57] is a chatbot framework taking the role of a virtual assistant of a psychologist. Although the authors implemented rules in order to ask the right questions to a patient, their expert-system is not technically similar to CAMI's traditional expert-system that consists of a fact base, a knowledge base with rules and an inference engine. Their claimed expert-system consists of only rules from an expert, without the advantage of an inference engine and a fact base. Aquabot uses NLP methods to process text from the user; CAMI does not implement any advanced text processing. A similar chatbot for answering generic questions in healthcare is implemented by Kavitha et al. [38]. Similar to Aquabot and unlike CAMI, this chatbot implements advanced NLP methods for text processing from the user response and uses a rule-based expert-system.

Output	Visual	Textual	Auditory	Question	Answer	Comment
CAMI	✗	✓	✗	✓	✓	✗
Neural Conversational Model. [89]	✗	✓	✗	✓	✓	✓
Memory Networks QA [85]	✗	✓	✗	✗	✓	✗
Memory Networks Dialogue [85]	✗	✓	✗	✓	✓	✓
Docchat [104]	✗	✓	✗	✓	✓	✓
AQA [13]	✗	✓	✗	✗	✓	✗
Webclopedia	✗	✓	✗	✗	✓	✗

Table 2.2: A few conversational agents categorized under the dimension of output. The table describes the format of the output presented. It also describes whether the chatbot is able to question the user, answer or provide a comment like "Thank you".

The First Sinhala Chatbot [33] is another chatbot that is built upon a rule-based expert-system framework using Java SWI-PROLOG [100]. Its purpose is to answer generic questions, asked in the Sinhalese language, which may include about current date and time. Unlike CAMI, it uses advanced NLP methods to process and analyse questions in Sinhalese and uses rules from a knowledge base to create an answer in Sinhalese. Unlike CAMI, enough information is not provided about their expert-system architecture except that it has a knowledge base with rules. No explanation is provided on their inference engine.

Virtual Tour Agent [73], a chatbot that finds best cities with hotels and travel dates for people looking to book a tour, is similar to CAMI because it implements an expert-system like the one mentioned in this thesis complete with a knowledge base of rules and inference engine in the form of Java Expert System Shell [25]. Like CAMI, it also has a MySQL database which is used to

Summary	System Type	Source of Dataset	Year Published
CAMI	Expert-System-based	Internal Knowledge Base	NA
Neural Conversational Model [89]	Neural Network based	Curated Datasets	2015
Memory Networks QA [85]	Neural Network based	Facebook bABI QA Dataset	2015
Memory Networks Dialogue [85]	Neural Network based	Facebook bABI Dialog Dataset	2015
Docchat [104]	Information Retreival based	World Wide Web	2016
AQA [13]	Reinforcement Learning based	SearchQA Dataset	2017
Webclopedia	Knowledge based	Multiple curated datasets	2000

Table 2.3: A few conversational agents summarized describing the kind of system, the source of dataset and the year published.

retrieve tour information. In a webpage, the user is asked to choose potential tour dates, hotel type, amenities and price ranges for potential hotels. CAMI also asks similar questions related to patient's medical conditions. However, unlike CAMI, Virtual Tour Agent is not a chatbot as it does not participate in a sequence of back to forth interactions with the user.

In this chapter, we describe a study of the characteristics of modern chatbots found in the literature and the industry. The design of template-based chatbots like Buoy Health motivated the design of CAMI, especially the fact that template-based chatbots use question templates to ask questions to the user in order to build a subject profile that can be later used to provide recommendations to the user. We also looked at how Buoy Health uses multiple choice answers to avoid ambiguity in the answers provided by the users if

Context Domain, Format	Closed Domain	Open Domain	Visual	Textual	Auditory
CAMI	✓	✗	✗	✓	✗
Neural Conversational Model [89]	✓	✓	✗	✓	✗
Memory Networks QA [85]	✓	✗	✗	✓	✗
Memory Networks Dialogue [85]	✓	✗	✗	✓	✗
Docchat [104]	✗	✓	✗	✓	✗
AQA [13]	✓	✗	✗	✓	✗
Webclopedia	✗	✓	✗	✓	✗

Table 2.4: A few conversational agents categorized under the dimension of context. This table describes whether the domain of the conversational agents is open or closed domain and in what format the context is available.

they used text instead, and we implemented the same for CAMI. However, learning-based chatbots did not influence the design of CAMI since there were no available conversational datasets to learn from.

Chapter 3

Method

In this chapter we briefly explain the concepts behind expert-systems followed by defining the architecture behind CAMI. We then describe how different components of CAMI interact with each other including the expert-system and Question Answering System, and explain the core algorithm in detail with examples. We also describe the design of the resource database and the interface of the chatbot. Unlike conversational agents that use end-to-end models of learning based on sequence-to-sequence predictors to generate a response, and unlike typical domain specific Question Answering Systems that use hard-coded rules and a set of templates with a slot filling process to generate answers, our chatbot, CAMI (Coaching Assistant for Medical Information), uses an expert-system framework.

3.1 Definitions

In this section, we define the most important terms used for this thesis. The definitions introduced in this section are more specific to CAMI wheras those in Chapter 2 are more generic.

Definition 3.1. *An **agent** (chatbot) is the artificial conversational system responsible for communicating with the user via an textual interface by asking questions to the user and finally offering a recommendation.*

Definition 3.2. *A **user** is defined as a person who communicates with a chatbot via a textual interface. In our project, it is either a parent of the* *subject suffering from a neurodevelopmental disorder or a professional health practitioner responsible for a specific subject.*

Definition 3.3. *A **subject** is defined as a person with relevant medical facts for whom a recommendation will be offered by the chatbot. S/He can be a child, a teenager, or an adult.*

Definition 3.4. *A **resource** is a website with information related to NDD for parents, or health care providers who are seeking help on behalf of a subject suffering from NDD.*

Definition 3.5. *A **trigger** is a temporal state of the subject that may include anxiety, depression, etc.*

Definition 3.6. *A **challenging behaviour** is a subject behaviour demonstrated when a subject is suffering from a neurodevelopmental disorder aggravated by a trigger. This includes self-harm, physical aggression, non-compliance, and/or temper tantrums.*

Definition 3.7. *An **intervention** is an action taken by a parent or a health practitioner to help the subject suffering from a challenging behaviour.*

Definition 3.8. *The **resource database** is a structured database for resources containing background information initially asserted by a domain expert and attributes that can be related to subject facts*

3.2 Expert-Systems

In the field of artificial intelligence, expert-systems are computer programs that encode knowledge within a set of rules, and implement an inference engine to reach decisions or conclusions given a set of facts from a fact base [36, 95]. Expert-systems are task-oriented and domain-specific. The rules encoded within an expert-system is decided by experts of the domain of application. Expert-systems, although an old technology, is still used in many different applications including, but not limited to, manufacturing planning [82], medical diagnosis [31], robotics [43], etc. These systems have three components are defined as follows:

Definition 3.9. *The **knowledge base** consists of a set of rules representing the expertise of the expert-system.*

Definition 3.10. *The **fact base** stores information about the environment that will be used by the rules.*

Definition 3.11. *The **inference engine** is a program based on logic that deduces consequent facts and reaches conclusions based on rules from the knowledge base, and established facts from the fact base.*

A generic structure of an expert-system connecting the knowledge base, the fact base and the inference engine together is shown in Figure 3.1. Information is only one way as it is passed from the knowledge base to the inference engine when rules are processed by the inference engine. However, the information travels both ways between the fact base and inference engine. When the inference engine makes conclusions that can be used as facts for rules that are chained, the inference engine passes these facts to the fact base [95].

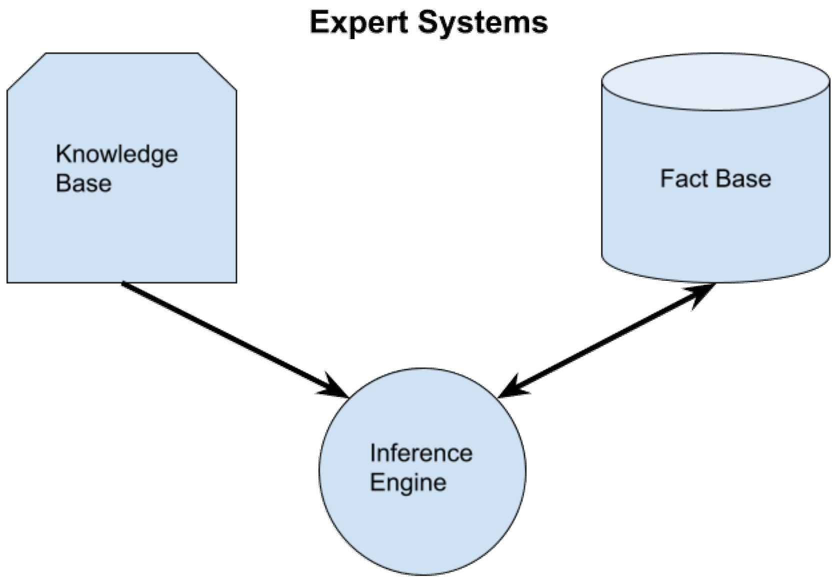


Figure 3.1: A generic structure of an expert-system. The knowledge base and the fact base are connected by the inference engine, which makes decisions based on rules in the knowledge base and facts in the fact base [95].

3.2.1 Knowledge Base

The knowledge base consists of a set of rules representing the expertise of the expert-system while the fact base stores temporary information about information related to the application. The rules of the knowledge base are developed by an expert who has domain knowledge about the application. Each rule in the knowledge base has an antecedent — the condition of the rule, and a consequent — the conclusion of the rule. For example, a rule may be "If it is cloudy today, then it will rain tomorrow". In this case the antecedent is the condition whether it is cloudy today and the consequent is the deduced fact that it will rain tomorrow. The antecedent is expressed with facts potentially stored in the fact base. When the antecedent is satisfied — i.e. there is a fact in the fact base asserting that the weather is cloudy, the consequent is asserted to be true and facts from the consequent are inserted into the fact base to update current information that "it will rain tomorrow".

3.2.2 Inference Engine

The inference engine is a program based on logic that deduces consequent facts and reaches conclusions based on rules from the knowledge base, and established facts from the fact base. It may update the fact base with new facts that are deduced by applying rules to the established facts. The two most common forms of reasoning in an inference engine are forward chaining and backward chaining. Forward chaining is a method of reasoning that reaches conclusion facts by searching all the rules of the knowledge base to check if each of their antecedent is satisfied. Backward chaining works in the opposite direction. It checks whether it can establish the consequent of a rule, the goal, by checking whether the antecedents of that rule and any other chained rules are satisfied. The inference engine goes backward from the goal to establish the consequent of connected rules before establish the goal itself. In complex expert-systems, rules are chained together to create a DAG-like structure. They may lead to deduced facts that are used in chained rules to derive conclusive facts. Some rules may connect user-provided facts with deduced facts in antecedent.

Figure 3.2 provides an example of an application where the rules are chained together. User-provided facts "Temperature Between -4 C and 0 C" and "Freezing Rain = True" are joined in conjunction using an "AND" relation, meaning they both must be True to satisfy the conclusion that the "chance of black ice forming on the roads and pathways" is high. The conclusion of the previous rule provides the fact "Black Ice = High". This consequent of the first rule, now a derived fact, acts as antecedent of the second rule in conjunction with another user-provided fact "Car = False", which means the user does not own a car, leading to the decision to work from home today because there is a chance that someone may fall on black ice while walking, and get injured.

By forward chaining, the expert-system deduces first whether the first rule is satisfied leading to the conclusion that the chance of black ice is high. Chaining the previous rule with the next, it can deduce whether to work from home. By backward chaining, the inference engine tries to establish whether to work from home by looking at the final rule and establish whether the user already has a car and whether the chance of black ice is high. Since the chance of black ice is determined by the first rule, it will go back to the first rule and establish the facts for temperature and freezing rain. The inference engine will look to establish facts that are needed to deduce whether the user needs to work from home by going backward down the chained rules.

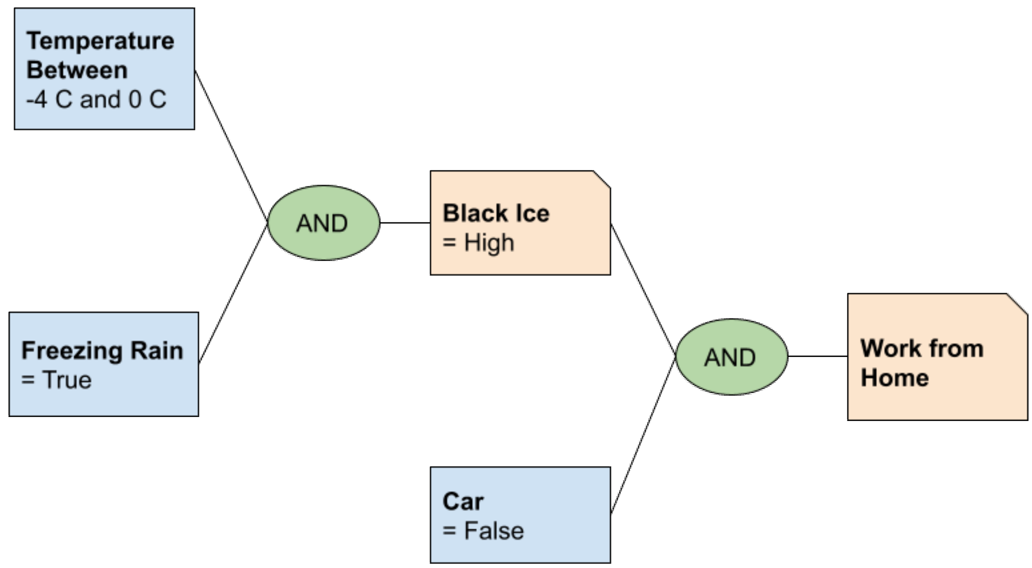


Figure 3.2: An example set of chained rules in an application that recommends whether to work from home after freezing rain. The circular nodes represent connections, for example "AND" or "OR" between facts in the antecedent. The rectangular nodes, colored light blue, represent facts provided by the user. The light orange rectangular nodes represent conclusive facts derived by the expert-system. In this example, the conclusive fact "Black Ice = High" is deduced by the expert-system to be later used as a deduced fact for the next rule. The decision to work from home is also a conclusive fact for the second rule.

3.3 CAMI Expert-system design and implementation

In this and the following sections, we discuss the architecture of CAMI. Figure 3.3 shows the diagram of the architecture of the CAMI chatbot system. The framework is divided into an expert-system and a Question Answering System connected to a resource database. The Question Answering System collects facts from the user and passes them to the expert-system. When an antecedent in a rule is satisfied and the rule is fired, the expert-system formulates a query and passes it to the Question Answering System. The Question Answering System executes the query to the resource database in order to retrieve relevant resources and passes them to the user via the user interface. Figure 3.4 shows how the user communicates with CAMI. We decided to implement an architecture that is compartmentalized and divided into core components like the expert-system and the Question Answering System. It is motivated by the need to decouple the rules in the expert-system from the resources in the resource database so that the information about the resources can be independently modified without requiring to change the rules in the expert-system.

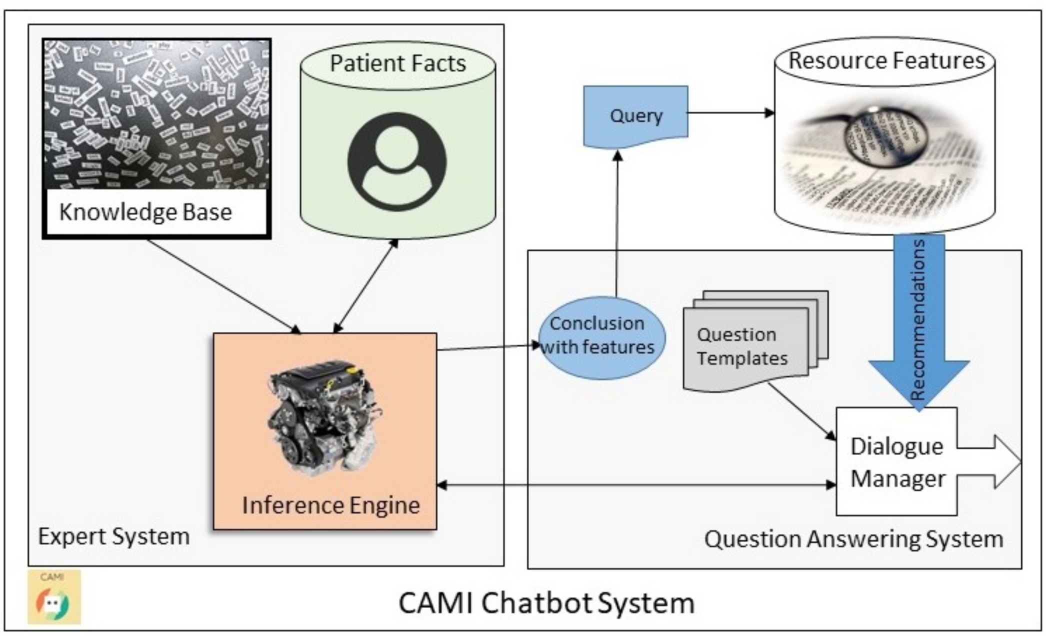


Figure 3.3: CAMI uses an expert-system, a resource database, which stores resource facts in the form of attributes of resources, to recommend resources, and a Dialogue Manager with question templates. The Question Answering System asks a question, retrieves a subject fact for that question, and passes it to the expert-system, which fires rules to create a query for the resource database.

