Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

WOzBot : A Wizard of Oz Based Method for Chatbot Response Improvement

by

Chaitanya Joglekar

Under supervision of Prof. Vincent Wade

August 19, 2022

A dissertation presented in fulfillment of the requirements for the Degree of Master in Computer Science

Declaration Of Authorship

I, Chaitanya Joglekar , declare that the following dissertation, except where otherwise stated, is entirely my own work; that it has not previously been submitted as an exercise for a degree, either in Trinity College Dublin, or in any other University; and that the library may lend or copy it or any part thereof on request.

Signed: _____ Date: _____

Acknowledgements

I would like to sincerely thank Prof.Vincent Wade for his immense support and valuable guidance throughout the year. Prof.Vincent Wade consistently encouraged me to give my best and leverage my own ideas and knowledge in this research while always helping me with all the technical as well as other administrative work that was pertinent in bringing this research work to fulfillment.

I would also like to extend special thanks and appreciation to Mrs.Anna Attwood for taking part as an expert user in the experiment conducted which was a crucial part of the research. I appreciate the time and valuable input from her throughout this process which helped me in successfully completing the experiment.

Last but not the least, special thanks to my parents for their continuous support and encouragement in all possible forms throughout the master's course which has helped me to pursue and successfully accomplish my career goals.

Abstract

With the exponential surge in the use and popularity of Artificial Intelligence, Conversational Agents also known as Chatbots are becoming prevalent as they are being used across innumerable domains and applications.Today chatbots and virtual assistant systems like Siri, Alexa, Google Assistants are becoming an intrinsic part of human lives and becoming more human alike. However, with the adoption of conversational bots in various domains, developing highly efficient and engaging bots from scratch remains difficult and demands extensive supervision from human developers. This could be due to a variety of factors, such a scarcity of domain specific training data as collecting the data of sufficient quantity and quality can take a lot of time and cost, a lack of domain knowledge to efficiently respond to user queries/questions, or undefined scope of the agents.

This work proposes a system based on the Conversation-Driven Design concept for iteratively developing conversational agents (Chatbots). This essentially gives a conversational agent the ability to learn and adapt from actual user conversations. The proposed system's architecture modifies the traditional Wizard of Oz (WOz) technique in order to incrementally capture the domain-specific chatbot corpus while also improving the conversational agent's responses based on input from a domain expert user acting as a Wizard.

The experimental results and evaluation presented in this work corroborates the idea that the proposed system can be used efficiently to iteratively evolve conversational agents in closed domain applications to make them more engaging, intuitive, and human-like.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the main idea and motivation behind the research. This also defines a research question and the research objectives proposed to address it. Finally, it provides a brief overview of the technical approach proposed in this research, as well as an outline illustrating the overall structure of the report.

1.1 Motivation

Conversation in natural language is an essential component of human evolution and culture. Indeed, language is the distinguishing feature of the human species in general (1). Today, thanks to advances in fields such as Artificial Intelligence and, more specifically, Natural Language Processing, it is even possible to create a computer program that can understand and converse in natural language. Traditionally, these programs have been referred to as "Conversational Agents" or "Digital Assistants" (1). These programs typically interact with the end user through text or speech. The ideation and development of the Virtual Assistants started more than 50 years ago with dialog systems like ELIZA (2) , PARRY (3) which are static rule-based systems with no intelligence. But with tremendous research and progress in this area, today digital assistants like Siri, Alexa, and Google assistant are becoming ubiquitous and can perform a variety of tasks such as controlling home appliances, assisting users in finding directions and locations, playing music, and so on. These agents are broadly classified under the category of task-based agents as they are designed to perform a specific set of tasks. Most of the agents/bots implemented today like a chatbot on a banking website come under this category as they perform a fixed set of tasks like a money transfer.

However, today Conversational agents are being used across innumerable domains and applications, and in many of these domains, these agents are designed to have a natural extended conversation with the user or to answer arbitrary questions from the user rather than focusing on a fixed set of tasks. Typically these types of agents are being adopted in domains like historical museums, art galleries, and libraries to provide an engaging channel to interact with the users. But building such a closed domain conversational agent from a scratch is highly challenging because of the following reasons -

- Scarcity of quality domain specific training data that is the type of queries or questions users can ask.

- Lack of domain knowledge for generating coherent and engaging responses.
- Development phases may take significant time inhibiting testing of the agent in real-world conversations.

This creates the need for a system that will allow to iteratively gather new training data, improve the responses of the agents, and be able to test the agent in an early phase of development. Some common approaches like (4) suggest training bots/agents on a huge corpus of crowdsourced data, but collecting such a large corpus of data demands extensive time and effort and they may not reflect the dialogs or a set of words that can occur in domain specific to an agent (5).

Some state-of-the-art methods like (6) uses similar Wizard of Oz (WOz)(7, 8) based methods on speech interfaces rather than conversational interface. Other state-of-the-art methods (9, 10, 11) which have proposed similar iterative designs of the agents using the concept of crowd-sourcing for gathering training data and responses are either too complex or fail to define a common technique to address all the main challenges listed previously.

Thus this work fills the inherent gap in state-of-the-art research by proposing a system to enable conversational agents to iteratively learn and adapt throughout their development. The proposed system addresses all the main challenges faced in the development of conversational agents by enabling incremental capturing of the domain-specific corpus (gathering what question user might ask about through crowd-sourcing), incorporating domain knowledge to process and respond to the user questions (by allowing domain experts to analyze and correct how the bot responses to those questions) and leverage it into a conversational model (retrain the model based on expert wisdom and data).

1.2 Research Question

This dissertation poses a question of how to design a methodology/system for iterative design and improvement of Conversational Agents (Chatbot).

1.3 Research Objective

This work has identified the following research objectives to address the research question posed -

1. To design and implement a baseline Q&A agent/bot using an open source chat framework.
2. To design a platform for conducting a Wizard of Oz based experiment (7, 8) to simultaneously gather the new training data from the end user and improve the responses of the bot based on inputs of an expert user (wizard).
3. To design a methodology to track the end-user conversations and use them along with expert user inputs for iterative training of the agent.

4. To evaluate the proposed system through the feedback provided by the end users based on their interaction with the agent, and feedback provided by the expert users.

1.4 Implementation Overview

To achieve the above-mentioned research objectives, this work has implemented a baseline chatbot using a Rasa open source bot framework (12). The domain of this conversational bot is related to the project "Beyond 22" [1] which is an international collaborative research project aimed at creating a Virtual Record treasury that will be the virtual recreation of Ireland's public record office which was destroyed during the opening engagement of a Civil War. So conversational bot implemented is a Q&A-type bot that would answer the questions of the users related to the specific aspects of the project "Beyond 22" such as the history of Ireland's public record office building, and the fire incident that destroyed the record office, etc. This work also has used some data cleaning and augmentation methods which can be used for bootstrapping the training data creation.

Further, this work has implemented a middleware system for conducting a modified "Wizard of Oz" (WOz) type experiment (7, 8). This middleware hosts two interfaces. A user interface allows the end users to interact with a bot and an Expert user interface allows domain experts to improve bot responses using domain knowledge. An iterative pipeline defined as part of the middleware is then used to continuously capture the end-user conversations and use them for further refinement and gathering of the training corpus of the bot and simultaneously use inputs from expert users to make agent's responses highly engaging and comprehensive. The proposed interface is evaluated by performing the experiment on real users by inviting them to converse with an agent. Also, historians involved in the project Beyond 2022 have been asked to act as a Wizard in this experiment.

1.5 Dissertation Outline

Chapter 2 contains a detailed discussion about the background and overall motivation behind the idea. It also gives a detailed overview of the existing research methods and the inherent gap left by the previous work which is fulfilled by this proposed work. Chapter 3 gives implementation details and overall architecture of the proposed system along with the intuition behind the technology choices. In Chapter 4 evaluation of the proposed methodologies has been presented along with experimental results. Finally, Chapter 5 discusses the research contribution of this work along with the research objective achieved. It also specifies the advantages as well as limitations of the proposed system finally defining the scope for future work.

[1] https://beyond2022.ie/

Chapter 2

State Of the Art

This chapter goes into great detail about the motivation for this research project. It also provides a comprehensive overview of the existing methods and techniques for iteratively developing and improving conversational agents. It concludes by highlighting an inherent gap in current research work and explaining how the proposed system facilitates filling that gap and addressing the proposed research question.

2.1 Background

Conversation in natural language is an essential component of human evolution and culture. Indeed, language is the distinguishing feature of the human species in general (1). Today with the revolution in fields like Artificial Intelligence and specifically in Natural Language Processing, it is even possible to build a computer program to understand and converse in natural language. Traditionally, these programs have been referred to as "Virtual Assistant" or "Digital Assistant" (1). The development of Virtual Assistants began more than 50 years ago with dialog systems such as ELIZA (2), PARRY (3), which are static rule-based systems with no intelligence. With continuous research and development in this domain, virtual assistant systems such as Siri, Alexa, Google Assistants, and chatbots are becoming an integral part of people's lives. These agents are broadly classified under the category of task-based agents as they are designed to perform a specific set of tasks. Most of the agents/bots implemented today like a chatbot on a banking website come under this category as they perform a fixed set of tasks like a money transfer.