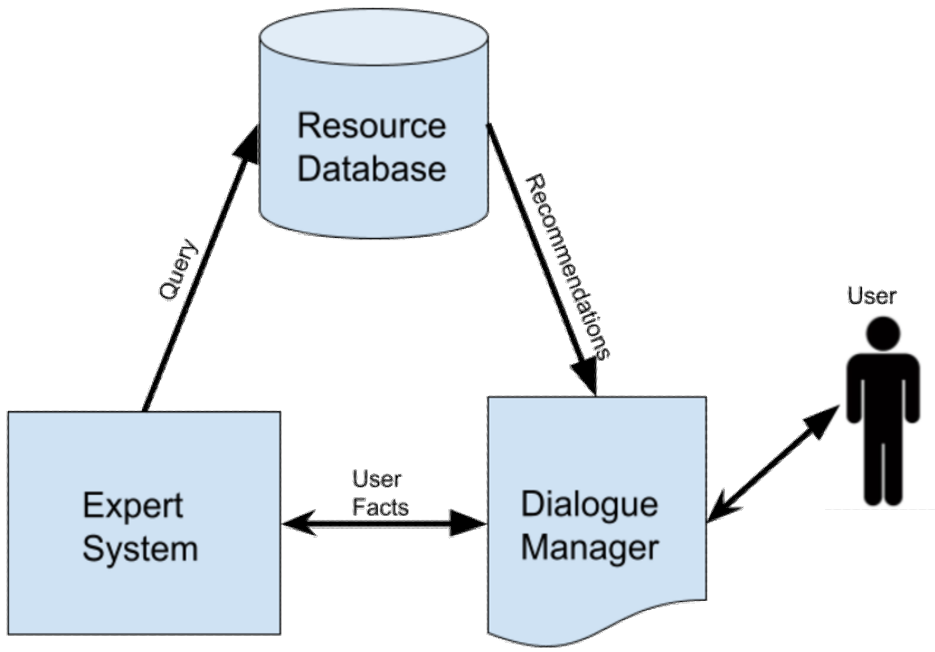


Figure 3.4: The relationship between different components of our chatbot, CAMI. It depicts how the resource database, which stores resource facts in the form of attributes of resources, is separated from the expert-system only accessible via a query when a rule is fired in the expert-system. This allows the research team to change information about resources in the resource database without modifying the expert-system.

In CAMI, the expertise of the NDD therapists and advisors is encoded within the knowledge base in the form of rules, and used when recommending resources to users about subjects. The inference engine applies the rules from the knowledge base and subject facts from the fact base in order to reach a conclusion that creates an SQL query that depends on subject facts. The query is then passed to the Question Answering System to extract relevant resources from the resource database. If no rules are satisfied, the Dialogue Manager will select a question for a missing subject fact to ask the user, e.g. "What is the age of the subject". Currently, CAMI can process five different types of subject facts as literals of a rule's conjunctive antecedent. They are challenging behaviour, condition, age, intervention and location.

Once a rule having the antecedent with established subject facts is satisfied, the rule is saved in the temporary session variable, unique for every session, to provide an explanation of the recommendation if required. An advantage of using expert-systems is that they can provide explanations to decisions made by the inference engine. We provide explanation of the decision made in terms of all the subject facts in the saved rule that matched with the resource attributes. For example, if a recommended resource has two attributes, city as “Edmonton” and age of subject as “Child”, these attributes have been matched with the same subject facts and these attributes are provided as an explanation as to why the expert-system chose the resource.

We implement an expert-system because it provides the efficiency and ease of developing a rule-based system with an already existing inference engine. We only need to write the rules using the expert-system language, which is simple to use and easy to read. There is only a need to write code to connect the expert-system with the Question Answering System. Furthermore, the expert-system is easy to maintain and can be reused when new rules from experts are available.

3.3.1 The fact base

The fact base is the expert-system’s temporary information database. We populate the fact base with facts from the subject as we collect them during a session with the user. These facts along with the rules from the knowledge base are used by the expert-system to fire rules and reach conclusions, which correspond to the resource database. Returning users need to start the chatbot session from the very beginning since subject-specific facts are deleted after the end of a session. We have not implemented a system where subject-specific facts can be stored for use in a future session because the functionality is beyond the scope of the project.

3.3.2 The knowledge base

We define rules in the knowledge base driven by the motivation of providing the user with recommendations as the user keeps answering questions with subject facts. Initially, the recommendations retrieved by the query may be insufficient for the subject's requirements since not all facts about the subject have been collected. With each new subject fact, the recommendations will be more refined and appropriate for the subject.

Rule Example

An example of a rule is as follows:

"If ADHD *IN* **condition** *AND*
  Physical Aggression *IN* **challenging behaviour** *AND*
  age *IS PRESENT AND*
  intervention *IS PRESENT AND*
  location *IS PRESENT*,
ATTRIBUTES FOR QUERY"

If a rule is satisfied, the system makes a query to the recommendations table of the resource database with subject facts. The conclusion fact is simply a query created from all the subject facts from the antecedent matched with their respective columns in the recommendation table. For example, an SQL query that is formulated from the conclusion of a rule, "ATTRIBUTES FOR QUERY" is :

SELECT website-url *FROM* RECOMMENDATIONS *WHERE*
  condition-column="ADHD" *AND*
  challenging-behaviour-column="Physical Aggression" *AND*
  age-column="USER-AGE" *AND*
  location-column="USER-LOCATION" *AND*
  intervention-column="USER-INTERVENTION"

Hence, we name the conclusion fact "ATTRIBUTES FOR QUERY".

The subject age, intervention, and location must be present for the rules to fire while condition and challenging behaviour determine rule classification. The query is executed to the database where resource attributes are matched with subject facts to retrieve a list of matching webpages. The links to these matched resources are recommended back to the subject via the chatbot messaging interface.

Rule Examples

Rules in the knowledge base are designed with the goal of query generation with each new subject fact collected by the Dialogue Manager. They are broken down into multiple categories, each with a different number of facts in the antecedent. The categories and their rules are described below.

Category 1 rule examples:

"If ADHD *IN* **condition**,
ATTRIBUTES FOR QUERY"

"If **age** *IS PRESENT*,
ATTRIBUTES FOR QUERY"

Category 2 rule examples:

"If ADHD *IN* **condition** *AND*
Physical Aggression *IN* **challenging behaviour**,
ATTRIBUTES FOR QUERY"

"If ADHD *IN* **condition** *AND*
age *IS PRESENT*,
ATTRIBUTES FOR QUERY"

Category 3 rule example:

"If ADHD *IN* **condition** *AND*
Physical Aggression *IN* **challenging behaviour** *AND*

age *IS PRESENT*,

ATTRIBUTES FOR QUERY"

Rule categories 4 and 5 consist of four different facts and five different facts joined by conjunction in the antecedent respectively. There are a total of 11 rules in category 1 with four different rules for four different challenging behaviours, four different rules for four different conditions and three rules for each of location, intervention type and age facts. For category 2, combinations of different facts are considered resulting in a much higher number of rules.

3.3.3 The Inference Engine

The inference engine is responsible for making decisions, in our case building the SQL query for the resource database, given the subject facts and the rules encoded by the knowledge base. We implement forward chaining as the method for the inference engine to fire rules from the knowledge base given subject facts from the fact base. Our expert-system is implemented using the python open source inference engine library called pyknow. When support for pyknow was dropped, we shifted our library to experta since it was well maintained [60]. CAMI expert-system consists of three parts: the knowledge base, the inference engine and the fact base.

3.3.4 Query Relaxation

If a user is interested in resources from Edmonton, a city in Alberta, and the system is not able to find any resources for Edmonton, the system can relax the **Location** attribute from the city to the province. With query relaxation [19, 20], the system allows resources from a wider area than specified by the user.

3.3.5 Query Tightening

If a query returns too many resources for a certain subject profile, we can apply query tightening techniques to filter the resources. By adding more constraints to the query like the quality of resources returned or the top-k rated resources from ratings collected from user feedback, we can implement query tightening.

3.3.6 Intermediary Rules

If the user wants to offer the numerical age of the subject, an intermediary rule that assigns the age of the subject from a number into a class (TEEN, ADULT, or CHILD) will benefit this use case. For example:

"If **age** *IS LESS THAN 12*,
  age=CHILD"

"If **age** *IS BETWEEN 12 AND 18*,
  age=TEEN"

"If **age** *IS GREATER THAN 18*,
  age=ADULT"

In order for these rules to be fired, we also need to allow users to submit written answers. We only allow multiple choice answers from the interface, but the ability to provide written answers is outside the scope of the current project and is a planned feature for a future implementation.

3.4 CAMI Question Answering System

The role of the Question Answering System is to ask questions, acquire facts from the interlocutor, and offer recommendations to the user. The questions are organised in templates where each question pertains to a fact. For example, the Dialogue Manager will ask "What is the child's age?" when it needs to determine the child's age. This information is collected by the Question Answering System independent of the expert-system and later transferred to the fact base of the expert-system.

3.4.1 Dialogue Manager

The Dialogue Manager is responsible for communicating appropriate questions to the user. The sequence of questions asked is decided by a rule-based algorithm. We introduced a graph to encode the ordering of questions and responses by the user. The graph is defined as questions representing the nodes, and the user responses as edges between questions. For example, if the user answered "No" to the question about whether they wished to be anonymous, the next node traversed would ask the user about the login email. If the user answered "Yes", the node traversed would be different.

When filling out a subject profile, the Dialogue Manager needs to ask appropriate questions. The response to a question adds a fact to the expert-system fact base. Each resource in the resource database is labelled with subject facts that we name "attributes" for clarity. During the resource annotation process, an expert labelled a certain website with appropriate attributes. For example, if a resource caters to children suffering from ADHD, that resource will be labelled for "Child" as age attribute and "ADHD" for condition attribute. Considering all the resources in the resource database where each resource is labelled with attributes, the frequencies of these attributes per resource are not the same due to each resource being different. The answer to the question of whether a subject is suffering from ADHD will reduce the number of possibilities of appropriate resources for that subject. By calculating the frequencies of the attributes distributed over resources in the resource database, the Dialogue Manager chooses the attribute with the highest frequency and asks the question pertaining to that attribute. The process continues in a loop until a rule in the expert-system is satisfied, resulting in a query that provides recommendations. The process described here is called highest information gain, where the question to ask pertains to the attribute with the highest frequency in a list of possible resources.

The Dialogue Manager implements the following algorithm in a loop in Algorithm 1 with $g(x)^D$ as the function for calculating the frequency of x, denoted by attributes over the resource dataset D (list of resources). Every resource is mapped with appropriate attributes by an expert. The attribute a with maximum frequency over the dataset D is then used to retrieve question q from a list of questions Q, which is mapped by attributes to questions. Currently, each resource in the resource database D is labelled with resource attribute a, age. Therefore, age occurs the most — the maximum frequency — over D compared to other attributes like location or challenging behaviour, etc. Question q pertaining to attribute age will be asked — 'What is the age of the subject". Once the question q is asked by the Dialogue Manager, the answer, f - a fact of the subject pertaining to question q is added to the fact base. The expert-system is then executed, rules are fired and queries are made to the resource database. A resource r is retrieved from the resource database with the execution of the query. Later, D is reduced to only include resources that match fact f to attribute a in the resource database. The program runs in a loop until D is empty, meaning there are no more resources applicable for the subject profile, or Q is empty, meaning all the available questions have been asked.

In any instance of the loop, the selected question q can only be asked once and any asked questions and its corresponding attributes are discarded from their respective lists. The question q is chosen randomly if multiple attributes have the same frequency. This approach may not be ideal in a real subject-doctor interaction, since the required question may be dictated by a medical reason rather than the frequency of facts in the remaining recommendations.

Algorithm 1 Program Logic

Input: X,Q,D,F ▷ // X is the list of applicable attributes, Q is the list of applicable questions mapped, D is the list of applicable resources and F is the Fact Base

$refine = False$ ▷ If $refine$ is True, resources are recommended with each question asked

$r = NULL$ ▷ Resource to recommend to the user

while D NOT EMPTY AND Q NOT EMPTY **do**
  $a \leftarrow \text{argmax}_x\, g(x)^D$ ▷ a is attribute with highest frequency over D
  $q \leftarrow Q(a)$
  DELETE q from Q
  DELETE a from X

  $f \leftarrow ask(q)$ ▷ Ask question q to user
  $F \leftarrow append(F, f)$ ▷ Add fact to fact base
  $r \leftarrow ES(F)$ ▷ Run Expert-System and get resource

  if $refine$ AND r EXISTS **then**
    print(r)
  $D \leftarrow D(a = f)$ ▷ Applicable resources reduced due to more facts

if $\not{r}efine$ AND r EXISTS **then**
  print(r)

In order to explain the algorithm better, let us define two resources $r1$ and $r2$ in dataset D. $r1$ has attributes "Age = child" and "ADHD = True" while $r2$ has only one attribute "Age = child". For the attribute age, the relevant question to ask is "What is the subject's Age", and for the attribute condition ADHD, the relevant question to ask is "Does your child suffer from ADHD". When the system starts, $g(Age) = 2$ since resource attribute age is present in both $r1$ and $r2$ — it occurs twice over all the resources in the database. $g(ADHD) = 1$ because only one of the resources is labelled with attribute ADHD. Since attribute age has the highest count, the question linked with age will be asked first. If the response to the question "What is the subject's Age" is "adult", the loop ends immediately because no resources has "adult" as a value for age — the list of applicable resources D becomes empty. If the answer is "child", the next question to ask is "Does your child suffer from ADHD". If the value is True, both resources $r1$ and $r2$ are returned. If the value is False, only resource $r2$ is returned. Once the resource link is shown in the interface, the loop ends since there are no more questions to be asked; Q is empty.

The idea behind selecting questions based on the highest frequency of attributes, $a \leftarrow \operatorname{argmax}_x g(x)^D$, in the resource dataset is to make sure that the chatbot do not ask irrelevant questions, and optimize the number of questions asked. Based on the example above, if the subject is an adult, none of the resources are applicable. The chatbot gets to that conclusion by asking one question about subject's age. However, if the chatbot ignores the formula for frequency of attributes and asks the question about the condition ADHD first, the chatbot still needs to ask the question about age before it can rule out that no resources are applicable for the subject. The chatbot needs to ask both questions which is not optimal. By applying the idea behind information gain, the chatbot only needs to ask the question about age to finish the conversation.

The Dialogue Manager uses a tree that generates a series of initial questions about consent in the order defined by the tree. The user needs to answer "Yes" to all the consent related questions; otherwise the chatbot will stop the conversation. After initial questions of consent are asked, the Dialogue Manager uses the Algorithm 1 to generate another tree of questions and answers, as shown in Figure 3.5. The structure of the generated tree changes if information about the resources D changes in the resource database. The Dialogue Manager fills out the subject profile and chooses the appropriate questions to ask as dictated by the generated tree. The chatbot is going to keep asking questions until there are no more resources left that match the criteria of the subject facts, or the list of questions to ask is exhausted.

In order to better understand how the system traverses the tree, let us consider a running example of a subject. The first relevant question asked is "Which city are you located in" to which the user answers "Edmonton" on behalf of the subject. From the tree, the system is going to traverse the

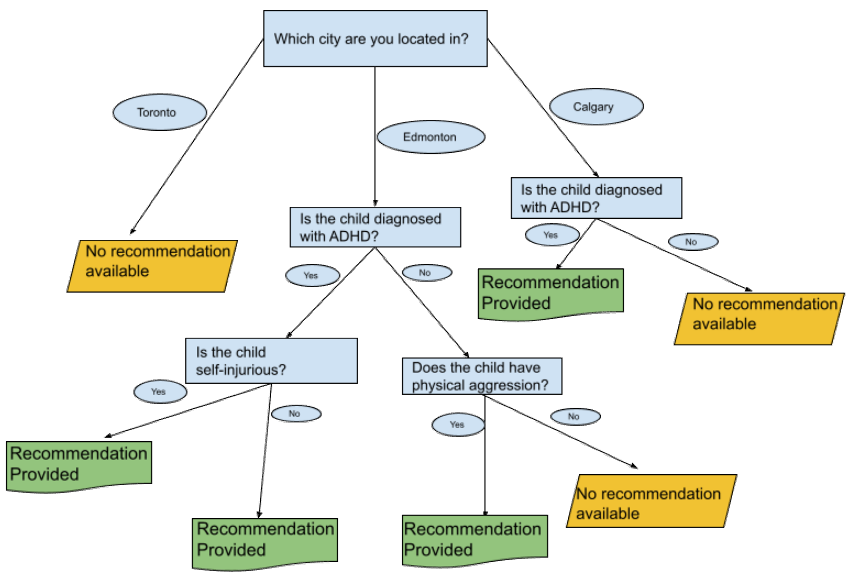


Figure 3.5: An example tree generated by the described Algorithm 1. The rectangular nodes of this tree are questions asked by the agent and the edges between the nodes are possible responses by the user drawn in circles. By using this algorithm, the chatbot is going to keep questions in a path where recommendations are available and until recommendations are found.

path connected to the answer “Edmonton” that leads to the question “Is the child diagnosed with ADHD”. The user answers “No” to this question. Then the system asks the user “Does the child have physical aggression?” The user answers “Yes” to the aforementioned question. The system responds with a relevant recommendation following the path in the tree.

In a practical situation, the order of questions generated by Algorithm 1 may not be accurate. A doctor may ask questions in an order that is clinically justified. However, our framework does not allow changing the ordering of questions once it is generated. We can improve the question order by including questions as rules in the expert-system.

An example of a simple rule maybe - “If **ADHD** is EMPTY, ASK question **ADHD**”. Once the user has answered the question about ADHD, other question rules that depend on the answer to ADHD may fire, e.g. “If **ADHD** NOT EMPTY and **LOCATION** is EMPTY, ASK question **LOCATION**”. Therefore, we can create a set of question rules that determine the order of asking questions vetted by an expert. In the examples above, the expert-system will ask the user whether the subject has ADHD first, and once the user has answered that question, it will ask where the subject is located. The order of asking questions is determined by the rules and not by the frequency of attributes in a set of given resources which may change from time to time. In this way, medical professionals can control the ordering of questions asked.

3.4.2 Website Framework

Our chatbot is based on a model-view-controller web application framework as depicted in Figure 3.6. The conversational agent CAMI is text-based. While speech-to-text and text-to-speech can be added, our project partners, practitioners in the NDD field, opted for a text-based application.

A web application framework needs to have a front-end where users can interact with the app, and a back-end server where the requests sent by the users are processed. Our conversational agent is built on a lightweight web framework named Flask [28], which uses Python. We use a model-view-controller framework, where the view is the front-end that users interact with; the con-

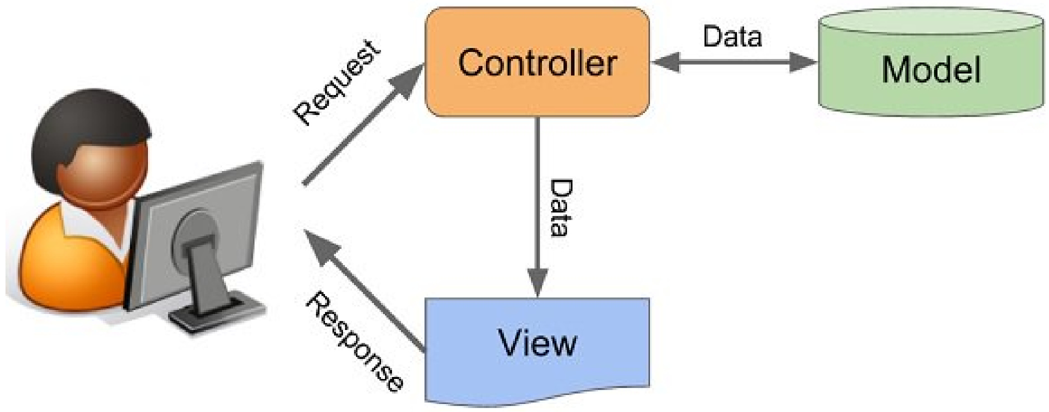


Figure 3.6: The structure of a web application made under a model-view-controller framework.

troller represents the back-end code that connects the front-end with the Question Answering System and the expert-system. The entire application is hosted on a uWSGI server that receives requests from users and handles traffic to multiple flask applications running concurrently on the back-end server. We use NGINX as the server protocol for our application.

Front-End

The front-end is written in dynamic HTML5 templates embedded with Python and Javascript code. This technique allowed variables from the back-end server to be used in the HTML templates. The questions and answers from the conversation are embedded into the HTML templates every time the chatbot asks a question or the user answers a question.

We use Javascript call back functions for “Yes”, “No”, and “Why” response buttons that can be clicked by the user on the interface. We implement response buttons to questions since it eliminates the complications introduced with processing text. With text-based answers, users may provide incorrect or vague responses that the Question Answering System may find difficult to process. By avoiding text-based responses, we make sure that the answers provided by users are exact and easy to handle. There are a few questions that are text-based such as the name and email address of the user, but the recommendations are not dependent on the answers to these questions.

When a link to a resource is provided by the chatbot, there is a 5 star rating system implemented in Javascript and HTML. The callback function for clicking on the 5 stars is written in AJAX. After the user clicks on any of the 5 stars, the rating information is received by server and saved to the resource database. The rating provides a quantitative assessment of the links provided by the application that can be analysed later to rank certain resources over others. AJAX [67] is an asynchronous process of sending data to the server via Javascript callback functions without having to load the entire page, only the portion the function is assigned to.

We use custom CSS designs implemented from an open source CSS framework called Twitter Bootstrap [15]. The framework provides built-in Javascript functions for using simple buttons, colours, and layouts that are tested as user-friendly. We implement a design that is easy to follow, user-friendly, and aesthetically pleasing.

Back-End

When requests are sent from the front end by the user, they are received by the server and sent to the flask application controller that handles the request. The back-end has functions that handle AJAX requests sent by the front-end. It has a function to handle POST requests sent by the user in the form of messages sent to the server.

The most important variable in the back-end is the Session variable, which is transferred from the front-end to the back-end. The Session variable is a dictionary object containing key-value pairs of variables. It is possible to add custom variables and their values to the dictionary object. The Session variable temporarily saves information from the conversations, such as the questions asked by the agent and the answers provided by the user, and the state of the agent. Without the session variable, it is impossible to keep track of the state of the agent. Once the session between a user and chatbot ends, the Session variable is destroyed.

The back-end stores the code behind the expert-system and the Question

Answering System. Once the back-end receives a response from the user, the response is passed to the Question Answering System for analysis and saved to the database.

3.4.3 Facts and Variables

The important variables are defined beforehand by the experts as the information required by the chatbot system to recommend a resource. Facts for these variables are gradually filled with each interaction between the user and a chatbot. By asking the user questions, the chatbot fills up a subject profile that we refer to as subject facts later required by the inference engine to recommend a resource to the user. However, there are other facts that the chatbot collects from the user such as their name and email, which are not needed by the chatbot for recommendations. The user may choose to not provide the name and the email, and the conversation will still continue with an anonymous user. The variables that these important facts belong to are :

Condition

It is defined as the NDD from which the subject suffers. The condition is one of ADHD, Autism, Intellectual Disability, and Neurodevelopmental Delay. The subject can potentially suffer from one or more of these conditions.

Challenging Behaviour

It is defined by an action that the subject is not in control of, and is triggered due to the condition they are suffering from. The subject can exhibit one or more challenging behaviours such as physical aggression, self-harm, temper tantrums, and non-compliance.

Age

It is one of child, teen, or adult.

Type of Intervention

It can be one of Parental, Professional or General. It categorizes the resources and webpages that the experts have gathered. For example, if a webpage describes an intervention by a professional such as a caregiver or a health practitioner, it belongs under the category of Professional interventions.

Location

It is the place where the user and the subject lives, where they will be able to access resources. However, the user can opt to receive recommendations of resources from places other than the local residence area. There are multiple values of locations and they are categorized into cities and provinces of Canada.

3.4.4 Database Controller

The database controller connects the back-end code with the resource database. It is defined by functions that can add, modify, and delete items in certain tables in the resource database. It is also responsible for adding, editing, and deleting information about resources from the resources table (known as interventions). The resource database is separated from the entire system, accessible by the database controller.

3.4.5 Administration Console

The administration console is an interface connecting an administrator to the resource database via the database controller. An admin, an expert or an employee from the research team is authorized to have access to the database and its contents. The console is designed to make it easy to change information in the database without any programmer expertise. The console is also used to change attributes of websites and recommendations, via the interventions table, to represent the decoupling of the rules of the expert-system from the resource database. Figure 3.7 shows the login webpage to the administration console. After logging in, the webpage shows a list of tables available in the database that is accessible by the administrator. These tables that are

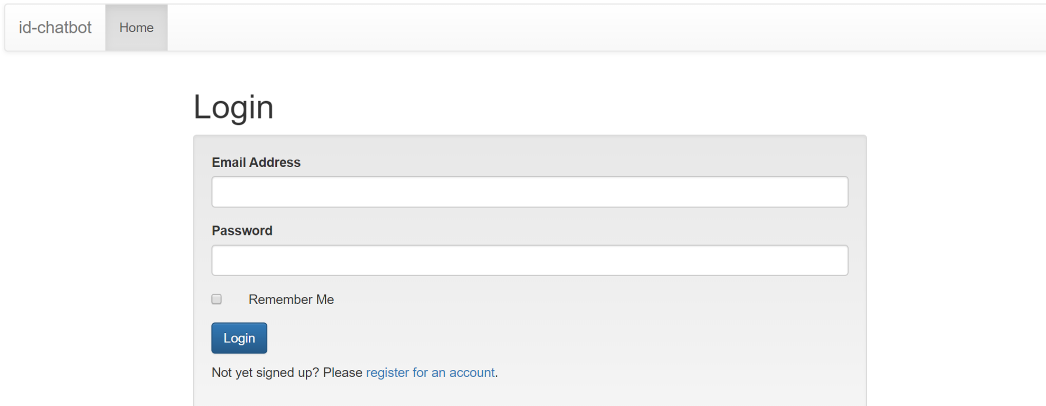


Figure 3.7: The website for logging into the administration console.

described in the next section.

3.5 The CAMI Resource Database

The CAMI MySQL resource database is an important aspect of the conversational agent since it has tables with information about resources, and information about conversations that has been saved during sessions. Each table in the database has a mandatory column called the identifier that gives a unique integer to any item in the table.

The challenging behaviour table stores information about all the challenging behaviours that the application handles. Figure 3.8 shows all the examples of challenging behaviour and how the table is displayed in the administration console.

The condition table stores information about four different NDD conditions that the application handles. They are Autism, ADHD, Neurodevelopmental Delay, and Intellectual Disability. Each subject can be linked with one or more of these conditions. Figure 3.9 shows all the examples of conditions and how the table is displayed in the administration console interface.

The intervention table stores information about useful websites and links that can help the caregiver of a subject understand what they are going through and what steps to take to alleviate the situation. It stores whether

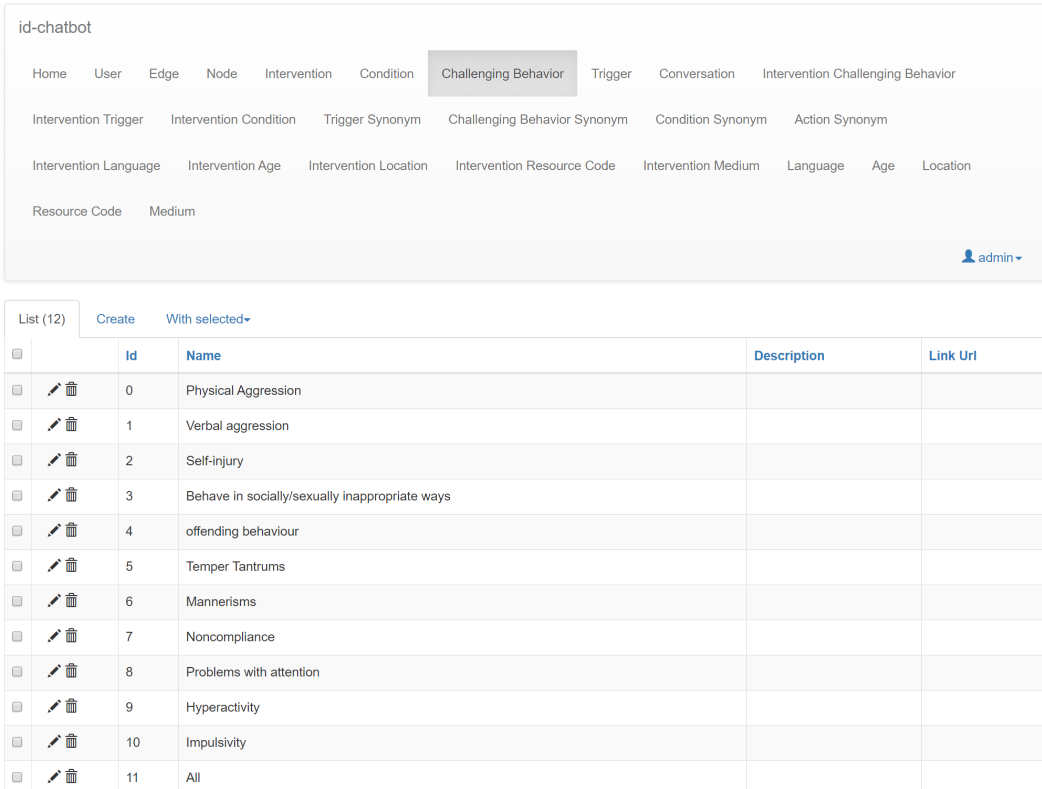


Figure 3.8: The website that lists all challenging behaviours in the administration console.

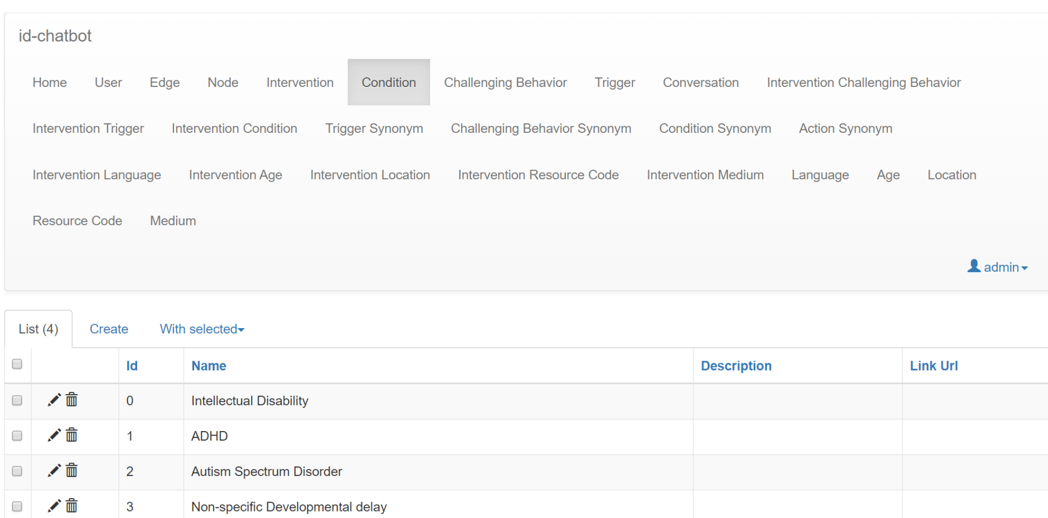


Figure 3.9: The website that lists all the conditions in the administration console.

an intervention is Parental, Professional, or General; the suggested age of the subject for which the resource applies; the topic of the resource; some tag words; and the website link.

The intervention table is also linked with the intervention challenging behaviour and intervention condition tables. The database controller has functions that link the interventions with challenging behaviours, conditions, age, location, and other subject facts. Figure 3.10 shows an example of an intervention linked with a challenging behaviour in its own intervention challenging behaviour table. The identifier for the intervention is linked with the identifier of the corresponding challenging behaviour in this table. All linked tables are designed simply by having the identifiers of the linked entities as columns in the tables.

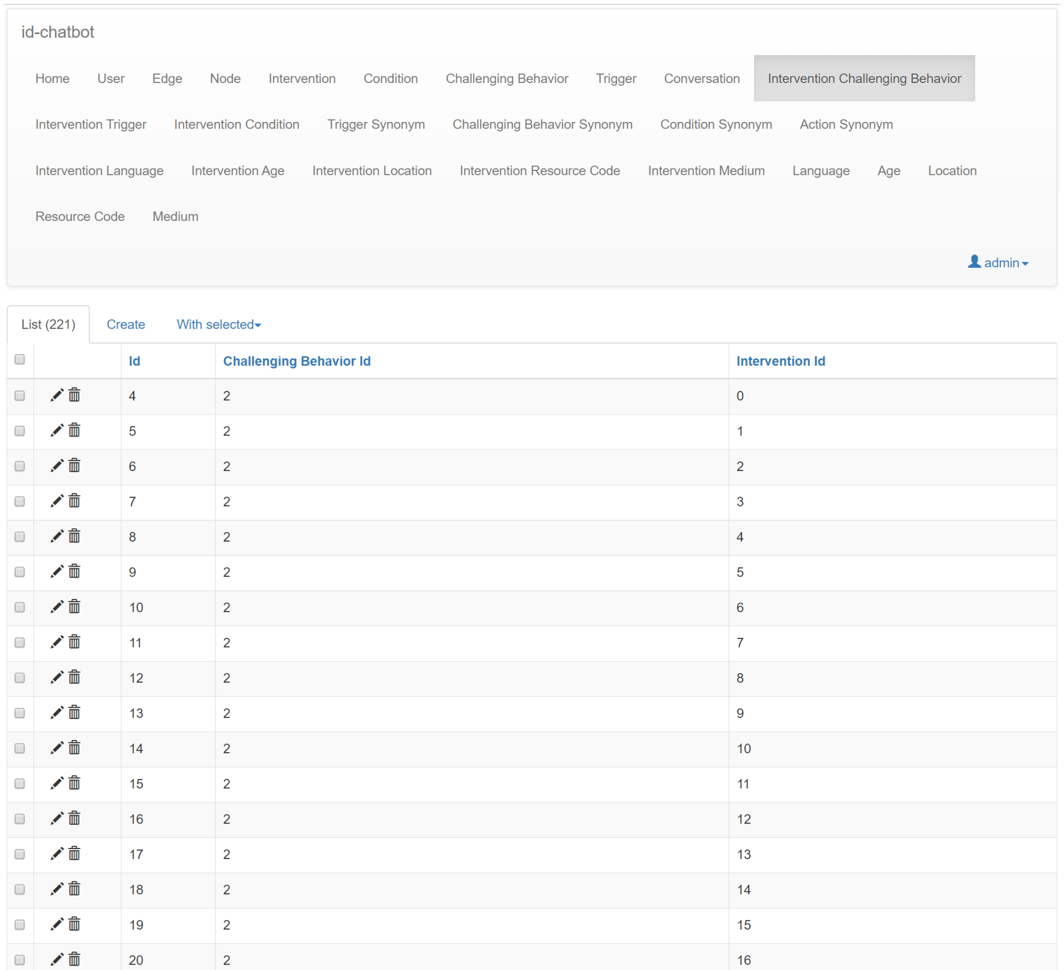
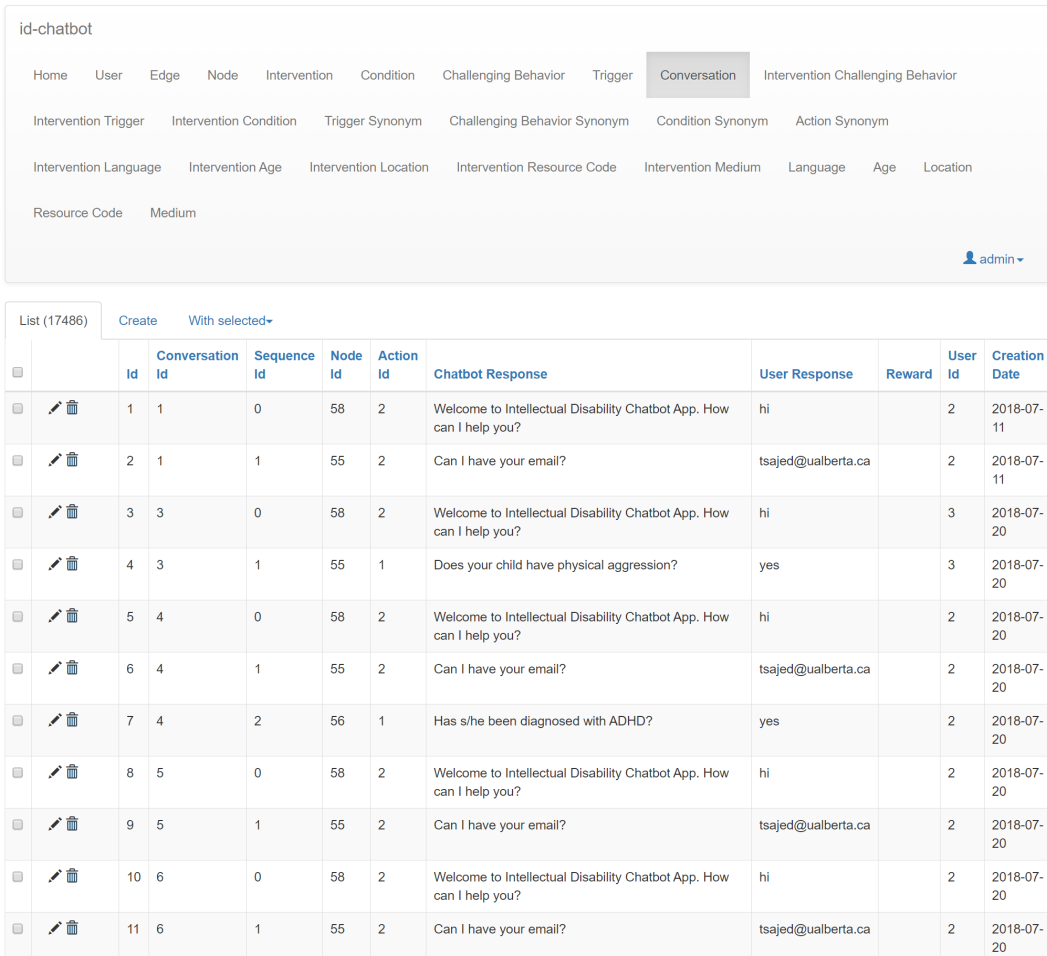