However, today conversational agents or dialog systems are being adopted in innumerable domains including online learning (13), customer servicing (14). Even the traditional institutions like museums, art galleries and libraries are deploying these on their websites to provide an engaging way for their users to know about the various books and resources (15, 16). But in all these use cases conversation agent systems are designed to have a natural extended conversation with the user or to answer arbitrary questions from the user rather than focusing on a fixed set of tasks. So these agents also referred to as dialog agents are used to mimic human-to-human conversations. Despite being an advancement in technology and its widespread adoption building such conversational or dialog agents in any domain remains challenging. In fact, natural conversations between a human and these systems still remain far from natural and fail to engage the users (17). Many research in the field of Human-Computer

Interactions (17, 18) attributes this to poor quality of responses from the bot. As highlighted in the section this is due to the result of some of the challenges like -

- **Lack Of Training Data**
  The Efficiency of the Natural Language Understanding (NLU) model within conversational agents depends on the quality and quantity of training data which in turn depends on the amount of effort the developer of these systems invests to gather or create the conversational data. But in use cases where an agent is required to conduct a natural conversation with the end user and understand the answer to any free-flowing queries in the particular domain creating data from a scratch by anticipating different ways users can interact with a bot is almost impossible. Hence there is a need to iteratively gather the training data.

- **Lack of Domain Knowledge For Response Generation**
  Developers of the agents usually have to encode the responses to train the dialog management model so that an agent can learn how to respond in specific scenarios. Therefore there is a need to encode the domain knowledge in the responses to handle specific queries of users in the domain of the agent. But often developers of these agents don't have appropriate domain knowledge and have to involve domain users to create the responses.

Natural Language Understanding (NLU) engine and Dialog Management Engine are the two core parts of any conversational agent. The NLU engine is responsible for understanding the natural language and converting it into recognizable actions (intents) while the Dialog Management module is responsible for performing the appropriate actions and generating the responses. Different types of techniques are being in use to gather augment training and response data for the training of the NLU engine and dialog management model respectively. But these techniques differ based on the type of conversational agent. Conversational agents are broadly classified into rule-based and data-driven conversational agents (1). While on the one hand rule-based agents are generally used to develop the goal-oriented bot where all the possible user utterances and responses are encoded by the developers by manually defining domain-specific rules and mappings between the user query and a bot response. As a result, the natural language capabilities of such agents are limited and not extensible.

On the other hand, data-driven agents are trained using open-sourced conversational data (4) and instead of encoding rules, the user intents are identified using probabilistic similarity/matching with a training example. Similarly, responses are generated using the same technique by finding the best matching response. But the response generated may not be coherent and engaging as it depends on the quality of training samples. Also as highlighted in (5) the conversational corpus used for training may not contain the dialogs or a set of words that can occur in a domain specific to an agent.

So irrespective of the underlying category of the dialog systems, there is a need for a methodology to iteratively gather the training data (user queries) and also enable developers of the agent to involve the domain expert to gather and improve the agent's responses. However as the user plays a very critical role in any dialog system, these systems are closely related to the field of Human-Computer Interaction (HCI) (1). And thus many principles and techniques in the field of HCI can be leveraged to come up with such a methodology.

The methodology proposed by this work is thus based on some of the techniques in the field of HCI like the Wizard of Oz (7, 8, 19) system based on the principle of user-centered design as suggested in (20). The proposed system makes modifications to the traditional Wizard of Oz technique and makes use of conversational-driven design philosophy to build a platform to gather the training data from actual user conversations and also allows the domain experts to improve the responses of the agent at the same time. This will help to make the conversational agent more engaging and human-like.

2.2 Wizard Of Oz (WOz)

The Wizard of Oz (WOz) is a well-known method in the field of Human-Computer Interactions (HCI) for testing a prototype of an automated system in its early stages of development. The technique's name is derived from the story book called "The Wizard of Oz"(21) in which a magical Wizard is revealed to be nothing more than a human-controlled simulation. It is a technique in which end users are led to believe they are interacting with a computer system when, in fact, the system is mocked by a researcher or a human (Wizard) behind the interface (1, 7). As a result, the main advantage of this technique is that end-user interactions with a system can be interpreted and recorded without having to build/implement the system as Wizard can control the interactions without the involvement of any actual system.

As a result, Wizard of Oz is a common technique used to develop systems that lack standardized design patterns and necessitate predicting how end users will interact with or use the system (7, 19). To implement this technique, a research setup must include end-users interacting with a system in one location and researchers (wizards) simulating the system in another location via a separate wizard interface.

This technique is highly flexible and truly iterative in that, in the early stages of system development, a Wizard will simulate most of the system functionality, and then data recorded from earlier phases can be used to improve the system while minimizing the wizard's involvement and making the system truly automated (19, 22).

With its close relation to the field of HCI, this technique is used as a major tool to build dialog systems (8, 23, 24). In general, the wizard gets the user queries and can respond through a graphical interface. However, it is being majorly used as the method for collecting the training corpus in which results from Wizard of Oz systems are in turn used for training the dialog agents.

2.3 Conversation Driven Development

Past research (5, 17) has highlighted that the quality of conversational agents depends on the quality and quantity of training data including the encoded responses. But it is always difficult to create the training data from a scratch and that also in a large amount as required for training of the natural language models used in any conversational agents. Particularly a domain-specific chatbot training corpus must have words or jargon specific to that domain yet most of the open-sourced conversation data may not reflect it accurately.

There originates a need for a philosophy of Conversational Driven Development (CDD) which is based on the principle of user-centered design. This philosophy reflects the principle of learning from the actual user interactions (25, 26).That is to use interactions with the real users to gather the training data as well as to learn where agents are not performing well. This process generically contains the following actions (25, 26) which enables the iterative improvement of the conversational agent-

1. **Share**
   Developing a conversational agent from a scratch is quite challenging and may require a significant investment of time. So this action in the CDD principle requires sharing the prototype of the agents with users as early as possible to get to know how the agent performs in real-world conversation. And in such as way developers will be able to test early what can go wrong. Now some techniques like Wizard of Oz as discussed in previous section can be used to share the prototype of the agent with the end users.

2. **Review**
   The user conversations captured from early prototyping of such systems can act as a goldmine and can be used to create effective learning data for an agent. Hence the developers should invest the time to carefully review the conversation data.

3. **Annotate**
   Once reviewed appropriate techniques should be used to annotate the conversation data to turn it into a training example that can be used to train the NLU engine of the agent.

4. **Testing**
   End-to-end testing of the agent is always effective and required after each iteration and cycle of retraining of the agent. So After retraining the agent with conversational agent developers should create end-to-end test cases to test the agent.

5. **Track**
   This action corresponds to the evaluation of the overall performance and efficiency of the agent which will help to analyze how an agent is improving over time. Different parameters such as feedback from the end users can be used to perform the evaluation. Also, agents can use other implicit factors and self-learning for the evaluation.

2.4 Methods Based on Wizard Of Oz And Crowd-Sourcing

Wizard Of Oz With its close relation to the field of HCI, has been used as a major tool to build dialog systems across multiple research and studies. All of this research either proposes customized platforms and systems or leverages publicly available platforms like Amazon Mechanical Turk [1] to conduct Wizard Of Oz experiments through actual participation of the users (Crowdsourcing).

[1] `https://www.mturk.com/`

(24) uses the Wizard of Oz method to analyze how users of the agents would interact with agents having different personas like an emotional bot. The research has conducted a study with 14 users and has the individual expert users act as a wizard for each persona of the bot (emotional bot, fun bot, etc). However, results only reflect what users expect from these kinds of mood bots or bots who can perform chitchat. It does not highlight how this study can be used for the iterative design of the bot. Also, the results in the study are based on open-ended bots rather than a domain-specific agent.

(23) makes use of the Wizard Of Oz study to similarly gather different ways a user can interact with. Rather it focuses on a goal-based agent where the expert user acting as an agent were performing the task of booking a vacation as requested by the end user. As there is no involvement of an actual agent in this study there is no way to test out the agent in real-world conversation. Also as this study focus on goal/task-based agent, it is not applicable for either open-ended or domain-specific conversational agents.

The system proposed in (27) aligns with most of the research objectives proposed by this work. This study provides a platform for iteratively gathering the training corpus for the conversational agent. It provides an interface to end users who can perform conversations with a bot. In addition, it provides a separate interface for the privileged (Expert) users to annotate the recorded conversation. Though the main aim of the conversation annotation is to generate the data that would be suitable for the training of the agent. For this, the proposed system provides an easy and intuitive interface where an expert user can annotate the utterances and responses by scoring them in terms of the following factors -

- Subjectivity: Score for response coherency and suitability within a dialog context.
- Polarity: Score for indicating whether the text is negative, positive, or neutral in the linguistic sense.
- Offensiveness: Score for offensive words and phrases in utterances.
- Swear: Score for any vulgarity in utterances/responses.

Hence all the utterances and responses in the conversational data with higher annotations scores are filtered out. As described this work does use principles of the Wizard Of Oz technique but it does not allow the developers to engage the expert users to directly improve actual responses from the agent. This work also doesn't involve the actual agent in live conversation inhibiting its testing in the earlier phase.

Studies (28) show that to further improve the chatbot responses the Wizard of Oz type studies can be used where developers can create an interface to allow expert users to modify the response.

(6) proposes a system called "SUEDE" which defines the tool for performing the Wizard of Oz experiment. Though leverages this technique for designing of speech interface. In a way, it provides an intuitive interface that developers can use design the speech interface in an iterative manner in form of user prompts(speech) and script/response from the agent.

A system called "chorus" is proposed in (11). "Chorus" system provides an alternative interface for performing the Wizard of Oz experiment. In fact, it allows multiple users to act as expert users by integrating it with a crowd-sourcing platform. It uses a voting mechanism in order to attain coordination between multiple expert users and to generate the single best possible response to user queries but the overall voting process can be slow and also only works when a minimum of one other expert user has voted for the particular response. Also, this system takes into consideration tasks-based agents where the expert user just assists to complete the user tasks rather than requiring to input domain knowledge as required in domain-specific agents.