Figure 3.10: The website for intervention and challenging behaviour links.

The User table stores information about users that have consented to having the information of the subject saved during a session. It saves the email and password for the user and any other relevant information needed during the chat session. The User challenging behaviour and User condition tables save information about challenging behaviours, and conditions linked with the user via the user identifier.

The conversation table records all the conversations between the user and the chatbot. For every conversation, there is a user identifier assigned to it. However, if the user decides to not have their information recorded, they are anonymized and the user identifier is left blank. Every conversation also includes at least one interaction. It includes the chatbot question/answer and the user response to the chatbot. The conversation has a column called reward, which is the rating of the user on a website if the chatbot has returned a website. When a user rates a website, it is saved in the conversation table, with the reward being the number of stars the user has assigned to a particular website. Figure 3.11 shows a representation of the conversation table in the administration console and a list of conversations that are stored in the table.



Id	Conversation Id	Sequence Id	Node Id	Action Id	Chatbot Response	User Response	Reward	User Id	Creation Date
1	1	0	58	2	Welcome to Intellectual Disability Chatbot App. How can I help you?	hi		2	2018-07-11
2	1	1	55	2	Can I have your email?	tsajed@ualberta.ca		2	2018-07-11
3	3	0	58	2	Welcome to Intellectual Disability Chatbot App. How can I help you?	hi		3	2018-07-20
4	3	1	55	1	Does your child have physical aggression?	yes		3	2018-07-20
5	4	0	58	2	Welcome to Intellectual Disability Chatbot App. How can I help you?	hi		2	2018-07-20
6	4	1	55	2	Can I have your email?	tsajed@ualberta.ca		2	2018-07-20
7	4	2	56	1	Has s/he been diagnosed with ADHD?	yes		2	2018-07-20
8	5	0	58	2	Welcome to Intellectual Disability Chatbot App. How can I help you?	hi		2	2018-07-20
9	5	1	55	2	Can I have your email?	tsajed@ualberta.ca		2	2018-07-20
10	6	0	58	2	Welcome to Intellectual Disability Chatbot App. How can I help you?	hi		2	2018-07-20
11	6	1	55	2	Can I have your email?	tsajed@ualberta.ca		2	2018-07-20

Figure 3.11: The website that lists all the conversations in the administration console.

3.6 CAMI Interface

Once the user connects with the chatbot interface, they are introduced to the ethics-approved online consent and information letter detailing the purpose

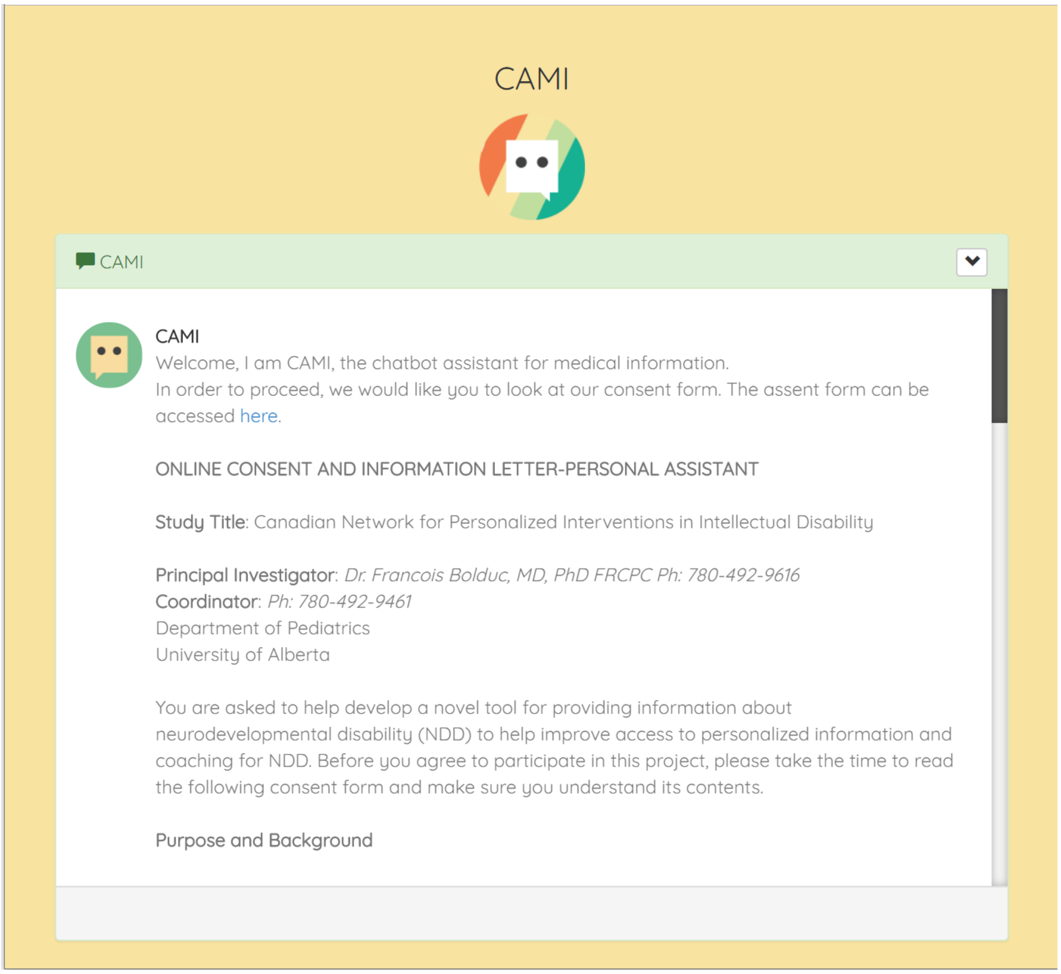


Figure 3.12: CAMI showing the initial consent page.

and background of the CAMI chatbot. There is a corresponding assent form on the website. If the user consents to using the chatbot and participating in the conversation, they are asked some additional questions about the chatbot. These are "Yes" or "No" questions, to which the user must respond in the affirmative; otherwise the chatbot disconnects and ends the conversation. The Dialogue Manager, shown in Figure 3.3, is responsible for handling the path of the conversation with each user response and question template.

Figures 3.12, 3.13, and 3.14 show snapshots of the consent section of the chatbot. The consent, however, is not part of the expert-system. A user is only allowed to progress if they agree to all the conditions asked in all these questions. Figure 3.15 shows a user agreeing to the consent of the conversation.

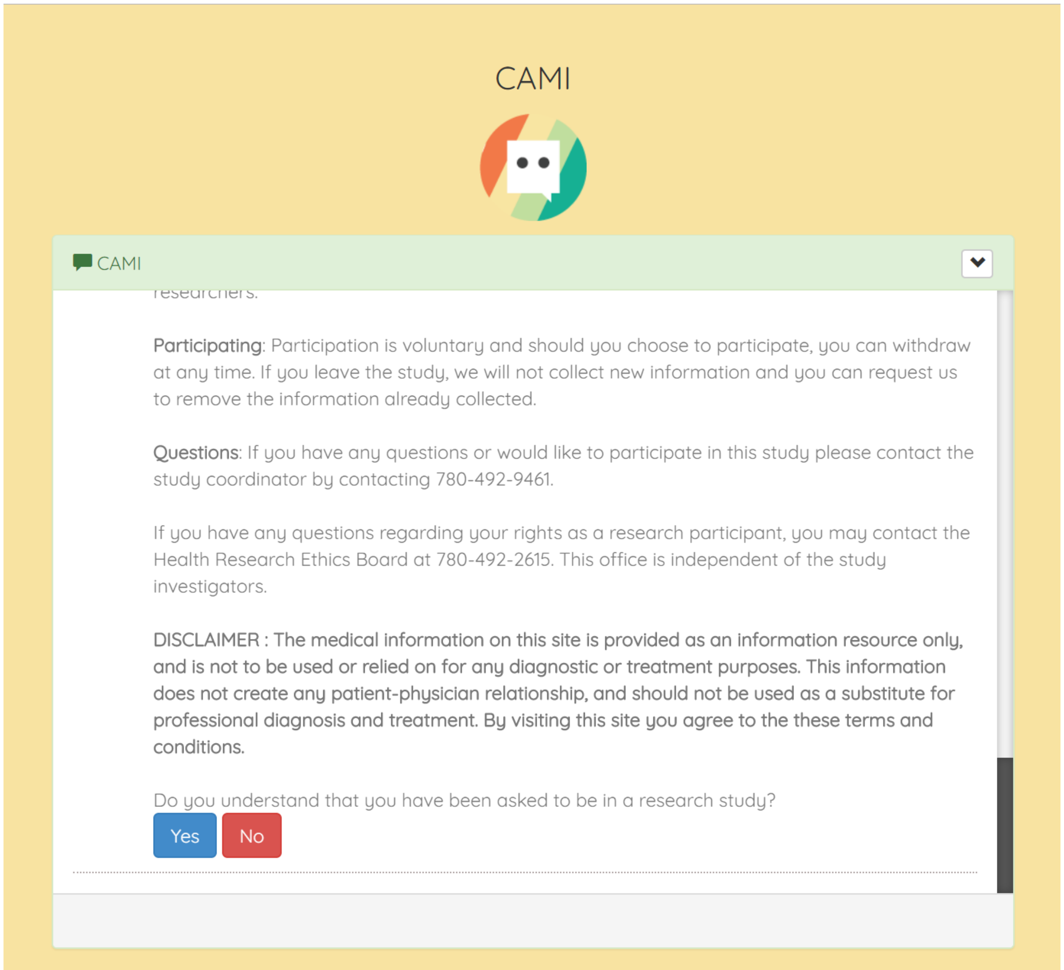


Figure 3.13: CAMI showing an intermediate consent page.

.

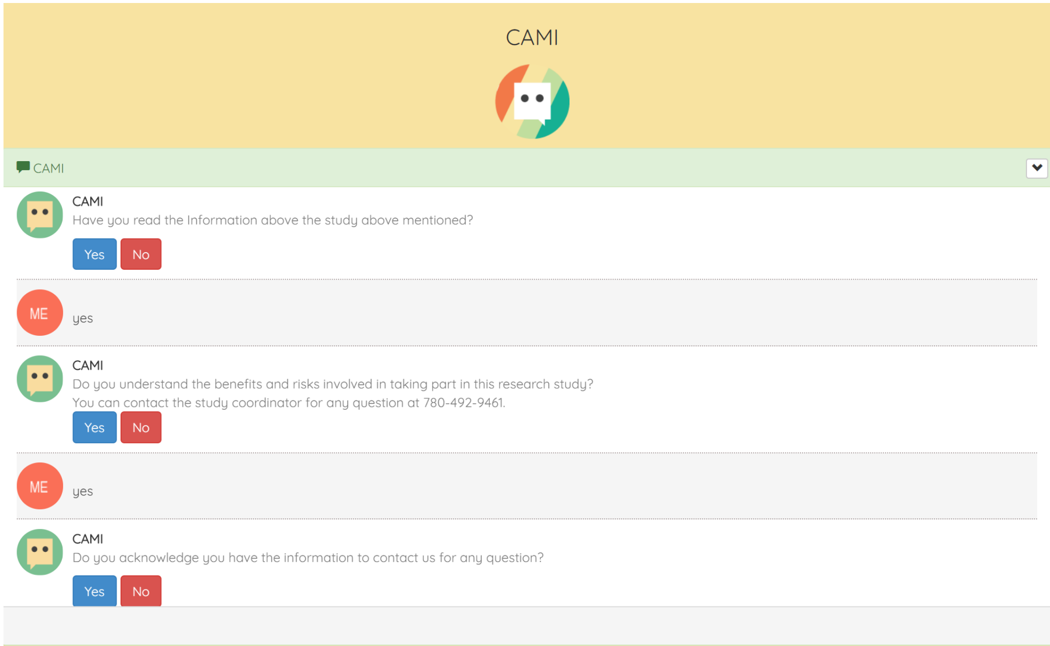


Figure 3.14: CAMI showing some consent-related questions and answers between the user and the chatbot.

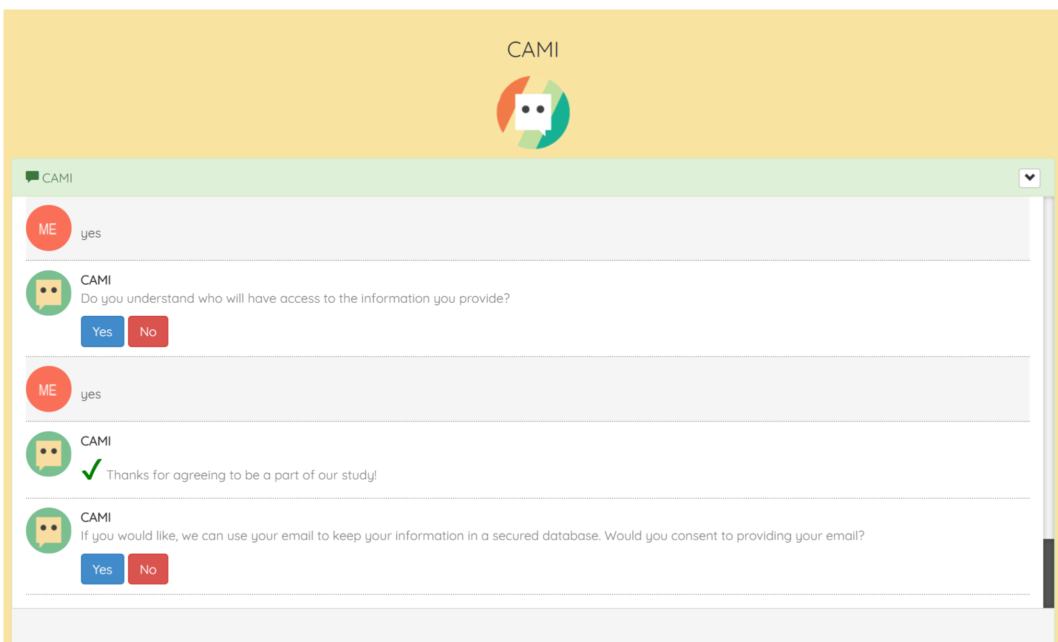


Figure 3.15: CAMI showing that the user agrees to the consent.

The chatbot continues the conversation by asking some personal questions, such as whether the user is a parent or health care professional. CAMI also asks for the user's email and whether they want their conversation recorded for the future. Figures 3.16 and 3.17 show some personal questions asked about the user. The questions asked until this point follow a static graph with a pre-defined order of questions, given the answers to the previous questions. Any questions asked after this point are determined by a dynamic graph, depending on the state of annotated resources in the resource database, generated by the Dialogue Manager in the Question Answering System.

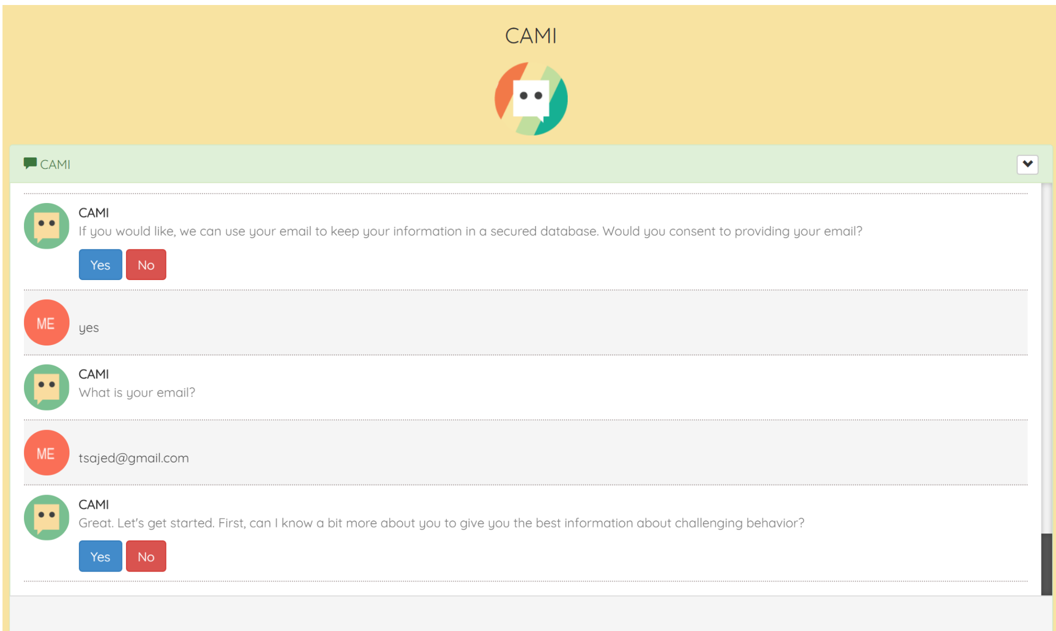


Figure 3.16: CAMI asking for email after user has consented to the study.

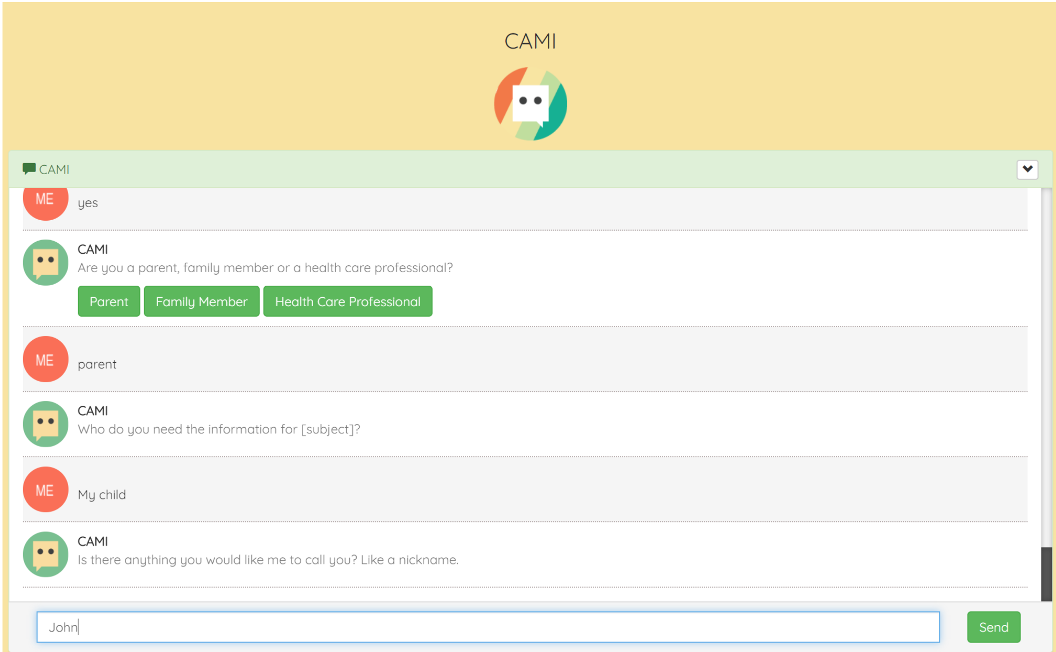


Figure 3.17: User responding with email after consenting to providing the email.

Chapter 4

Implementation of a prototype and appraisal of its robustness to dynamic resources

The biggest advantage of our expert-system-based chatbot for recommendation is that we separated the expertise about medical features from the information about the resources (i.e. webpages). The expertise about the medical features, while dependent on the application domain, here NDD, is fixed. We converted this acquired *savoir faire* into rules in the knowledge base. The information about the webpages is gathered separately and saved in the resource database. This separation, depicted in 3.4, allows updating the resource database by non-technical staff without jeopardizing the proficiency encoded in the knowledge base. Indeed, many new webpages related to NDD continue to be published and existing resources also frequently change. The resource database simply stores the url and the characteristics of the webpages, in terms of the subject facts used by the inference engine. Updating this database allows the chatbot to improve and adapt to the new reality online as new resources are made available and old resources get deprecated.

4.1 Adding A Website

We decided to test whether a new website added to the chatbot framework is offered to the user under the right subject profile. This addition is done via the administration console of the application, without needing any change in the code or in the expert-system. Information about the website is simply added to the resource database and instantly becomes a candidate for recommendation if the facts of subject satisfy the appropriate attributes for the website, for example, a condition such as ADHD, a challenging behaviour such as physical aggression, etc.

We decided to add a website, the 211 community support group for mental health issues in Edmonton and surrounding area [1] to the resource database. Annotated by an expert, the website applies to any child, teen, or adult suffering from any condition ranging from ADHD, Autism, or Intellectual Disability, exhibiting any challenging behaviour from physical aggression, self-injury, and living in Edmonton. Figure 4.1 and Figure 4.2 show a progression of images about the flow of questions and answers in the conversation between the user and the chatbot. We chose an imaginary subject to represent the path followed by the chatbot. The subject is a child who lives in Edmonton and suffers from Intellectual Disability, exhibits physical aggression and self-injury. The user representing the subject is looking for Parental intervention. By answering these questions, the chatbot recommended the right website — the one selected for this experiment. This demonstrates the fact that it is very easy to add additional websites to the chatbot framework through its administration console without having to change the inference engine or rules. The images used here are a mock-up and do not represent the currently implemented interface. However, the current images from the interface are used and edited to create the mock-ups. The original images are shown in the Appendix figures A.1 and A.2.

4.2 Editing A Website

In order to show that the chatbot adapts after changing information about a resource from the administration console, we decided to change the attributes of the 211 support website. We only changed the intervention type of the website from Parental to General intervention. Under this adjustment, the

[1] https://edmonton.cmha.ca/programs-services/211-alberta-edmonton-and-area/

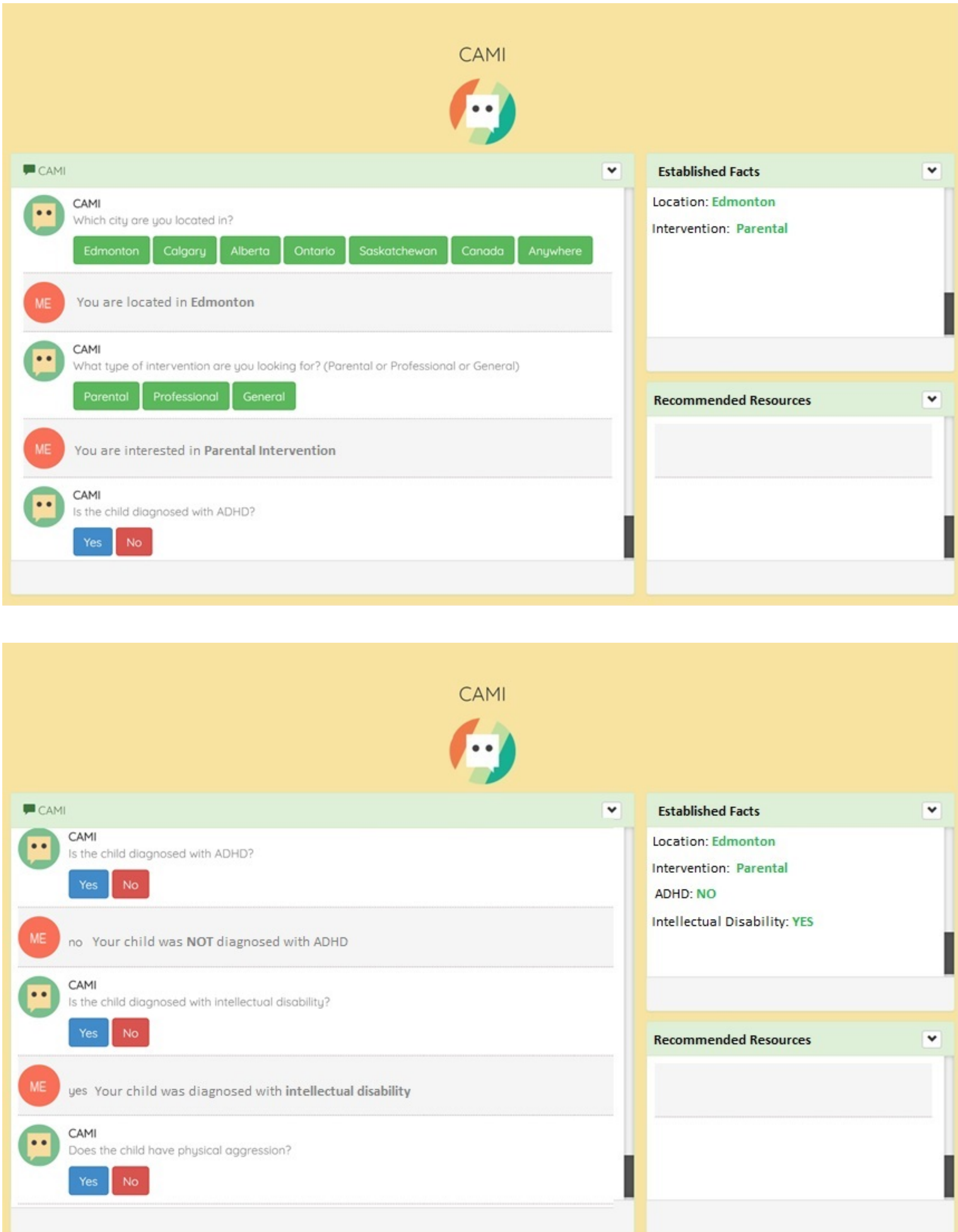


Figure 4.1: The progression of a conversation between the chatbot and a user that ends in the recommendation of the 211 support website (Mockup).

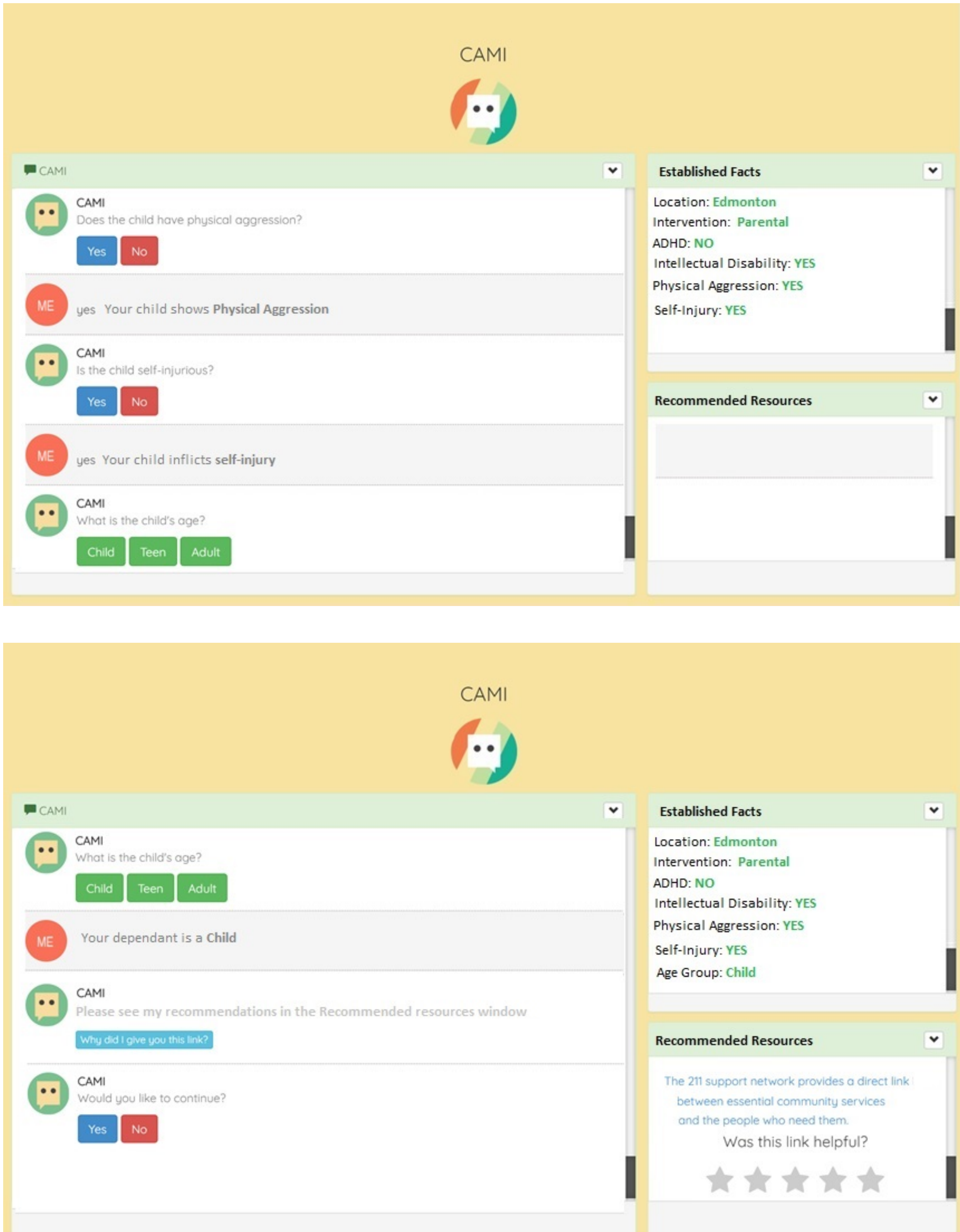


Figure 4.2: The progression of a conversation between the chatbot and a user that ends in the recommendation of the 211 support website (Mockup).

chatbot should offer the recommendation of 211 support website under General intervention, not Parental intervention.

We logged into the administration console with admin privileges in order to change the features of the 211 support website. The intervention type, e.g. Parental, is saved under the table Resource Code. The table Intervention Resource Code links the intervention and the intervention type together. In order to make a change to the intervention type of a website, we changed the resource code identifier of the link between an intervention and resource code from 2 to 3, 2 being Parental and 3 being General. It is important to be able to change the attributes of websites since they maybe updated with new information that need to be reflected on the recommendations.

Once we made the change, we tested a conversation between the chatbot and an imaginary user in order to show that the change has indeed taken place. Before the change, we knew that there were no websites recommended under the city of Edmonton and General intervention. If the 211 support website is indeed recommended via the right features, our framework can adapt to changing website attributes. To make sure the right set of facts are selected, we chose an imaginary subject who lives in Edmonton, suffers from Intellectual Disability and shows physical aggression. However, we made the user representing the subject choose General intervention instead of Parental intervention — the only change we made from the experiment of adding a website under Section 4.1. Figures 4.3 and 4.4 show the progression of a conversation between the imaginary user and the chatbot that resulted in the correct recommendation of the 211 support group website.

Since the recommendation applies for either physical aggression or self-injury as facts of challenging behaviour, selecting one or the other or both is going to result in the recommendation of the website. However, if we choose "No" for both challenging behaviours, the chatbot is going to decide against offering the recommendation since the user has to choose at least one of the annotated challenging behaviours for the recommendation of the 211 support website. Not choosing any challenging behaviour suggests that the subject is looking for something else that is not covered by the 211 support website since challenging behaviour is an integral feature to the recommendation of that website.

4.3 Deleting A Website

Sometimes, websites lose domain name and hosting services resulting in their cancellation. Sometimes, recommendations are updated and they loose a certain level of quality of information acceptable by an expert. Therefore, it is important that information about these websites can be deleted from the administration console easily without having to change the expert-system.

In order to show whether the chatbot adapts to an expert deleting a website, we decided to conduct an experiment where we deleted the 211 support website from the administration console. There are two ways to perform that action. An expert can delete the website by clicking on the delete button in the intervention table view of the admin console. They can also decide to edit the intervention and change the value of Export from 1 to 0, meaning the website is going to be kept in the resource database, but will no longer be recommended by the chatbot.

Once we deleted the information about the website by changing the value of the Export from 1 to 0, we decided to test the chatbot. We chose the city of Edmonton, and once we clicked on General intervention, the chatbot showed that there were no recommendations available (as in Figure 4.5). We also tested the route of clicking on the Parental intervention which provided no recommendations either. Therefore, the experiment showed that deleting the information about the website from the resource database succeeds, resulting in the appropriate adaptation of the chatbot recommendation system.