(10) proposes a conversational agent called "Edina" based on the concept of "self-dilaogs". This system tweaks the Wizard of Oz experiment in such a way that the crowd-workers (participating users) play the role of both end user, as well as expert users that is a single user, helps to inculcate the data for user utterances as well as agent's responses. This helps to gather how users talk about a particular topic in a specific domain. Some of the main advantages of self-dialog policy as highlighted in this research is that the data collection process is fast and also economical as the system doesn't rely on two different users to capture a conversation. And it also helps to bring more naturalness to the conversation. Once the data is recorded in the agent's knowledge base this work then uses a rule-based engine and a matching score component at the run time to generate the appropriate responses. The matching score component given a user query calculates linguistic similarity with the response in the knowledge base using different methods like Inverse Document Frequency (IDF) (29). But this similarity score also takes into consideration context within a dialog so the similarity is measured based on Bag-of-Words model learned from a recorded corpus. The response generated by this component is also validated against the rule-based engine which uses deterministic rules to determine the response for a particular user query.

The Use of a rule-based approach may play as a limitation of this approach. Also, a single user may or may not be able to accurately reflect varied user queries and may also incept some bias in the response to the corresponding request. This technique uses this data to feed into a system knowledge base but doesn't define any pipeline or methodology to iteratively improve the same.

In general, as highlighted in the research (30) which has done a comprehensive analysis of the wide variety of methodologies and other research work being used field of Human-Computer Interaction, most of the state-of-the-art research work utilizes the techniques in HCI like the Wizard of Oz only as methods for usability/experience testing of the system prototypes or as means of data collection methods.

Hence there is a scope where the traditional Wizard of Oz method can be extended which can be used to not only test the conversational agent prototypes and how users interact with it, but also to iteratively gather the training corpus of the conversational agent and also to improve the response quality by enabling engagement of domain experts.

2.5 Methods Based on Conversation Driven Development

Conversation Driven Development (CDD) is based on the principle of training the conversational agent by leveraging the interaction or conversation they have with the real users this enables the agent to iteratively learn and adapt itself. The Rasa X (12, 31) which has been used in the proposed architecture is the tool developed by Rasa open source bot framework team and it supports the philosophy of Conversation driven design (25, 32). Other chatbot frameworks like dialogflow [2] also provide similar tools and interfaces. Although these tools provide easy and intuitive ways to conduct conversations with users and use those for learning and adapting the share the agent in its early development phase to the users and then annotate the conversational data captured to convert it into training data, it doesn't allow the developers to embed domain knowledge into the agent's knowledge base. That is it does not allow developers of these agents to involve expert users to improve the responses which are cardinal processes in any domain-specific conversational agent.

Also as suggested in research (33) training naively on the conversational corpus can lead to poor results because it can reinforce the failure paths selected by the agent also it may lead to learn out-of-domain conversation paths. (33) tries to automate the iterative construction of training data from agents' conversations with the users. work like tries to detect the failure paths in the agent's conversation flow based on the user feedback utterances. So from the user's feedback, it detects how useful or unuseful an agent's response was. For example, user utterances like "that's great" indicates positive feedback on the other hand utterances like "That's not what I meant" indicates negative feedback. This work further tries to build a language classification model to detect failure scenarios using user feedback. But the issue is the method proposed can work on general-purpose conversational data but not on domain-specific data. Also, the work does not propose the mechanism for how to use these failure scenario detection for the iterative improvement of the agent.

To overcome this some of the studies suggest automatic learning for the improvement of the agent (34, 35, 36). (36) uses a type of self-feeding chatbot which automatically learns from the user messages and response pairs in the training data. It uses sentiment analysis and pattern matching model to mine the response by matching a user query to a previously seen query/message in training data. But it also defines pattern matching thresholds between 60% to 90% for deciding when to learn/retraining is required. Pattern matching results lower than each of the thresholds will cause the agent to re-learn the user message response pair and any matching results below 60% will cause the agent to not give the selected response and to deflect the conversation.

While (34) proposes use a reinforcement learning model to predict when to ask questions or user feedback, to learn the accurate response based on the user input/feedback.

As highlighted in the research (37) the main limitation of these approaches is that as these methods depend on machine learning models to predict when learning/retraining is required,

[2] `https://dialogflow.cloud.google.com`

somehow a less accurate or bad model will also be bad at predicting when the agent has failed. Hence regardless of the user feedback these techniques typically also require other factors to gauge the usability of the user feedback to learn from it. Hence in the absence of a reliable model, these agents using these techniques will also start learning inappropriate responses or even the knowledge which is out-of-domain. Hence, all of these techniques based on the concept of automatic learning can adapt the agent through its conversations require complex machine learning models and still don't allow any means of incorporating domain-specific knowledge or answers in the agent's knowledge base.

2.6 Conclusion

Previous sections give a detailed overview of the current state of the research and various techniques and methods in use based on the Wizard of Oz and Conversation Driven Development techniques.

But analyzing all this work it is visible that there is a clear gap in the area of how these methods related to the field of Human-Computer Interaction (HCI) can be used for iterative design and improvement of the conversational agent. Most of the research work uses these methodologies either just to test the prototypes of the conversational agents or just to gather the training corpus. Other works do allow the developers of the agents to involve the expert user in response improvement but these methods are either too complex and/or don't offer any iterative methodology of how it can be leveraged to make conversational agents learn and adapt throughout their life cycle.

The work proposed in this work aims to fill that inherent gap by proposing a system that essentially combines the Wizard Of Oz and Conversation Driven Development techniques. The system proposed essentially defines a methodology that can be used to iteratively increase the scope of the conversational agent (gather training corpus) as well as improve its responses making the conversational agent more engaging and human-like.

Chapter 3

Design & Implementation

This chapter gives the implementation details of all the phases involved in this work. Section 3.1 describes the process adopted to gather and augment the training corpus required to build a baseline bot. Section 3.2 describes the implementation of the baseline Q&A bot which can answer specific queries regarding the project "Beyond 2022" and the underlying technologies used. Lastly, section 3.3 describes the architecture proposed by this work which allows the baseline conversational agent to iteratively learn and improve throughout its lifecycle.

3.1 Data Gathering, Cleaning and Augmentation

3.1.1 Seed Data Gathering

About Project Beyond 2022

As the conversational agent implemented by this work is part of the project Beyond 2022. This section gives a quick overview of this project. Project Beyond 2022[1] is an international collaborative research project funded by the government of Ireland that aims to create the virtual reconstruction of the Public record office of Ireland which was unfortunately destroyed in the fire incident caused during the opening engagement of Irish Civil War dated on 30th June 1922. With 5 core partner universities within Ireland along with over 40 partnering institutions in Ireland, UK, and USA this work tries to recover all the records which were lost in the incident and make them freely accessible to millions of people in the world. Many historians in Ireland and across the globe are also participating in this project to know more about what was lost in this and how it can be rediscovered.

The project has been launched on the centenary of the event in June 2022 and is freely available and accessible to people within and outside Ireland where people can virtually search and view these historic documents and records. Thus as part of this effort, this project also ideates deploying a chatbot on the virtual record treasury website [2] to provide interactive informative assistance to the users while accessing this virtual record treasury.

[1] https://beyond2022.ie/

[2] https://www.virtualtreasury.ie/virtual-treasury

Data Collection through Interactions

As the domain of the conversational agent implemented in this work is related to project Beyond 2022 it was necessary to extract some of the queries users might have about this project or queries users might ask when they visit the virtual archive website created as part of the project Beyond 2022. So that it can be used as a seed corpus to create a baseline bot. The initial training data for a baseline bot was extracted from Q&A sessions conducted with historians involved in the project Beyond 2022.

The recorded interaction data was converted into a training data format as required by the Rasa framework as described in section 3.2.1. Simple python utility script was created using pandas API [3] to parse and convert the conversation data in a way that the questions asked were used as sample intents or user queries and the inputs/answers from the historians were used as a response to those queries/questions. This data was fed to the data cleaning and paraphrasing components before being used to train the baseline bot. The data cleaning and paraphrase(data augmentation) components described in subsequent sections are being reused in phase whenever the new training data and responses were captured using the system defined in section 3.3.

3.1.2 Data Cleaning

This work used simple data cleaning techniques to clean the conversation data including, user queries and responses from domain expert users before it is used to train the conversation agent. It mainly filters the redundant messages. With the use of crowd-sourcing to gather the conversation data, it is always necessary to clean any messages containing any sensitive or rage words. This need has been highlighted by the example of the Microsoft bot Tay (38) about the problems that can arise when conversation data is directly used to train the bot. Hence as part of pre-processing step, this work uses empath client package [4] in python as proposed in (39) to detect and remove the messages having rage and/or sensitive words.

3.1.3 Data Augmentation Using Paraphrasing

Paraphrasing is a popular task of creating alternative variations of a sentence that preserve its meaning with variations in grammatical rules and words (40). Today it's been used heavily used in other tasks such as content creation, language translation, and data augmentation. Many different techniques and algorithms in Natural Language Processing have been in use for paraphrasing (41). These include rule-based algorithms as suggested in (42) or data-driven approach suggested in (43). More recently it's been solved by treating it as a language translation task. While some work (44) also implements the same using bilingual training corpus where we can translate the corpus from one language to other the process known as back-translation.

The efficiency and quality of any conversation agent depend on the efficiency and accuracy of its Natural Language Understanding (NLU) models which is the heart of such systems. NLU models are used to classify and convert the natural language into an action that an agent can perform. While developing any chatbot or conversational agent the main task is to gather

[3] `https://pandas.pydata.org/`

[4] `https://github.com/Ejhfast/empath-client`

and create the training corpus for the intents and actions the agent can handle to improve the accuracy of NLU model. More specifically to be able to make the bot/agent to perform efficiently in real-world conversation it is pertinent to collect different variations in which an end user can ask the same question/query.

Hence the accuracy and performance of the NLU model can be improved by incorporating varied data into the chatbot corpus. But with limited availability of domain-specific data availability, this work uses a Natural Language Paraphraser as a tool for training data augmentation to further make the NLU model accurate and robust. In fact, many latest studies (45) have shown that data augmentation can be used to improve the accuracy of many Natural Language Processing tasks including the text classification used in NLU models.

Paraphrasing models can be customized or built from a scratch using various methods discussed previously. But building it from a scratch was beyond scope of this research and hence this work has used an existing paraphrase model called Parrot (46) . Rasa core framework team has also developed a paraphraser for augmenting text data but is still in the experimental phase. (46) model is built specifically for the data augmentation task in NLU models in conversations agent; As the paraphrasing model has been trained on short sentences of about 32 words because text which is typed or spoken by the end-user on conversational agent interface is usually very small.

(47) suggests different evaluation metrics which are used to evaluate the goodness of the paraphraser. As discussed in (47) there are three main key metrics used to evaluate the performance of a paraphraser. These are "Adequacy" (is the meaning of the original sentence preserved adequately?), "Fluency" (is the generated variation that uses fluent language), and "Diversity" (linguistic differences in the original sentence and its paraphrase). Parrot API[5] provides the additional configurations in order to tune these quality metrics according to the underlying use case.

3.2 Implementing Baseline Conversational Agent

This works implements a baseline conversation agent using Rasa open-source framework (48). Rasa gives the open-source machine learning framework for Natural Language Understanding used in text or voice-based conversations. It also allows to build and connect a bot with different types of channels like Facebook messenger, Slack etc. Later the REST-based channel has been used to connect the Rasa bot server instance to a custom Wizard of Oz middleware implemented in python as a web server. Initially, Rasa command line interface (CLI) was used which gives the set of commands to perform common actinons like training a bot, starting a bot server with a trained bot etc. But further, as this work adopts a Conversation driven approach that uses actual user conversations to adapt and improve the bot/agent (25) . Hence this work has used tool called Rasa X (12, 31) provided by the Rasa community. Rasa X is a browser-based tool layered on top of Rasa open source server which allows to you view, analyze filter the real conversations with the user, and turn these conversations into training data.

[5] https://github.com/PrithivirajDamodaran/Parrot_Paraphraser

Following are the main components in Rasa framework used for building the baseline bot of related to the domain of project "Beyond 2022" -

3.2.1 Training and Response Data Setup

Training data for all the intents (actions) is configured in YAML file format as required by the Rasa open source framework. The YAML file consist of the training data/samples grouped by intents shown in figure 3.1 Rasa has two main components within it which are NLU engine and

```
- intent: faq/archiveARModel
  examples: |
    - Do we have a AR representation of the same as shown in video ?
    - is there an ar model of an archive?
    - show me the model in the archive?
    - where can i find the ar model of the archive?
    - how can i run a real-time ar model of the archive?
    - how can you use an ar model of an archive?
    - where can i find an amazing virtual reality model of archives?
    - where do i get the virtual reality model of archives?
    - how do i get the virtual reality model of the archives?
    - how does a virtual reality model work?
- intent: faq/fireLoss
  examples: |
    - What kinds of records were lost in the fire?
    - which types of documents were lost in the fire?
    - what types of records were lost in the fire?
    - what kinds of documents were lost in the fire?
    - what is lost in a fire?
    - What was lost in the fire ?
    - what was lost by the fire?
    - what was lost ?
    - what has been lost in the fire?
```

Figure 3.1: Training Data in YAML format

Dialog Management engine.NLU engine uses the data presented in format discussed earlier to train the machine learning model in order to understand the and convert natural language into a recognizable intent.

On the other hand, the Dialog Management engine is used to decide what action or response agent should select. Rasa core uses two types of training data format to train its Dialog Management engine namely Rules and Stories to define conversational flow between the end user and agent in form of user intents and corresponding actions the bot should take. Rules are typically used in use cases where conversations happened usually in small pieces and always requires the agent to take the fixed action or response.

This is especially suitable for this work which develops a Q&A conversation agent where the user can ask queries/questions related to the domain of the project "Beyond 2022" and agent replies with a particular response. Although each type of question from a user is represented as a separate intent, the dialog management engine handles it in the same way that is to retrieve the appropriate response based on specific user questions. Hence instead of writing a separate rule/action for each type of possible question, this work uses a feature of retrieval intents in Rasa. It basically allows to group together intents. So all the intents are defined as sub intents of the main intent faq (this can be also seen in figure 3.1) so that we can easily

define a single action or rule to handle them. This feature has been used as it makes writing rules for the dialog management engine quite easy. Now this action in turn makes use of ResponseSelector (49) which is a machine learning model used to select the exact response based on a specific question user has asked.

3.2.2 Natural Language Understanding (NLU) Pipeline

Rasa provides a baseline configuration (50) which are used for training the NLU model. However, main advantage of Rasa is that it allows to easily extend its default configurations in order to create the model as per the requirement of specific use case. The configurations consist of various components (50) which forms the pipeline used by NLU engine for intent recognition , response extraction , text pre-processing. Figure 3.2 shows the final NLU pipeline and different components and configuration this work has used to train the agent.

```
language: en
pipeline:
  - name: WhitespaceTokenizer
  - name: RegexFeaturizer
  - name: LexicalSyntacticFeaturizer
  - name: CountVectorsFeaturizer
  - name: CountVectorsFeaturizer
    analyzer: char_wb
    min_ngram: 1
    max_ngram: 8
  - name: DIETClassifier
    intent_classification: true
    epochs: 150
    constrain_similarities: true
  - name: EntitySynonymMapper
  - name: ResponseSelector
    epochs: 100
    constrain_similarities: true
  - name: FallbackClassifier
    threshold: 0.7
    ambiguity_threshold: 0.1
```

Figure 3.2: NLU Pipeline Configuration.

Following are the main components within the NLU pipeline

1. **Text pre-processing components**
   Following components have been used to perform text pre-processing and feature calculations on user messages. The features calculated are in turn used for the intent classification and response selection models.

   (a) **Tokenizers**
   Whitespace Tokenizer has been used to pre-process the input user message and split it into individual tokens (words). It generates the token for all the words or group of character separated by whitespace. This also cleans the user input message as every character which is not in the regex set (a-zA-Z0-9_#@&) is replaced by the whitespace.

(b) **Featurizers**

Featurizers in Rasa are used to calculate token as well as sentence level features (51). Featurizers mainly generate features in form of dense or sparse feature vectors which are then used by different models like intent classifier and entity recognition. Typically token level features generates the feature vectors of size (No.of tokens * No. of features) are used to train the sequential classifiers. On the other hand the diamensiaonlity of sentence features is (1 * No. of features) and is typically used in Bag-of-word based models/classifiers (52) . This works has used out-of-the-box available Featurizers in Rasa framework which are as follows -

i. **RegexFeaturizer**
This is used to detect the list of regular expressions within the user messages. It takes in the output of the tokenizer. So for each Regex this will count whether the that Regex is present or not finally creating a list or vector of features which can be used by classifier.

ii. **LexicalSyntacticFeaturizer**
Use to analyse and calculate linguistic and syntactical features in a user message. It creates the features using sliding window mechanism over all the tokens in a user message. Some of the default features which are calculated includes prefix and suffix of token, checking whether the token is start of the sentence etc.

iii. **CountVectorsFeaturizer**
It creates Bag-of-word vector/matrix of the user utterances as well as responses. Internally it uses CountVectorizer API in Scikit-learn library (53). It basically counts the frequency of occurrence of each word within the user message, hence the dimensions of vector/matrix depend on number of unique words in entire training corpus. This featurizer has been configured to create bag-of-word feature vector for both word as well as character level n-grams. By default it works on word level n-grams but its been also configured to calculate these features based on character level ngrams using "char_wb" analyser in Rasa as shown in pipeline figure 3.2.

2. **Intent and Response Classifiers Models**

Rasa open source provides multiple out-of-the-box classification models based on simple word embeddings such as LogisticRegressionClassifier or simple KeywordIntentClassifier. Although these classifier fails to handle complexity involved in use case under consideration.

To handle the complexity and variety involved in the user input data , this work uses the DIET (Dual Intent and Entity Transformer) language model suggested recently by study (54). It is a transformer based multi-task model which can be used for both intent and entity recognition task and has a built in support in Rasa open source framework. But main advantage of this architecture is that it can work with a combination of sentence level dense features like word-embeddings and sparse token level or character ngram based sparse features as calculated in in the featurizer pipeline. Also the overall architecture is very lightweight and is faster to train as compared to other simi-

lar models language understanding models like StarSpace(55) which makes use of only sentence level dense features like pretrained word embeddings like GloVe (56). Infact the DIET implementation in Rasa allows to import these pretrained word embeddings into pipeline. And hence this work has used DIETClassifier in Rasa for intent recognition.

As described in section 3.2.1 the dialog management engine within Rasa makes use of ResponseSelection model (49) to select the response for the specific user question or query. ResponseSelection model is also a supervised model classification model with the structure very similar to DIETClassifier which predicts the bot response from configured response set. It can also work with sentence level dense features as well as token based sparse features. It uses learned embeddings of both user messages and response messages and then calculates the similarity between these vectors and try to maximise the similarity between user message and corresponding response configured. At inference time this model will receive the input from pre-processing and intent classifier stages and it will try to predict the response which has the highest similarity with the user intent/message.