4.4 Heuristic Evaluation

Our team performed a preliminary heuristic evaluation of the chatbot to ensure that it met minimum usability standards. Evaluating usability can catch up to 80 per cent of usability problems before ever exposing a real user to the system [59]. Our evaluators, trained in the human-computer interaction method of

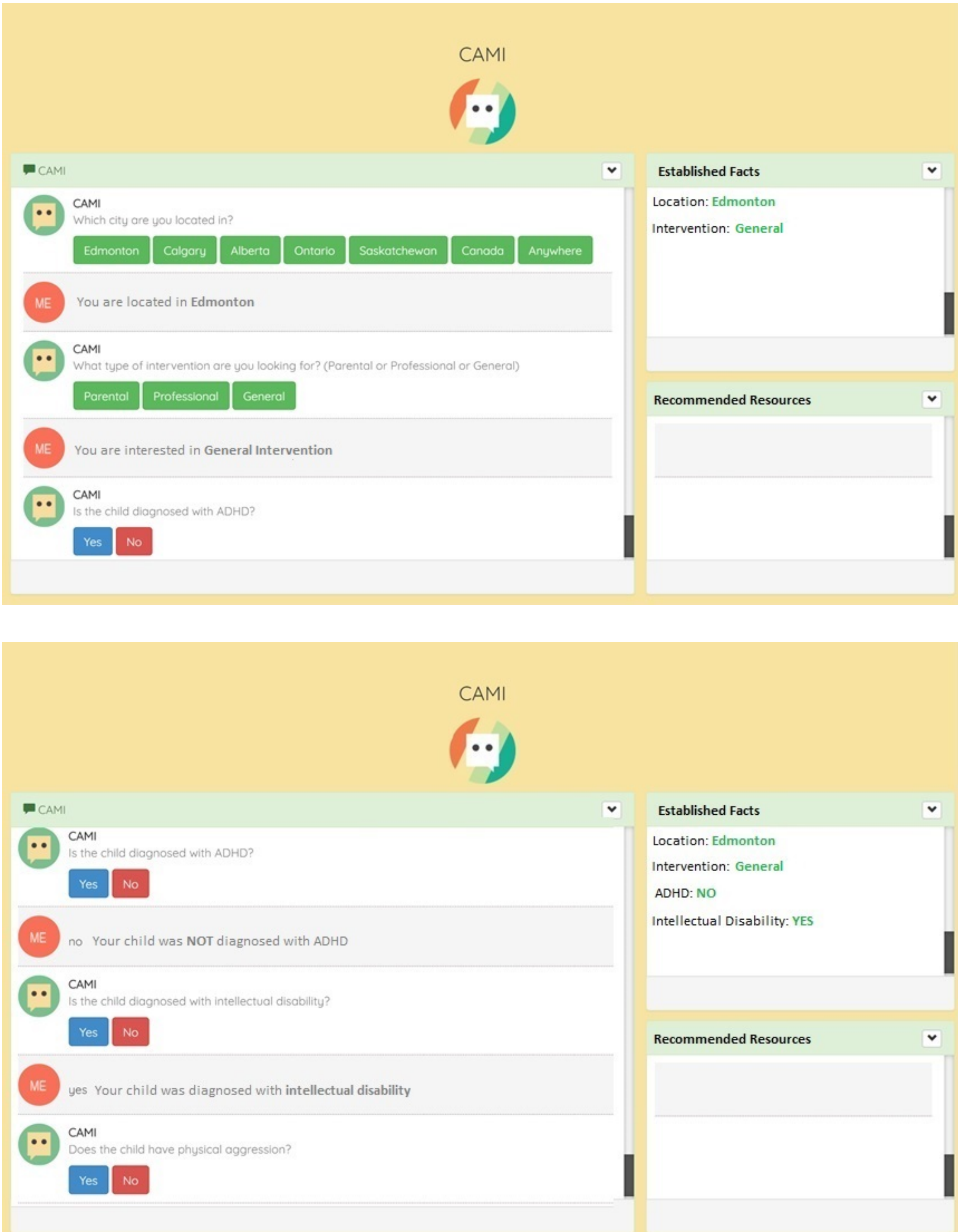


Figure 4.3: The progression of a conversation between the chatbot and a user after editing information about the 211 support website (Mockup)

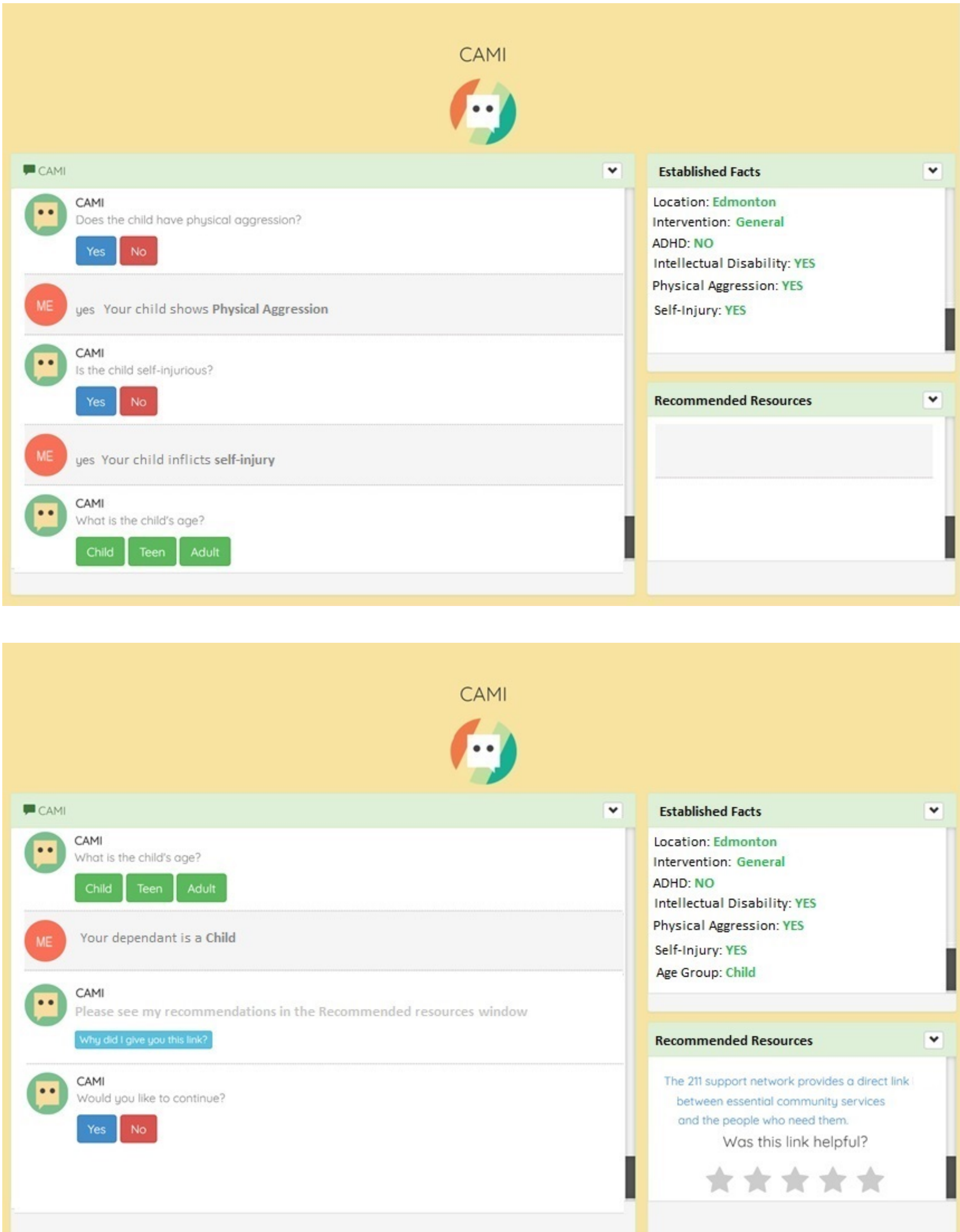


Figure 4.4: The progression of a conversation between the chatbot and a user after editing information about the 211 support website (Mockup).

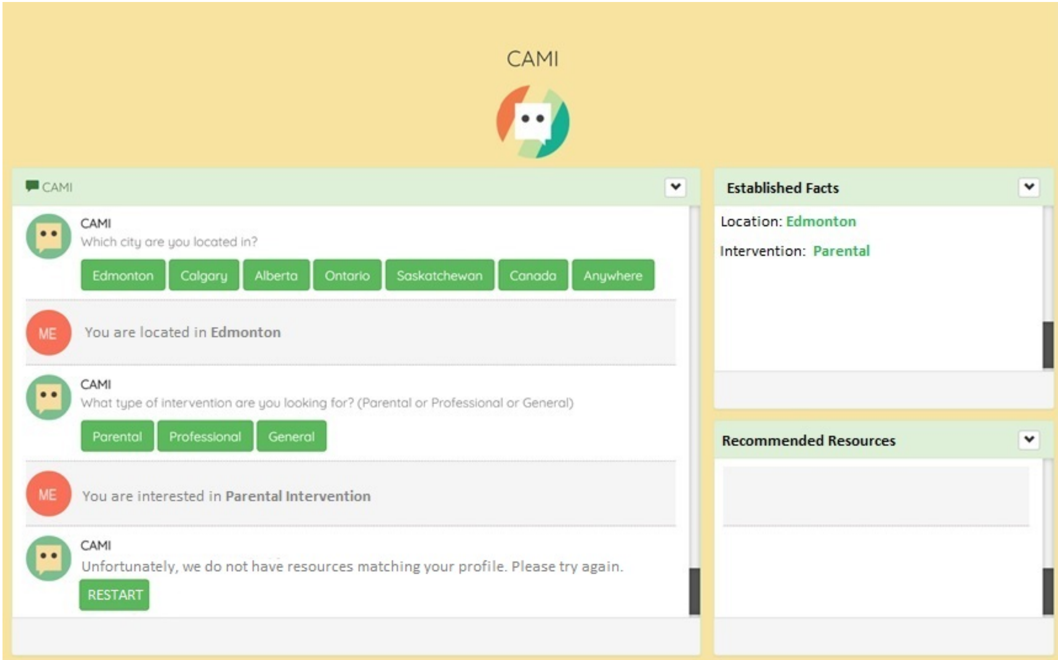


Figure 4.5: The progression of a conversation between the chatbot and a user under after the deletion of the 211 support website.

heuristic evaluation [58], tested the chatbot against qualitative usability standards. Basic design principles [102] and Nielsen's usability heuristics, which were created from a corpus of over 290 usability problems, were applied during the evaluation of the existing chatbot. The experiment revealed several usability design problems within the chatbot and several bugs that affected the functionality of the system. For each of the Nielsen's heuristics that was violated, the evaluators provided a severity score that determined how severely the violation impacted the functionality of the chatbot, a description of the violation, a potential fix for the violation, and a location where the heuristic was violated. Table 4.2 provides a frequency of all the violations starting from the most frequent to the least frequent. Table 4.1 provides a description of each of the 33 violations, including the nature of the heuristic violation, the location on the interface where it occurred, and the severity of the violation.

The first error in the Table 4.1 was given the severity of 4, (very severe), and it broke the chatbot functionality. It happened when we allowed the user to type answers in written format to "yes"/"no" questions. Typing anything other than “yes”/“no” to those questions broke the functionality of the chatbot and it stopped responding. The user had to start again from the very beginning in order to continue the conversation, and lost all the information provided to the chatbot in the current conversation. It disrupted the user experience and rendered the chatbot unusable. In order to fix the issue, we prevented the ability of the user to answer in written format to “yes”/”no” questions. Instead, we allowed the user to click on “yes”/“no” buttons so that the answer provided by users is always absolute and any ambiguous or erroneous written answer can be avoided. After the change was made, system failures resulting from ambiguous answers were avoided.

Visibility and system status was the most frequent heuristic violation reported from the evaluation. There were multiple instances during the conversation flow when the chatbot stopped responding to the user without providing any explanation as to why the chatbot did so. The system status of the chatbot was not understood as a result, hence the violation was in the category of system status. This resulted when the chatbot had no recommendations to make. To fix the bug, we added a message from the chatbot that the system had no recommendations to make given the subject facts and urged the user to start from the beginning by clicking on a button that started the conversation from the beginning. The change allowed the user to understand that no recommendations were available for the subject, and to start the conversation again if necessary. The change fixed the issue and the violation was avoided in a later build of the chatbot.

There were other issues that came up during the heuristic evaluation such as the Application Name not being consistent. This fell under the category of Consistency and Standards. It was an issue of very low severity, hence the value of 0 assigned to the row number 33 of Table 4.1. In order to resolve the issue, we fixed all the instances where “Chatbot” was misspelled as “ChatBot”. We changed all the instances to “Chatbot” from “ChatBot”, thereby fixing the issue.

Out of 33 issues reported during the heuristic evaluation, all but 6 issues were resolved. Issue number six was reported as a result of a lack of documentation and help for the users to navigate through the chatbot. Help messages and documentation generally improve the user experience further, but we decided against it since a lot of work was needed in order to provide constructive help messages. An expert needs to create the documentation and help messages detailing the meaning of certain medical terms and their importance in the chatbot application. As a result, we left the modification of the chatbot interface for the future and we expect that the help messages and documentation can be implemented after consultation with an expert.

Tables 4.3 and A.1 describe the heuristic item and the description of the nature of violations that we were unable to fix. Future improvements should focus on improving the user interface further by fixing the six violations described in the table and then undergoing another heuristic evaluation to assess whether the changes made improved the usability standards of the interface. With a lot of the issues already fixed, the state of the user interface is better than before, and using the chatbot should be easy and simple. However, any new violations can only be found by another heuristic team evaluation on the interface, and although it will need additional funding, it is highly recommended and essential before the chatbot can be published for use with real users.

A heuristic evaluation of a product is essential since it exposes many usability issues not identified by the developers. As a developer, it is difficult to map every usability bugs that can arise when using a product, in our case a chatbot. The perspective of usability evaluators maybe different from the perspective of a developer. The process of analysing, understanding the feedback and fixing some of the issues raised by the evaluators helped me become more aware of the usability issues discussed in this section. Therefore, we feel it is important to include the heuristic evaluation anslysis in this thesis so that other developers can learn from the mistakes I made during the development of the chatbot, improve awareness of the usability of a product and potentially avoid usability bugs resulting from the development of a product.

	Location	Heuristic Violated	Severity
1	General	Error Prevention/Recovery from Errors	4
2	System	Visibility of System Status	4
3	Initial Chat	Flexibility & efficiency of use/ Visibility of System Status	3
4	General	User Control & Freedom	3
5	Chatting	Flexibility & efficiency of use	3
6	System	Help & documentation	3
7	link recommandation	Help users recognize, diagnose, recover from errors/Visibility of System Status	3
8	Chatbot Message	Visibility of System Status/Match Between System and Real World	3
9	Chatbot Text Input	Consistency & Standards	2
10	Chatbot Question	Match Between System & the Real World	2
11	My Message	Proximity	2
12	link recommendation	Contrast	2
13	link recommendation	Consistency & Standards	2
14	Suggestion Chatbot Message	Consistency & Standards	2
15	Initial Chat Message	Visibility of System Status	2
16	Initial Chat Message	Contrast	2
17	General	Aesthetic & Minimalist Design	2
18	Chatbot Text Input	Visibility of System Status	2
33	Chatbot Message	Consistency & Standards	2
19	Initial Chat	Visibility of System Status	1
20	Initial Chat	Aesthetic & minimalist design	1
21	initial Chat	Help users recognise, diagnose, and recover from errors	1
22	Chatting	Similarity	1
23	Chatting	Contrast	1
24	Chatting	Match between System & Real World	1
25	Message textbox	Match between System & Real world	1
26	Initial Chat Message	Contrast	1
27	Suggestion Chatbot Message	Visibility of System Status	1
28	Suggestion Chatbot Message	Contrast/Proximity	1
29	Initial Chat Message	Consistency & Standards	1
30	General chatbot question messages	Consistency & Standards	1
31	Chatbot Text Input	Flexibility & Ease of Use	1
32	App Name	Consistency & Standards	0

Table 4.1: This table shows all the heuristic violations, their violation locations in the chatbot interface, and the severity of those violations.

Heuristic Violated	Frequency
Visibility of System Status	8
Consistency & Standards	7
Contrast	5
Match between System & Real World	4
Flexibility & Ease of use	3
Help users recognise, diagnose, and recover from errors	3
Proximity	2
Aesthetic & minimalist design	2
Similarity	1
User Control & Freedom	1
Help & Documentation	1

Table 4.2: Descriptive Statistics of the Heuristic Evaluation.

	Location	Heuristic Violated	Severity
4	General	User Control & Freedom	3
6	System	Help & documentation	3
10	Chatbot Question	Match Between System & the Real World	2
33	Chatbot Message	Consistency & Standards	2
20	Initial Chat	Aesthetic & minimalist design	1
29	Initial Chat Message	Consistency & Standards	1

Table 4.3: This table shows the location, heuristic violation and the severity of all the heuristics that we were unable to fix and left for future improvements.

Chapter 5

Conclusion and Future Work

We have developed an expert-system-based chatbot framework to provide recommendations for subjects suffering from certain medical conditions pertaining to NDD. The application is meant as a resource of information and not as a substitute for doctor's advice. We focused on decoupling the rules made by the expert from the information of the resources so that anyone can update the information of resources without changing the rules of the expert-system. The idea of the framework may be used by chatbots for different applications that provide recommendations. The knowledge base and resource database will be different according to the variables and features of the application. However, the architecture of the chatbot should still follow a similar pattern.

The chatbot has the potential to be improved by integrating machine learning tasks like intent prediction, natural language understanding, and neural response generation. However, vast amount of data is needed to integrate these machine learning tasks. Without enough labelled and expert validated data, machine learning algorithms cannot be used to improve the chatbot. Reinforcement learning can also be potentially implemented to improve the chatbot performance by learning from the feedback of users, and making better decisions with experience. Currently, feedback from a user is saved when a resource is recommended (see Figure A.2). With data from real users, it is possible for the chatbot to adapt to users' preferences.