Rasa open source also allows to customize and configure the hyperparameter to customise and adapt these classifier models according to a use case. This includes hyperparameters like number of epochs used to train the model , the hidden layer size etc. Figure 3.2 shows the final configuration pipeline finalized using which the best performance was obtained. Table 3.2 shows the performance results obtained on the baseline training data. These results were obtained using command line utility (57) provided by Rasa by runnig 5-Fold cross validation on models.

Table 3.1: Intent and Response Selection Model Evaluation - Training

Model	Accuracy	Precision	F1-score
Intent Classifier	0.721	0.700	0.710
Response Selector	0.773	0.774	0.773

Table 3.2: Intent and Response Selection Model Evaluation - Test

Model	Accuracy	Precision	F1-score
Intent Classifier	0.651	0.651	0.651
Response Selector	0.687	0.680	0.683

3.3 WOzBot Architecture

This section describes the overall architecture of the proposed system which modifies traditional Wizard Of Oz technicque to enable the baseline conversation agent to iteratively learn and adapt.

The Wizard of Oz technique has long been used in the field of Human-Computer Interaction (HCI) for early prototyping and testing of software systems in the nascent stage of development (7). Many studies (19, 58) have used this technique to just mock or simulate some automated system through an expert user or a wizard, particularly in the case of conversational agents, such experiments are primarily being used to capture user messages/interactions that are in turn used as the training corpus (8, 23).

This work modifies the traditional Wizard of Oz technique in such a way that an expert user acts as a mediator between an end user and conversation agents rather than acting as a system itself. This modification enables iterative testing of conversations in actual real-world conversational scenarios, as well as modification of responses based on input from an expert user or a wizard.

Figures 3.3 and 3.4 depict the traditional WOz method and the modified WOz method proposed in this work, respectively. It is clear from figure 3.3 that in traditional settings, the expert user directly simulates a conversation agent.There is no actual involvement of a conversational agent in real conversations.The conversational data is then used directly to train/retrain the conversational agent.

However, as shown in fig 3.4, the suggested method involves the conversational agent in a live conversation with the end user. In live conversations, the expert user acts as a mediator and can change/modify the bot responses. This allows the developers of the agent to test the efficacy of the agent in real-world conversations, collect new training data and responses, and improve the existing bot within a single setting. The recorded responses are validated for correctness against the queries asked by the users. Further, the collected user messages along with the appropriate responses are fed to the data cleaning and paraphrasing pipeline before being used for retraining of the agent.

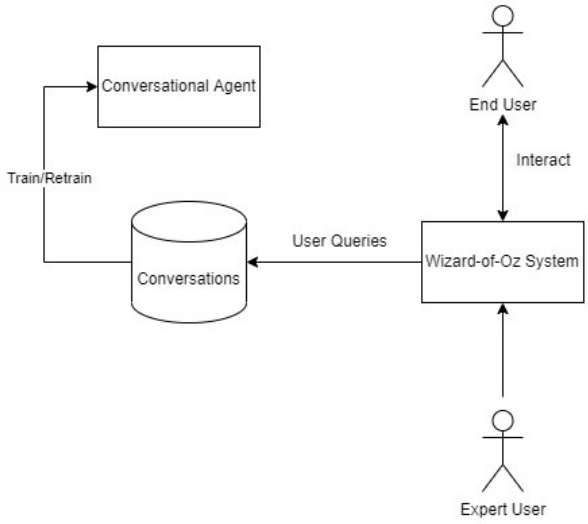

Figure 3.3: Traditional Wizard of Oz Flow

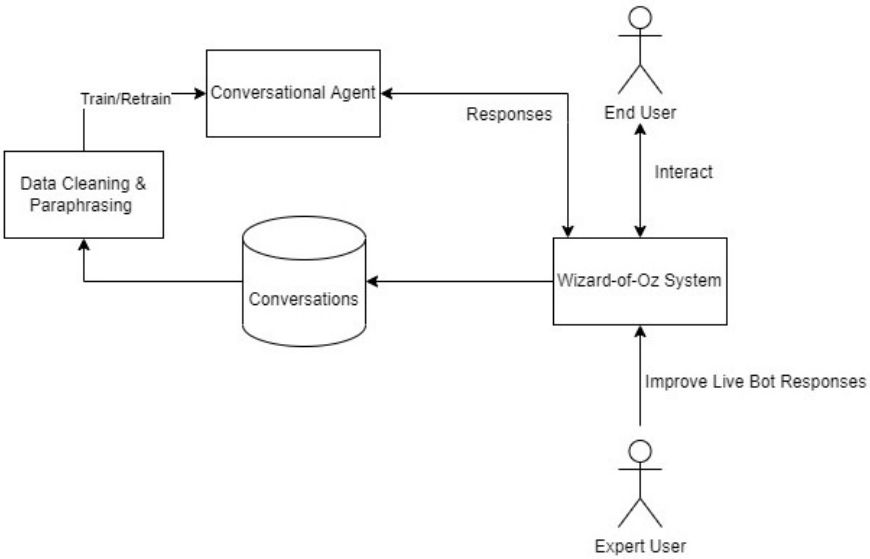

Figure 3.4: Modified Wizard of Oz Flow

Figure 3.5 shows the detailed architecture of proposed system and individual components involved in this architecture.The working details of each of the component within this architecture and underlying technologies used have been explained in detail in remaining section.

1. **WOz Middleware**
   The Wizard-of-Oz middleware is the cardinal part of this system that supports the implementation of the "Wizard-of-Oz" experiment. Essentially this middleware acts as an additional layer between an end user and a conversational agent, further allowing domain experts to act as a mediator.The middleware has been implemented using a python FastAPI [6] web server. There are three endpoints exposed through the FastAPI web server which are as follows -

[6] `https://github.com/tiangolo/fastapi`

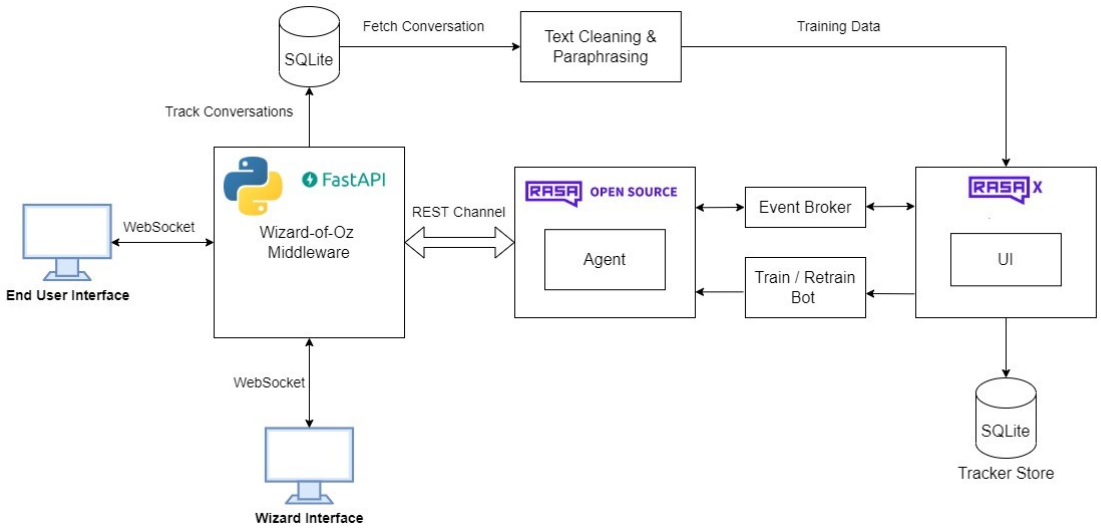

Figure 3.5: WOzBot Architecture

(a) A simple REST endpoint that connects with the REST channel endpoint of the Rasa bot.

(b) A WebSocket endpoint that connects with the end user dashboard. The end user dashboard provides a way for the end users to actually use and converse with the bot.

(c) A WebSocket endpoint that connects with the expert user dashboard. This dashboard provides an intuitive way for the expert user to improve inappropriate chatbot responses based on their domain knowledge if required.

The overall flow captured within the WOz middleware is as follows -

(a) Once the end user gets connected to the end-user dashboard. The user essentially gets connected to a conversational agent through the WOz middleware. From the end-user dashboard, a user can perform actual conversations with a bot. Whenever a user sends any query/question, that request will come to the WebSocket connection at WOz middleware. The middleware simply routes the user request to a Rasa conversational agent over the RESTful channel.

(b) On receiving a response from the Rasa over the Rest channel instead of directly forwarding the response to an end user, the system checks if there is any expert user available at this time. If so, the system will forward the original user message along with the original response of the bot to the expert dashboard over a WebSocket connection.

(c) On the expert user dashboard, the expert user will be able to see the original user message along with the chatbot response. This allows domain experts to use their domain knowledge to directly change/edit inaccurate chatbot responses. Once changed/edited an expert user can approve the response which will be sent back to the WOz middleware over the same WebSocket connection.

(d) In case if the original bot response has been changed/altered (which is identified in Wizard dashboard using simple edit events in Handsontable JavaScript API (59))

the WOz middleware will capture the original message from the user, a response from the bot along with a changed/edited response from the domain expert into the systems knowledge base (SQLite database)[7]. This captured data is in turn used for retraining the bot. This is also recorded in order to calculate number of bot responses altered by the expert user which has been used as intrinsic criteria for evaluating the system.

(e) After the inputs from domain experts, the final response is forwarded to the end user dashboard.

2. **End user Interface**
The end user interface allows users to interact with a conversational agent through a chat interface. This has been implemented using a Bootstrap framework [8]. As shown in figure 3.6 interface initially gives an overview of the scope of the conversation agent to a user. From here user can initiate a conversation with an agent by clicking on "Chat With Assistant" button. Figure 3.7 shows the chat interface. The interface allows the user to exit the conversation at any time. This interface forwards user queries to and receives a response from a Wizard-of-Oz middleware through a WebSocket channel. As shown in figure 3.8 at end of the conversation with a user, this interface also records feedback from the user about the overall chat experience and the quality of response which has been used for the evaluation of effectiveness of the proposed system.