However, in order to gather useful data, we have to conduct experiments that involve actual subjects suffering from NDD. We have not yet been able to perform those experiments as it requires additional funding and initiatives taken by experts in the field. Future improvements should focus on collecting actual conversation data using our interface, learning from that data to perform machine-learned tasks and produce appropriate recommendations given a set of subject facts. Open source datasets are not available and few hospitals have taken the initiative to gather conversational data from chatbots. We can potentially improve the order of the questions by emulating how a doctor asks questions to a subject. Currently, we implement a naive information gain algorithm to ask questions pertaining to website facts that occur most frequently in the recommendations. This procedure may not optimal and ideal for a user.

Since not all queries can lead to a recommendation from the resource database, the Question Answering System can be improved by including a query expansion that resubmits the query to the website database. In general, it is a query relaxation [19, 20] technique that is used to relax some of the constraints in the query. For example, if no website is found for a given location, the location may be relaxed all together or generalised along a geographical location hierarchy from city to state/region or country.

In order to improve the state of the interface, it is useful to fix the six issues mentioned in the previous chapter, leading to a second heuristic evaluation that will reveal whether there are user-centred heuristic violations. A user-friendly, easy to use interface is the first step before collecting user data using the chatbot interface.

Future attempts may include using the expert-system framework on a different application and building a chatbot using the system and process shown in this thesis. That may show whether our framework is suitable to be implemented for other applications and use cases. Some features of the design of CAMI included in Chapter 3 are not implemented because the project took a direction of implementation different from expert-systems, and thereby the implementation of any planned features would not have been used in the final version of CAMI. Table 5.1 makes a comparison of the list of discussed features between the current version and planned future version. Currently, the chatbot provides a resource, if found, when the SQL query to the resource database completely matches all the subject facts with all the resource attributes. However, we can modify the rules of the expert system easily as presented in Chapter 3 and produce SQL queries so that a resource is recommended when some resource attributes match subject facts. We have presented the design of such rules in Chapter 3; however, we leave its full implementation for the future. Appendix A.0.1 shows all the currently implemented rules. Chapter 3 also presents design of Query relaxation, Query tightening and Intermediary rules that can be implemented for CAMI in the future. Chapter 4 presents the mockups of UI, edited from actual images of example runs of the chatbot, intended for a future version of CAMI.

Features	Current Version	Future Version
Recommendations with exact match	✓	✓
Recommendations with a partial match	✗	✓
Personalised recommendations	✓	✓
Query relaxation	✗	✓
Query tightening	✗	✓
Textual answers from users	✗	✓
Intermediary rules	✗	✓
Implementation of UI Mockups	✗	✓

Table 5.1: This table shows the list of features for both the current version and planned future version of CAMI.

Although it is possible to develop our chatbot as a rule-based system instead of an expert-system, the reusability and extendibility of expert-systems may be missed. It is very easy to modify rules within the expert-system and implement forward and backward chaining. However, with a rule-based system, one has to develop the inference engine from scratch with functional code that may be difficult to maintain and improve upon in the long run.

References

[1] Bayan AbuShawar and Eric Atwell. "ALICE chatbot: trials and outputs." In: *Computación y Sistemas* 19.4 (2015), pp. 625–632. 2

[2] James Allen. *Natural language understanding*. Pearson, 1995. 1

[3] Amazon. *The Amazon Alexa prize and dataset*. 2019. URL: `https://developer.amazon.com/alexaprize` (visited on 06/30/2019). 32

[4] Andrea Andrenucci and Eriks Sneiders. "Automated question answering: Review of the main approaches." In: *Third International Conference on Information Technology and Applications (ICITA'05)*. Vol. 1. IEEE. 2005, pp. 514–519. 10

[5] Suket Arora, Kamaljeet Batra, and Sarabjit Singh. "Dialogue system: A brief review." In: *arXiv preprint arXiv:1306.4134* (2013). 10

[6] Sören Auer et al. "Dbpedia: A nucleus for a web of open data." In: *The semantic web*. Springer, 2007, pp. 722–735. 13

[7] Ricardo Baeza-Yates, Berthier de Araújo Neto Ribeiro, et al. *Modern information retrieval*. New York: ACM Press; Harlow, England: Addison-Wesley, 2011. 1, 21

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." In: *arXiv preprint arXiv:1409.0473* (2014). 24

[9] Satanjeev Banerjee and Alon Lavie. "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments." In: *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. 2005, pp. 65–72. 31

[10] Antoine Bordes, Y-Lan Boureau, and Jason Weston. "Learning end-to-end goal-oriented dialog." In: *arXiv preprint arXiv:1605.07683* (2016). 19, 33

[11] Sergey Brin and Lawrence Page. "The anatomy of a large-scale hypertextual web search engine." In: *Computer networks and ISDN systems* 30.1-7 (1998), pp. 107–117. 14, 22

[12] Bruce Buchanan. "Rule based expert systems." In: *The MYCIN Experiments of the Stanford Heuristic Programming Project* (1984). 3

[13] Christian Buck et al. "Ask the right questions: Active question reformulation with reinforcement learning." In: *arXiv preprint arXiv:1705.07830* (2017). 2, 24, 25, 33–36

[14] Nan-chen Chen et al. "Challenges of applying machine learning to qualitative coding." In: *ACM SIGCHI Workshop on Human-Centered Machine Learning. http://hcml2016. goldsmithsdigital. com/program.* 2016. 24

[15] David Cochran. *Twitter bootstrap web development how-to*. Packt Publishing Ltd, 2012. 58

[16] Antonio R Damasio and Ralph G Maurer. "A neurological model for childhood autism." In: *Archives of neurology* 35.12 (1978), pp. 777–786. 4

[17] Acolatse Dela Kwami. *Web-based pharmacist expert system for supporting community pharmacy practice – premier point chemists*. Senior Project thesis. 2018. URL: `http://hdl.handle.net/20.500.11988/418`. 3, 32

[18] Kerstin Denecke et al. "Self-anamnesis with a conversational user interface: concept and usability study." In: *Methods of information in medicine* 57.05/06 (2018), pp. 243–252. 29

[19] Arianna D'Ulizia, Fernando Ferri, and Patrizia Grifoni. "Query Relaxation in Cooperative Query Processing." In: *Methods and Supporting Technologies for Data Analysis. Studies in Computational Intelligence*. Ed. by Zakrzewska D., Menasalvas E., and Byczkowska-Lipinska L. Berlin, Heidelberg: Springer, 2009, pp. 167–185. 49, 85

[20] Arianna D'Ulizia et al. "Relaxing Constraints on GeoPQL Operators to Improve Query Answering." In: *International Conference on Database and Expert Systems Applications (DEXA)*. Lecture Notes in Computer Science, vol 4080. Springer, 2006, pp. 728–737. 49, 85

[21] Matthew Dunn et al. "SearchQA: A new Q&A dataset augmented with context from a search engine." In: *arXiv preprint arXiv:1704.05179* (2017). 24, 26

[22] J Durinck et al. "The role of knowledge based systems in clinical practice." In: *Knowledge and decisions in health telematics: The next decade* (1994), pp. 199–203. 26

[23] George Ferguson, James F Allen, Bradford W Miller, et al. "TRAINS-95: Towards a Mixed-Initiative Planning Assistant." In: *AIPS*. 1996, pp. 70–77. 30

[24] Christopher Brian Fleizach and Reginald Dean Hudson. *Intelligent text-to-speech conversion*. US Patent 8,996,376. Mar. 2015. 28

[25] Ernest J Friedman-Hill. *Jess, the java expert system shell*. Tech. rep. Sandia Labs., Livermore, CA (United States), 1997. 34

[26] Robert W Gehl. "Teaching to the Turing test with Cleverbot." In: *Transformations: The Journal of Inclusive Scholarship and Pedagogy* 24.1-2 (2014), pp. 56–66. 18

[27] Abby A Goodrum. "Image information retrieval: An overview of current research." In: *Informing Science* 3.2 (2000), pp. 63–66. 22

[28] Miguel Grinberg. *Flask web development: developing web applications with python.* " O'Reilly Media, Inc.", 2018. 56

[29] Ouriel Grynszpan et al. "Innovative technology-based interventions for autism spectrum disorders: a meta-analysis." In: *Autism* 18.4 (2014), pp. 346–361. 4

[30] Jiaxian Guo et al. "Long text generation via adversarial training with leaked information." In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018. 16

[31] B Herawan Hayadi et al. "Expert system of quail disease diagnosis using forward chaining method." In: *Indonesian Journal of Electrical Engineering and Computer Science* 5.1 (2017), pp. 207–214. 38

[32] Ulf Hermjakob, Eduard H Hovy, and Chin-Yew Lin. "Knowledge-based question answering." In: *Proceedings of the Sixth World Multiconference on Systems, Cybernetics, and Informatics (SCI-2002)*. 2000. 1, 17, 26

[33] Budditha Hettige and Asoka S Karunananda. "First Sinhala chatbot in action." In: *Proceedings of the 3rd Annual Sessions of Sri Lanka Association for Artificial Intelligence (SLAAI), University of Moratuwa* (2006). 34

[34] Kate S Hone and Robert Graham. "Towards a tool for the subjective assessment of speech system interfaces (SASSI)." In: *Natural Language Engineering* 6.3-4 (2000), pp. 287–303. 30

[35] Abraham Ittycheriah and Salim Roukos. *IBM's statistical question answering system-TREC-11.* Tech. rep. IBM THOMAS J WATSON RESEARCH CENTER YORKTOWN HEIGHTS NY, 2006. 10

[36] Peter Jackson. *Introduction to expert systems.* Addison-Wesley Longman Publishing Co., Inc., 1998. 3, 38

[37] Zongcheng Ji, Zhengdong Lu, and Hang Li. "An information retrieval approach to short text conversation." In: *arXiv preprint arXiv:1408.6988* (2014). 15, 21

[38] BR Kavitha and Chethana R Murthy. "Chatbot for healthcare system using Artificial Intelligence." In: *Int J Adv Res Ideas Innov Technol* 5 (2019), pp. 1304–1307. 33

[39] Veton Këpuska and Gamal Bohouta. "Next-generation of virtual personal assistants (microsoft cortana, apple siri, amazon alexa and google home)." In: *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE. 2018, pp. 99–103. 1, 28

[40] Evangelos Kotsakis. "Structured information retrieval in XML documents." In: *Proceedings of the 2002 ACM symposium on Applied computing*. ACM. 2002, pp. 663–667. 21

[41] Tobias Kowatsch et al. "The impact of interpersonal closeness cues in text-based healthcare chatbots on attachment bond and the desire to continue interacting: an experimental design." In: (2018). 29

[42] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems*. 2012, pp. 1097–1105. 22

[43] SP Leo Kumar. "Knowledge-based expert system in manufacturing planning: state-of-the-art review." In: *International Journal of Production Research* 57.15-16 (2019), pp. 4766–4790. 38

[44] Hector J Levesque. *Common sense, the Turing test, and the quest for real AI*. MIT Press, 2017. 1

[45] Bo Li et al. "Acoustic Modeling for Google Home." In: *Interspeech*. 2017, pp. 399–403. 28

[46] Jiwei Li et al. "A persona-based neural conversation model." In: *arXiv preprint arXiv:1603.06155* (2016). 31

[47] Jiwei Li et al. "Deep reinforcement learning for dialogue generation." In: *arXiv preprint arXiv:1606.01541* (2016). 17

[48] Chia-Wei Liu et al. "How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation." In: *arXiv preprint arXiv:1603.08023* (2016). 31

[49] Fei Liu and Julien Perez. "Gated end-to-end memory networks." In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. 2017, pp. 1–10. 20

[50] Gustavo López, Luis Quesada, and Luis A Guerrero. "Alexa vs. Siri vs. Cortana vs. Google assistant: a comparison of speech-based natural user interfaces." In: *International Conference on Applied Human Factors and Ergonomics*. Springer. 2017, pp. 241–250. 1, 28

[51] Ramón López-Cózar et al. "Review of spoken dialogue systems." In: *Loquens* 1.2 (2014), p. 012. 10

[52] Helen McConachie and Tim Diggle. "Parent implemented early intervention for young children with autism spectrum disorder: A systematic review." In: *Journal of evaluation in clinical practice* 13.1 (2007), pp. 120–129. 4

[53] William van Melle, Edward H Shortliffe, and Bruce G Buchanan. "EMYCIN: A knowledge engineer's tool for constructing rule-based expert systems." In: *Rule-based expert systems: The MYCIN experiments of the Stanford Heuristic Programming Project* (1984), pp. 302–313. 3

[54] Sewon Min, Minjoon Seo, and Hannaneh Hajishirzi. "Question Answering through Transfer Learning from Large Fine-grained Supervision Data." In: *Association for Computational Linguistics (ACL)*. 2018. 2

[55] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning." In: *arXiv preprint arXiv:1312.5602* (2013). 23

[56] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning." In: *Nature* 518.7540 (2015), p. 529. 23

[57] Sana Mujeeb, Muhammad Hafeez, and Tayyaba Arshad. "Aquabot: a diagnostic chatbot for achluophobia and autism." In: *Int J Adv Comput Sci Appl* 8.9 (2017), pp. 39–46. 33

[58] Jakob Nielsen. "Usability inspection methods." In: *Conference companion on Human factors in computing systems*. ACM. 1994, pp. 413–414. 79

[59] Jakob Nielson and Rolf Molich. "Heuristic evaluation of user interfaces." In: *Proceedings of ACM CHI'90 Conference*. 1990, pp. 1–5. 9, 76

[60] nilp0inter. *Experta: Expert systems for python*. URL: `https://github.com/nilp0inter/experta`. 33, 49

[61] Joel Nothman, James R Curran, and Tara Murphy. "Transforming Wikipedia into named entity training data." In: *Proceedings of the Australasian Language Technology Association Workshop 2008*. 2008, pp. 124–132. 21

[62] David Oniani and Yanshan Wang. "A Qualitative Evaluation of Language Models on Automatic Question-Answering for COVID-19." In: *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*. 2020, pp. 1–9. 24

[63] World Health Organization. *Neurological disorders: public health challenges*. World Health Organization, 2006. 4

[64] Maria B Ospina et al. "Behavioural and developmental interventions for autism spectrum disorder: a clinical systematic review." In: *PloS one* 3.11 (2008), e3755. 4

[65] Kishore Papineni et al. "BLEU: a method for automatic evaluation of machine translation." In: *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics. 2002, pp. 311–318. 31

[66] Alex Park, Timothy J Hazen, and James R Glass. "Automatic processing of audio lectures for information retrieval: Vocabulary selection and language modeling." In: *Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.* Vol. 1. IEEE. 2005, pp. I–497. 22

[67] Linda Dailey Paulson. “Building rich web applications with Ajax.” In: *Computer* 38.10 (2005), pp. 14–17. 58

[68] Rajpurkar Pranav et al. “SQuAD: 100,000+ Questions for Machine Comprehension of Text.” In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2016, pp. 2383–2392. 2

[69] Ashwin Ram et al. “Conversational ai: The science behind the alexa prize.” In: *arXiv preprint arXiv:1801.03604* (2018). 1, 8

[70] S Reddy, D Chen, and CD Manning. “Coqa: A conversational question answering challenge.” In: *CoRR arXiv:1808.07042* (2018). 2

[71] Ehud Reiter and Robert Dale. *Building natural language generation systems*. Cambridge university press, 2000. 2

[72] Bradley Rhodes and Thad Starner. “Remembrance Agent: A continuously running automated information retrieval system.” In: *The Proceedings of The First International Conference on The Practical Application Of Intelligent Agents and Multi Agent Technology*. 1996, pp. 487–495. 21

[73] Sevalaraj Rupesh and Asoka S Karunananda. “Virtual Tour Agent.” In: *SRI LANKA ASSOCIATION FOR ARTIFICIAL INTELLIGENCE* (2006). 34

[74] Minjoon Seo et al. “Bidirectional attention flow for machine comprehension.” In: *arXiv preprint arXiv:1611.01603* (2016). 26

[75] Iulian Vlad Serban et al. “Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models.” In: *AAAI*. Vol. 16. 2016, pp. 3776–3784. 1

[76] Lifeng Shang, Zhengdong Lu, and Hang Li. “Neural responding machine for short-text conversation.” In: *arXiv preprint arXiv:1503.02364* (2015). 16

[77] Lifeng Shang et al. “Overview of the NTCIR-12 Short Text Conversation Task.” In: *NTCIR*. 2016. 15

[78] Abu Shawar, Eric Atwell, and Andrew Roberts. “Faqchat as in information retrieval system.” In: *Human Language Technologies as a Challenge for Computer Science and Linguistics: Proceedings of the 2nd Language and Technology Conference*. Poznań: Wydawnictwo Poznańskie: with co-operation of Fundacja Uniwersytetu ... 2005, pp. 274–278. 2, 23

[79] Bayan Abu Shawar and Eric Atwell. *A comparison between Alice and Elizabeth chatbot systems*. University of Leeds, School of Computing research report 2002.19, 2002. 2

[80] Bayan Abu Shawar and Eric Atwell. “Using dialogue corpora to train a chatbot.” In: *Proceedings of the Corpus Linguistics 2003 conference*. 2003, pp. 681–690. 2

[81] Heung-Yeung Shum, Xiao-dong He, and Di Li. "From Eliza to XiaoIce: challenges and opportunities with social chatbots." In: *Frontiers of Information Technology & Electronic Engineering* 19.1 (2018), pp. 10–26. 2

[82] Ajay KS Singholi and Divya Agarwal. "Review of expert system and its application in robotics." In: *Intelligent communication, control and devices*. Springer, 2018, pp. 1253–1265. 38

[83] Tristram Smith et al. "Designing research studies on psychosocial interventions in autism." In: *Journal of autism and developmental disorders* 37.2 (2007), pp. 354–366. 4