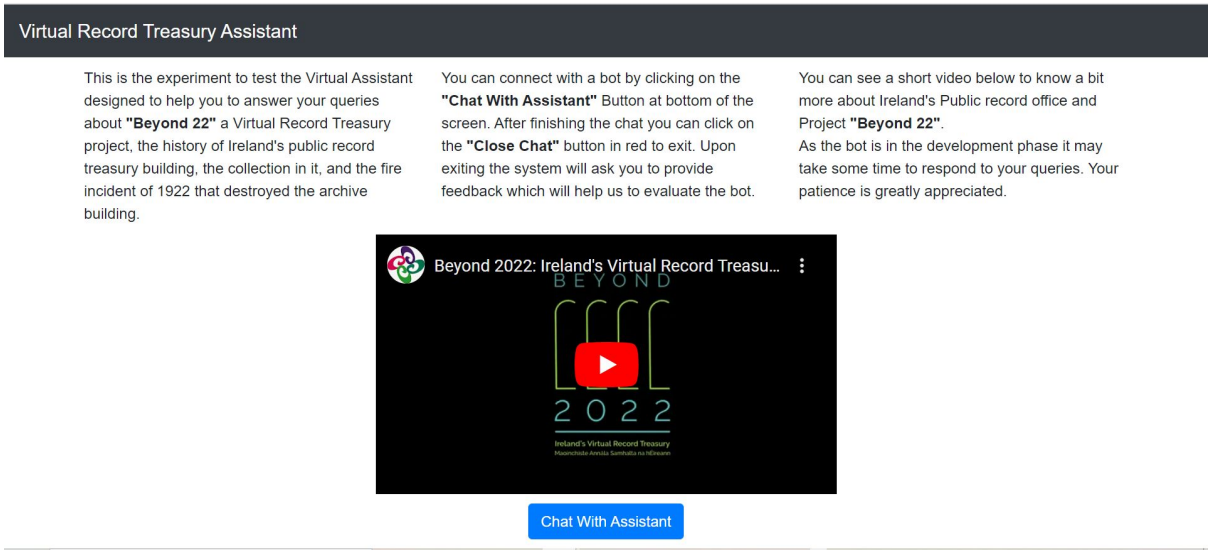

Figure 3.6: User Interface Landing Page

3. **Wizard Interface**
The Wizard/Expert user interface allows domain experts to get connected with a system and to act as a wizard. The Bootstrap framework and JavaScript package known as Handsontable (59) have been used to define an intuitive interface that expert users can use to improve the responses from the bot using their domain knowledge. This interface is connected with Wizard of Oz middleware through a WebSocket channel. Whenever a

[7] https://www.sqlite.org/index.html
[8] https://getbootstrap.com/

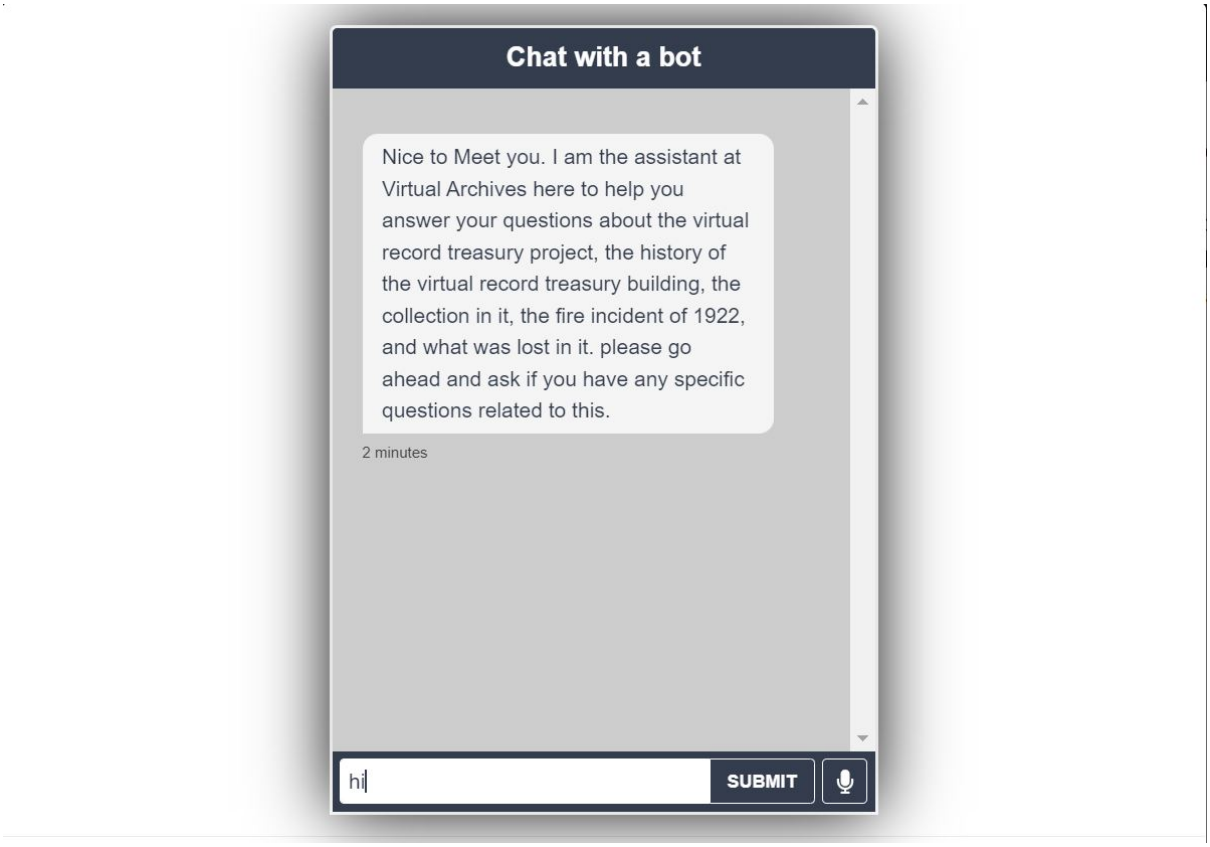

Figure 3.7: Chat Interface

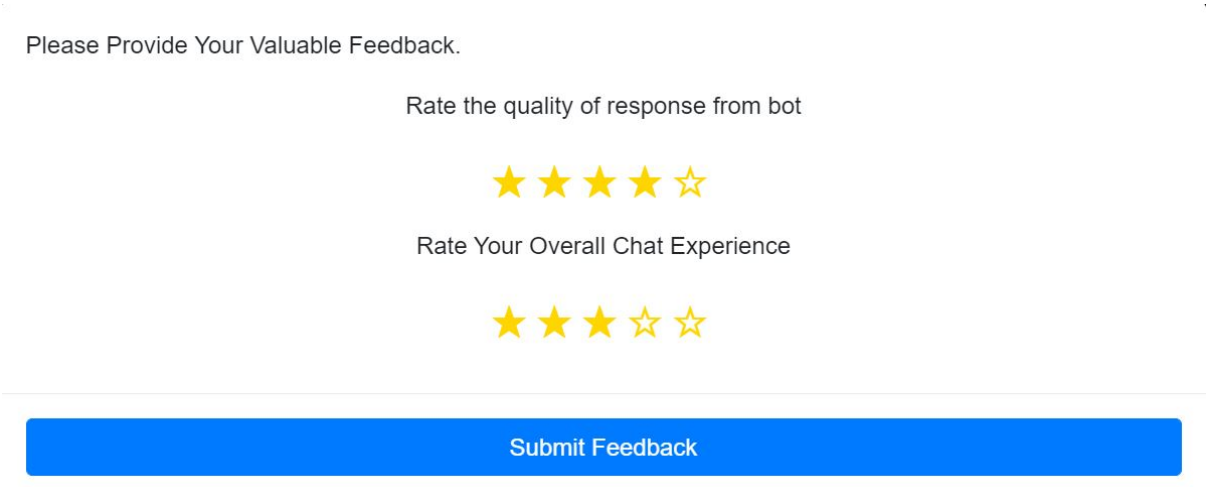

Figure 3.8: User Feedback Page

user message is received, it is first forwarded to Rasa core (48) agent to get the original response. Upon receiving the response from the agent instead of directly transferring it to the end-user, middleware forwards the original message along with the response of the bot to this dashboard over a WebSocket channel. As shown in figure 3.9 this dashboard

populates this data in a simple Handsontable spreadsheet/table. Further figure shows how easily this interface allows the expert users to modify the response from the bot if required. Once modified the expert users can click on the "Approve" checkbox as

shown in so that the approved response is sent back to the end user as shown in figure 3.11 and 3.12 respectively. The UI keeps track of the responses changed/altered by an expert user through a "afterChange" event hook in Handsontable JavaScript API. This is used to keep track of a number of times an expert user has to modify the original bot response which is being used as one of the intrinsic measures for the evaluation of this system.

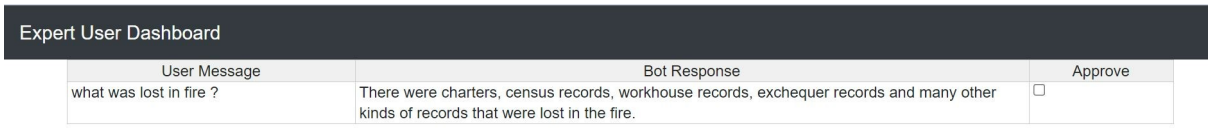

Figure 3.9: Expert Dashboard showing User message along with Bot Response.

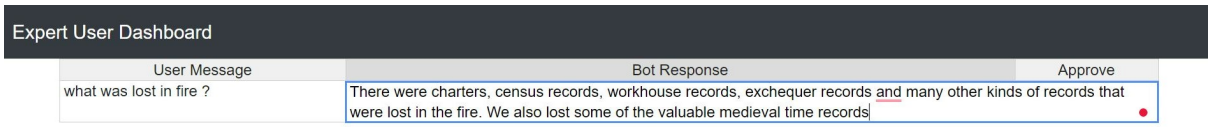

Figure 3.10: Editable Bot Response

Figure 3.11: Expert User Response Approval Functionality

4. **Rasa Open Source**
   Rasa is a popular open-source conversational framework which has been used to develop the conversational agent (48). There are two main components of the Rasa open-source framework which are as follows -

   (a) **Rasa NLU:**
   Rasa Natural Language Understanding (NLU) (60) engine is a crux of the Rasa framework which provides a way to extract a meaning or intent within the messages from the end user so that conversational agent can understand and act upon it. The Rasa NLU internally uses the machine learning libraries like TensorFlow, Spacy and it provides lot of extension hook points in order to configure and customise the NLU pipeline according to the dataset. The configurations of different parameters and hyperparameters used is explained in details in section 3.2.

   (b) **Rasa Dialog Management:**
   Rasa Dialog management is a machine learning based module within Rasa core which is used to select the appropriate action/response based on the user message and dialog context.

   The Rasa open source is connected with a Rasa X (31) instance and it forwards all the conversations (conversation events) to a Rasa X through an event broker (12).

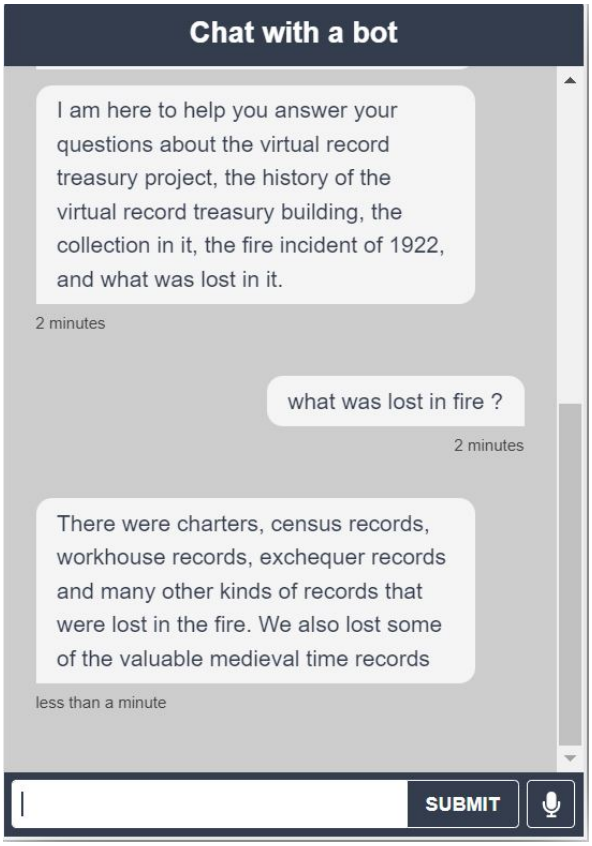

Figure 3.12: User Getting Back Approved Response.

5. **Rasa X**
   Rasa X is essentially a browser-based tool that layers on top of the Rasa open source (12). This tool is best suited for the Conversation Driven Design suggested by the proposed system as it allows to records the entire conversation history of the Rasa agent. Further it allows to sort, filter and analyse these interactions. Rasa X allows an efficient and intuitive way to add and/or correct recorded utterances and responses to the training data that can be used to retrain the agent (31). The conversational data collected from the actual user interactions hence can be use to -

   - Identify which intents the bot identifies successfully and/or unsuccessfully.
   - Improve the assistant response to the intents.

   The communication between Rasa Open Source and Rasa X is captured in figure 3.13. As shown in figure 3.13 Event Broker bridges the communication between Rasa and Rasa X. It uses event driven publisher-subscriber architecture to forward conversation (conversation events) from Rasa Core to Rasa X (12). So, Event broker receives the messages from Rasa core which acts as a producer and then publishes to message queue within Rasa X acting as a subscriber. Rasa X persist these conversation events in SQLite database which can be letter analysed as described earlier through an intuitive UI provided by Rasa X.

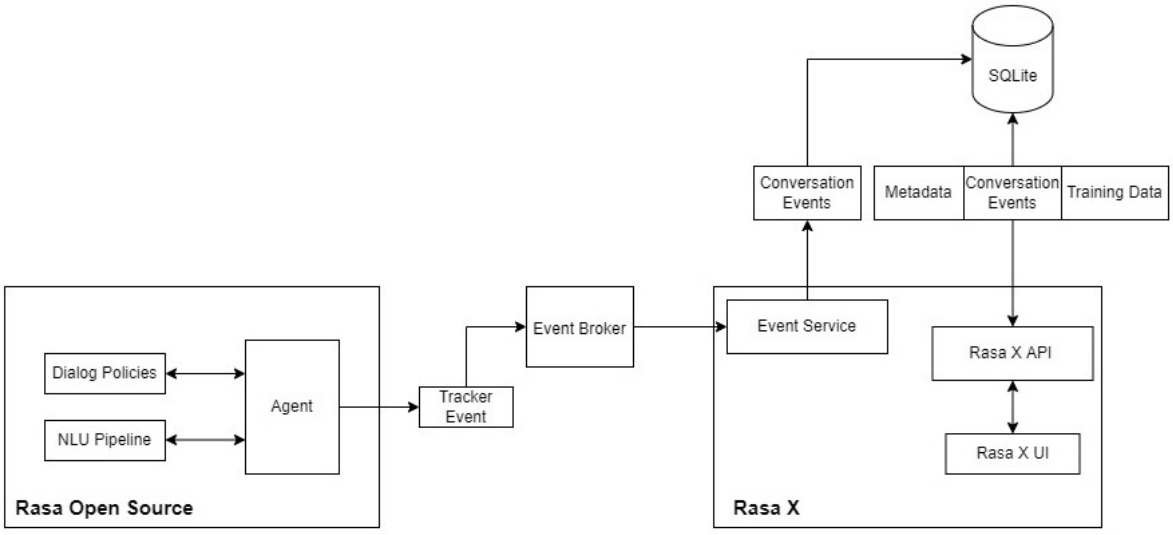

Figure 3.13: Communication Between Rasa Core And Rasa X.

6. **Data Cleaning, Paraphrasing and Agent Retraining:**
   WOz Middleware records the conversation history between the end user and a conversational agent along with the inputs from an expert user. Within each conversation all the responses which are changed/edited by the expert users gets recorded in SQLite database along with the conversation id, original user message and original response from the bot. After each experimental session recorded data is then use to manually retrain the bot through Rasa X. The overall process adopted is as follows -

   (a) The changed/edited response recorded are then cleaned using similar data cleaning methods described in section 3.1.2.

   (b) The original user message is expanded using paraphrasing as described in section 3.1.3. This helps to get a variety of ways users can express a particular intent (query) and helps to augment the training data.

   (c) Rasa X conversation inbox is then utilised in order to manually correct and/or add the data to retrain the bot. As shown in fig the Rasa X provides a conversation inbox to view all the conversations with the end user. Within each conversation we can view the predicted intents and actions(response) along with the confidence level. This enables to analyse the how successfully bot identifies the particular intent and corresponding response.

   (d) Within this conversation inbox Rasa X provides different types of filters to find out the relevant conversation. In this case conversations are filtered based on conversation ID's recorded earlier as shown in figure 3.14. For each recorded user message within the particular conversation following process is followed -

      - Check the predicted intent of the user message. If predicted intent is correct, only the response is changed/edited according to the input provided by the end user.

      - If predicted intent is incorrect, it is checked if the user query matches with any other intent within a training corpus. If it is, the intent is corrected using

the conversational UI as shown in figure 3.15.
In the figure 3.15 it can be observed that within the conversation the user is asking about any casualties in fire incidence but the agent identifies it incorrectly giving a response about the cause of the fire. Rasa x provides an easy interface to correct such misclassification as highlighted in figure 3.15 and store the corrected data in a training corpus.

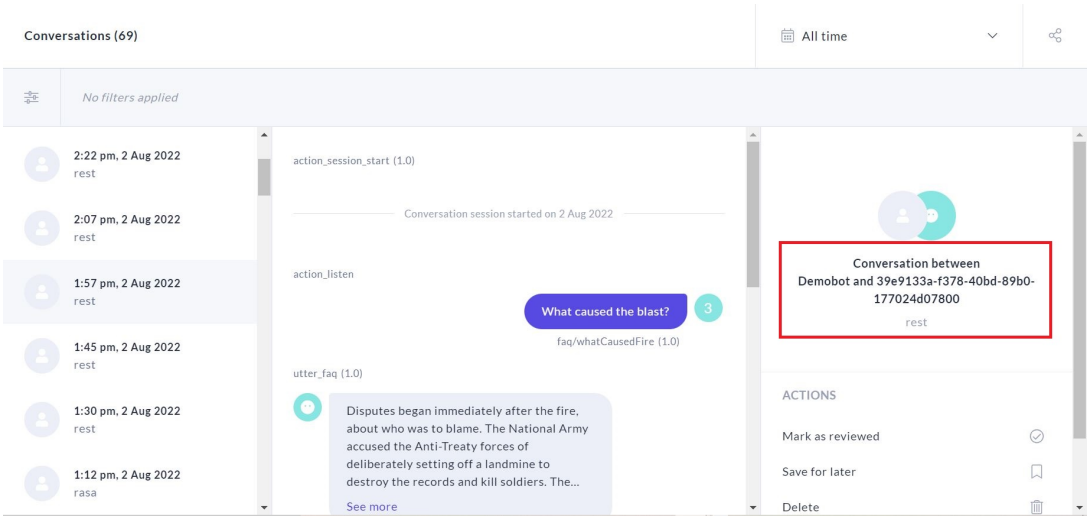

Figure 3.14: Reviewing Conversations Based on Conversation ID.