[84] Marco Antonio Calijorne Soares and Fernando Silva Parreiras. "A Literature Review on Question Answering Techniques, Paradigms and Systems." In: *Journal of King Saud University-Computer and Information Sciences* (2018). 10

[85] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. "End-to-end memory networks." In: *Advances in neural information processing systems*. 2015, pp. 2440–2448. 17, 19, 33–36

[86] Haitian Sun et al. "Open domain question answering using early fusion of knowledge bases and text." In: *arXiv preprint arXiv:1809.00782* (2018). 2

[87] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. "Sequence to Sequence Learning with Neural Networks." In: *Advances in Neural Information Processing (NIPS)*. 2014. 2

[88] Anu Venkatesh et al. "On Evaluating and Comparing Open Domain Dialog Systems." In: *arXiv preprint arXiv:1801.03625* (2018). 31, 32

[89] Oriol Vinyals and Quoc Le. "A neural conversational model." In: *arXiv preprint arXiv:1506.05869* (2015). 2, 12, 15, 17, 18, 20, 33–

[90] Christiane Völkle and Patrick Planing. "Digital Automation of Customer Contact Processes–an Empirical Research on Customer Acceptance of different Chatbot Use-cases." In: *Digitalen Wandel gestalten*. Springer, 2019, pp. 217–229. 29

[91] Ellen M Voorhees and Dawn M Tice. "The TREC-8 question answering track evaluation." In: *TREC*. Vol. 1999. Citeseer. 1999, p. 82. 27

[92] Ellen M Voorhees, Donna K Harman, et al. *TREC: Experiment and evaluation in information retrieval*. Vol. 63. MIT press Cambridge, 2005. 27

[93] Marilyn Walker, Candace Kamm, and Diane Litman. "Towards developing general models of usability with PARADISE." In: *Natural Language Engineering* 6.3-4 (2000), pp. 363–377. 30

[94] Florian von Wangenheim and Julian Ventouris. “Conversational Agents in Healthcare: Where Are We Going?” In: *Perspektiven des Dienstleistungsmanagements*. Springer, 2020, pp. 677–697. 29

[95] Donald A. Waterman. *A Guide toe Expert Systems*. Addison-Wesley, 1986. 3, 38–40

[96] Joseph Weizenbaum. “ELIZA—a computer program for the study of natural language communication between man and machine.” In: *Communications of the ACM* 9.1 (1966), pp. 36–45. 2

[97] Jason E Weston. “Dialog-based language learning.” In: *Advances in Neural Information Processing Systems*. 2016, pp. 829–837. 20

[98] Jason Weston, Sumit Chopra, and Antoine Bordes. “Memory Networks.” In: *CoRR* abs/1410.3916 (2014). arXiv: `1410.3916`. URL: `http://arxiv.org/abs/1410.3916`. 19

[99] Jason Weston et al. “Towards ai-complete question answering: A set of prerequisite toy tasks.” In: *arXiv preprint arXiv:1502.05698* (2015). 20, 30

[100] Jan Wielemaker et al. “Swi-prolog.” In: *Theory and Practice of Logic Programming* 12.1-2 (2012), pp. 67–96. 34

[101] Jason Williams, Antoine Raux, and Matthew Henderson. “The dialog state tracking challenge series: A review.” In: *Dialogue & Discourse* 7.3 (2016), pp. 4–33. 10

[102] Robin Williams. *The non-designer’s design book: Design and typographic principles for the visual novice*. Pearson Education, 2015. 79

[103] Caiming Xiong, Stephen Merity, and Richard Socher. “Dynamic memory networks for visual and textual question answering.” In: *International conference on machine learning*. 2016, pp. 2397–2406. 20

[104] Zhao Yan et al. “Docchat: An information retrieval approach for chatbot engines using unstructured documents.” In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vol. 1. 2016, pp. 516–525. 22, 33–36

Appendix A

Appendix

A.0.1 Knowledge Base Rules

We implemented 16 different rules for the expert-system. All the rules are as follows:

"If Autism *IN* **condition** *AND*
  Physical Aggression *IN* **challenging behaviour** *AND*
  age *IS PRESENT AND*
  intervention *IS PRESENT AND*
  location *IS PRESENT*,
 ATTRIBUTES FOR QUERY"

"If Autism *IN* **condition** *AND*
  Non-Compliance *IN* **challenging behaviour** *AND*
  age *IS PRESENT AND*
  intervention *IS PRESENT AND*
  location *IS PRESENT*,
 ATTRIBUTES FOR QUERY"

"If Autism *IN* **condition** *AND*
  Self-Harm *IN* **challenging behaviour** *AND*
  age *IS PRESENT AND*
  intervention *IS PRESENT AND*

location *IS PRESENT*,

ATTRIBUTES FOR QUERY"

"If Autism *IN* **condition** *AND*

Temper Tantrums *IN* **challenging behaviour** *AND*

age *IS PRESENT AND*

intervention *IS PRESENT AND*

location *IS PRESENT*,

ATTRIBUTES FOR QUERY"

"If Neurodevelopmental Delay *IN* **condition** *AND*

Physical Aggression *IN* **challenging behaviour** *AND*

age *IS PRESENT AND*

intervention *IS PRESENT AND*

location *IS PRESENT*,

ATTRIBUTES FOR QUERY"

"If Neurodevelopmental Delay *IN* **condition** *AND*

Non-Compliance *IN* **challenging behaviour** *AND*

age *IS PRESENT AND*

intervention *IS PRESENT AND*

location *IS PRESENT*,

ATTRIBUTES FOR QUERY"

"If Neurodevelopmental Delay *IN* **condition** *AND*

Self-Harm *IN* **challenging behaviour** *AND*

age *IS PRESENT AND*

intervention *IS PRESENT AND*

location *IS PRESENT*,

ATTRIBUTES FOR QUERY"

"If Neurodevelopmental Delay *IN* **condition** *AND*

Temper Tantrums *IN* **challenging behaviour** *AND*
age *IS PRESENT AND*
intervention *IS PRESENT AND*
location *IS PRESENT*,
ATTRIBUTES FOR QUERY"

"If Intellectual Disability *IN* **condition** *AND*
Physical Aggression *IN* **challenging behaviour** *AND*
age *IS PRESENT AND*
intervention *IS PRESENT AND*
location *IS PRESENT*,
ATTRIBUTES FOR QUERY"

"If Intellectual Disability *IN* **condition** *AND*
Non-Compliance *IN* **challenging behaviour** *AND*
age *IS PRESENT AND*
intervention *IS PRESENT AND*
location *IS PRESENT*,
ATTRIBUTES FOR QUERY"

"If Intellectual Disability *IN* **condition** *AND*
Self-Harm *IN* **challenging behaviour** *AND*
age *IS PRESENT AND*
intervention *IS PRESENT AND*
location *IS PRESENT*,
ATTRIBUTES FOR QUERY"

"If Intellectual Disability *IN* **condition** *AND*
Temper Tantrums *IN* **challenging behaviour** *AND*
age *IS PRESENT AND*
intervention *IS PRESENT AND*
location *IS PRESENT*,

ATTRIBUTES FOR QUERY"

"If ADHD *IN* **condition** *AND*
Non-Compliance *IN* **challenging behaviour** *AND*
age *IS PRESENT AND*
intervention *IS PRESENT AND*
location *IS PRESENT*,
ATTRIBUTES FOR QUERY"

"If ADHD *IN* **condition** *AND*
Self-Harm *IN* **challenging behaviour** *AND*
age *IS PRESENT AND*
intervention *IS PRESENT AND*
location *IS PRESENT*,
ATTRIBUTES FOR QUERY"

"If ADHD *IN* **condition** *AND*
Temper Tantrums *IN* **challenging behaviour** *AND*
age *IS PRESENT AND*
intervention *IS PRESENT AND*
location *IS PRESENT*,
ATTRIBUTES FOR QUERY"

"If ADHD *IN* **condition** *AND*
Physical Aggression *IN* **challenging behaviour** *AND*
age *IS PRESENT AND*
intervention *IS PRESENT AND*
location *IS PRESENT*,
ATTRIBUTES FOR QUERY"

A.0.2 Current implementation of CAMI interface

In order to provide the current implementation of the interface, we decided to follow a progression of conversation that resulted in a recommendation by choosing the Professional intervention. The 211 support group website should not be recommended under the Professional intervention route. Figures A.1 and A.2 show a conversation sequence between the chatbot and an imaginary user who is looking for Professional intervention, representing a child suffering from intellectual disability and exhibiting physical aggression and self-injury. The sequence of conversation led to the recommendation of a different website that already existed in the knowledge base. This website is annotated by an expert for a child or a teen, suffering from any of ADHD, Intellectual Disability, or Autism, exhibiting physical aggression, self-injury, temper tantrums and/or non-compliance, and living in Edmonton.

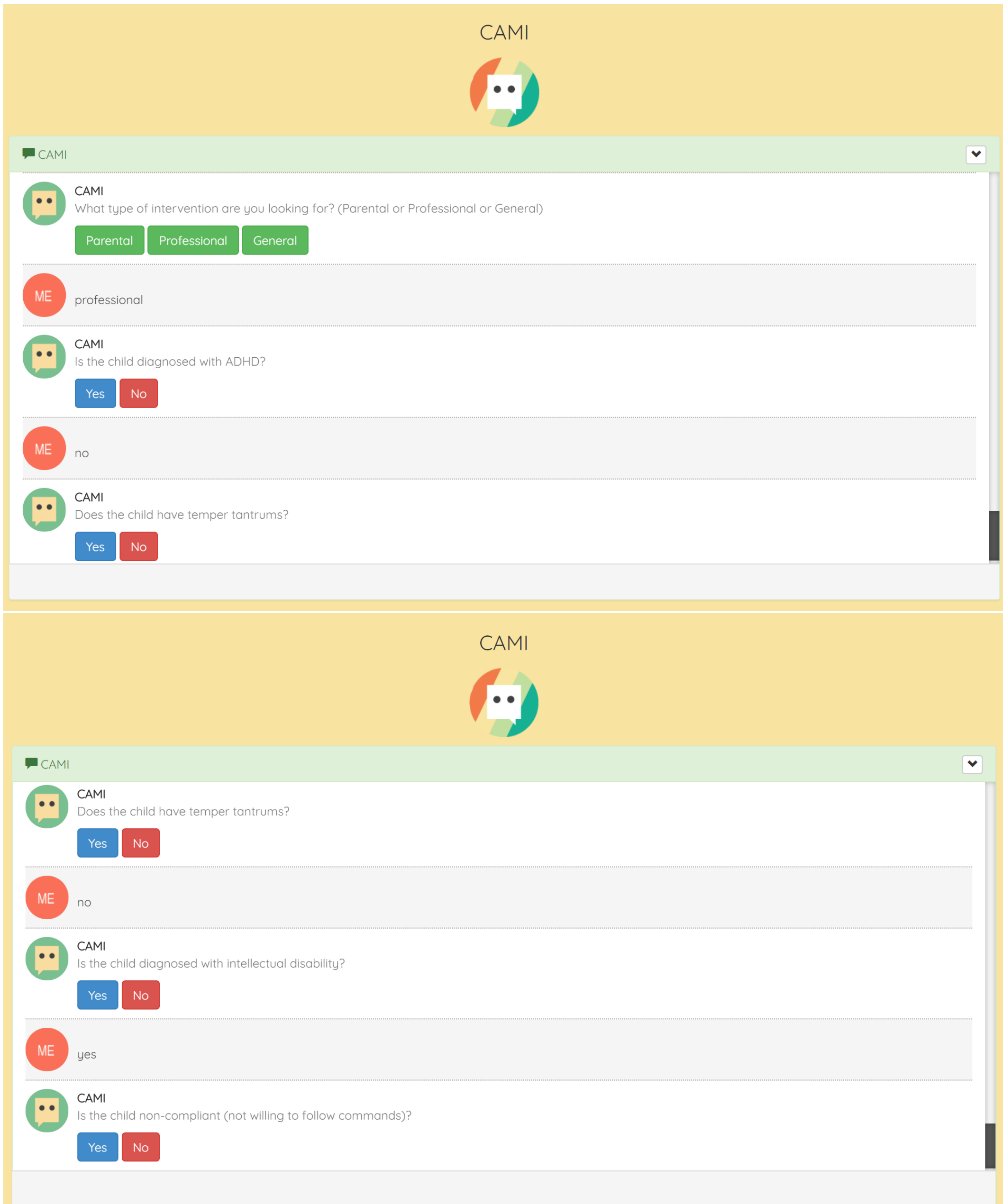


Figure A.1: The progression of a conversation between the chatbot and a user under the Professional type of intervention.

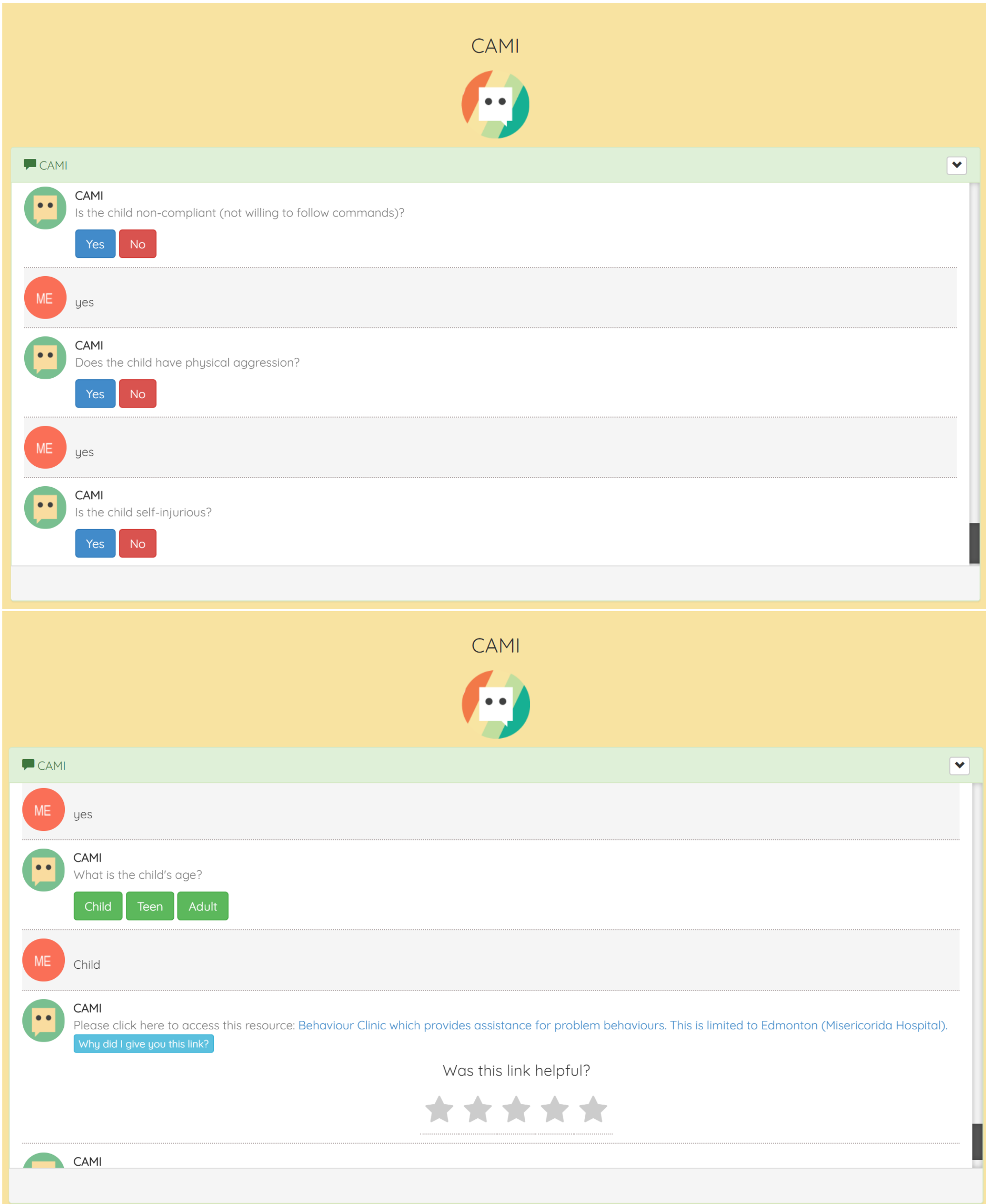


Figure A.2: The progression of a conversation between the chatbot and a user under the Professional intervention.

	Description (Nature of Violation)
4	Freedom if I press the back button on the browser, I can see the conversation "reversing", but then if I type a new message after that, it goes back to the beginning of the conversation, erases the messages I had typed previously, and says that it does not understand my message (even if the message I typed makes sense in the context of where I am). Going back from this point likewise "rewinds" the old conversation again.
6	There is no help or how to use the system guide, which could be difficult for users who don't have a lot of experience with chatbots.
33	The question "Which type of intervention are you looking for? (Parental or Professional or General)" is confusing. It is not clear what the difference is or what the purpose of this question it.
10	Some of the language used may be too complicated for users. For example, a question like "Has your child been diagnosed with 'non-specific developmental delay'?" has terms like 'non-specific developmental delay' that the user may not know the meaning of. Note that there are other places where it is explained further (Is your child non-compliant (not willing to follow commands)?)
20	Some of the information at the beginning of the test is too much and hard to follow
29	The words "consent" and "assent" are both used here in seemingly interchangeable ways, which could be confusing because the user might wonder if there is a difference even though they are essentially synonyms. Also, it's unclear who the user is from this. Some of it seems like the child would be using this chatbot but the later chat questions seems like child-use is inappropriate.

Table A.1: This table describes the nature of the heuristic violations that we were unable to fix and left for future improvements.