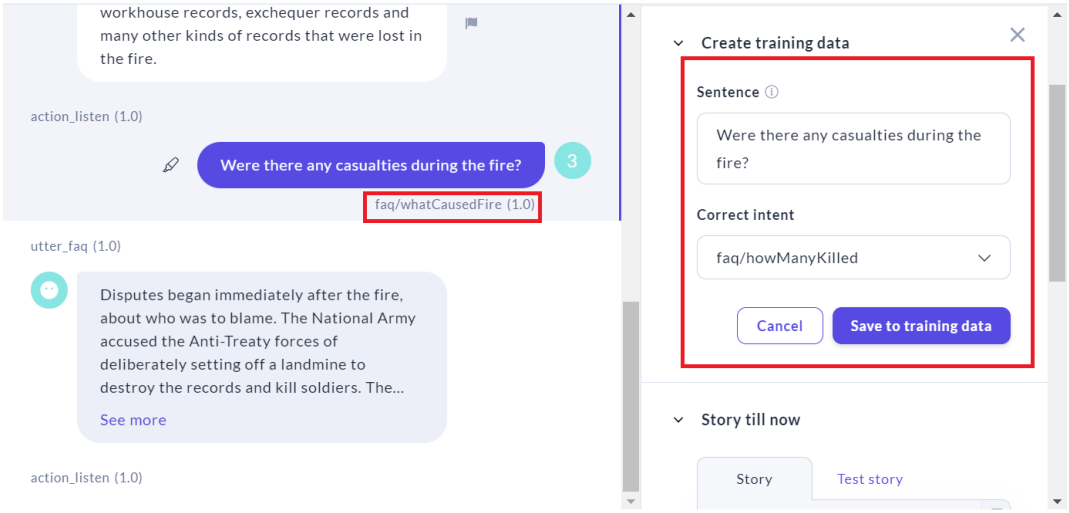

Figure 3.15: Correcting Misclassified Intent In Rasa X.

- If the user query results in a way does not correspond to any of the existing intent, a user message along with the other paraphrased sentences are added as utterances of the new intent.

After following the above process for all the recorded messages in all the conversations within a particular experimental session , agent is retrained. All the results of the training process are recorded and are used for the evaluation of the effectiveness of the the conversational agent at the end of each experimental iteration.

Chapter 4

Evaluation

This chapter discusses the experimental results which are used to evaluate the proposed system based on different factors.

4.1 Wizard Of Oz Experiment

The efficacy of the proposed system has been evaluated by performing the Wizard of Oz experiment on real users. Participants were invited to take part in this experiment, as a part of which they were required to interact with a conversational bot. The participants were asked to connect to an end user interface described in the section through which they can converse with a bot. During the experiment, users were unaware of the involvement of expert users.

Historians involved in project Beyond 2022 were asked to perform the role of the Wizard in this experiment. The expert users would connect to a system through an expert user dashboard allowing them to change/improve the bot response.

The experiment was conducted on a total of 10 participants over 3 sessions. The explicit user feedback from each of the sessions is analyzed separately. Also, the conversation data captured in each session has been used to retrain the conversation agent and measure its performance.

4.2 Evaluating User Feedback

Both explicit, as well as implicit mechanisms were used to capture the feedback of end users as well as expert users involved in the experiment. Some of the evaluation criteria are based on research (61) which standardizes different factors which can be used to evaluate any conversational system.

1. **Explcit Feedback**
   As at the end of the conversation with an agent, end users were asked to submit small feedback in terms of the number of stars on a scale of 1-5 regarding the following factors-

- Overall chat experience -
  The end users were asked to give their feedback about how good their overall experience was while conversing with a bot. This measure helped analyze the overall performance of the bot in terms of user engagement.
- Quality of response -
  The end users were asked to give their feedback about the quality of responses from the agent. This measure helped to analyze the quality of the response from the bot and iterative improvement in the response to satisfy the information need of the users.

  Similarly, historians or expert user was asked to give feedback in terms of rating on a scale of 1-5 about the quality of response from the agent.

2. **Implicit Feedback**
   As described in section 3, all edit events on the expert user dashboard were captured to keep internal track of the number of times an expert user had to change/modify the original bot response allowing to measure and analyze the overall response improvement of the bot.

Table 4.1 shows the result of feedback received from the end users after each experimental session conducted. Similarly table 4.2 shows the explicit feedback from the expert users as well as implicit feedback captured by the system.

Table 4.1: End User Feedback on scale of 1-5 Stars

	Average Rating for Chat Experience	**Average Rating for Response Quality**
Session 1	2.8	4
Session 2	3.5	4
Session 3	4	5

Table 4.2: Expert User Feedback on scale of 1-5 Stars Along with Implicit Feedback

	Average Rating for Response Quality	**Percentage Of Edited Response**
Session 1	1	95%
Session 2	3	65%
Session 3	4	48%

The clear co-relation can be observed in the collected feedback from the end users as well as expert users. It can be observed that in the first session of the experiment implicit feedback indicates that the expert user had to change/modify almost about 95% of the responses from the bot. This can be also confirmed from the accuracy measures of the conversation agent presented which indicates that the agent was not able to handle most of the user queries. As a result of this, the overall time the user had to wait for the bot response resulted in a lower rating for the overall chat experience from the user.

Although the average rating for the response quality from the end user is good as the response was anyway modified by an expert user based on the domain knowledge.

But it can be seen that as the conversation data captured from the earlier session was used to retrain the bot, the percentage of bot responses expert users had to modify gradually decreases further improving the overall chat experience of the end users and keeping them more engaged. This can be observed from the increased rating from the user for the overall experience in subsequent experimental sessions.

4.3 Evaluating Performance of Conversational Agent

One of the main objectives of the proposed system is to iteratively improve the conversational agent to make it more engaging and human-like. Therefore it is essential to evaluate the performance improvement of the conversational agent throughout each stage of the conducted experiment. There are different parameters being considered to gauge the performance improvements of a conversational agent the main parameters considered are as follows -

1. **Natural Langauge Understanding(NLU) model Metrics**
   The conversational data captured in each of the experimental sessions were used for retraining the bot. The performance of the intent classifier and Response selection model is being evaluated in terms of different metrics like accuracy, precision, and F1-score of the model. The results presented reflects how the performance of the conversational agent in terms of user query handling and response generation iteratively evolved through each experimental session highlighting the effectiveness of the proposed method. All these metrics were measured using the Rasa Command Line interface tools (57) which enable testing models and generate performance metrics reports. All of these metrics were obtained by testing these models using a 5-fold cross-validation method.
2. **Conversational Agent Corpus Expansion Metrics**
   Various metrics are being used to analyze how the proposed method helps to iteratively capture the training corpus of the bot and in a way increase the scope of the queries a bot can handle. These include metrics like the number of intents after each experimental phase and the average number of utterances per intent.

Table 4.3 presents the results evaluation results for NLU model performance. Table 4.4 presents results evaluating corpus expansion of the agent.

It can be observed from the evaluation results that experimental sessions allowed to iteratively capture varied user queries/intents through actual user interaction this also resulted in improvement of NLP models within the conversational agent. In table 4.4, it can be observed that the number of unique intents within a conversational agent increased over time resulting in an overall broadening of the overall scope of an agent.

Also results in table 4.3 shows that with the increase in quality and quantity of training corpus (including the user utterance and ideal bot response) at end of each experimental session, the performance metrics of both intent classifier and response selection model within the Rasa improved significantly.

Table 4.3: Intent and Response Selection Model metrics at the end of each experimental phase

Experiment Phase	Model	Accuracy	Precision	F1-score
Baseline Bot	Intent Classifier	0.7210	0.7001	0.7101
	Response Selector	0.7733	0.7748	0.7735
Session 1	Intent Classifier	0.8335	0.8495	0.8490
	Response Selector	0.8567	0.8572	0.8534
Session 2	Intent Classifier	0.9006	0.9149	0.9140
	Response Selector	0.9279	0.9220	0.9216
Session 3	Intent Classifier	0.9161	0.9288	0.9160
	Response Selector	0.9596	0.9620	0.9590

Table 4.4: Number of Intents at the end of each experimental phase

Experimental Phase	Number Of Intents	Total Number Of Utterances	Average Utterances Per Intent
Baseline Bot	20	300	15.00
Session 1	32	796	24.87
Session 2	40	932	23.30
Session 3	44	1040	23.63

Thus all the evaluation results presented in this chapter reflect the effectiveness of the proposed system to iteratively improve the scope and performance of the conversational agent making it adapt and learn from actual user conversations throughout its development lifecycle.

Chapter 5

Conclusion

This chapter concludes this report. It gives final remarks on the proposed system in the work and assesses it against the research objectives stipulated. It also discusses the limitations of the proposed work and possibilities of extension and future work.

5.1 Research Objective Assessment

Following is the overall review of this work in terms of research objectives as described in section 1.3.

1. **Developing a Baseline Bot**

   Research Objective - *To design and implement a baseline Q&A agent/bot using an open source chat framework.*

   This work has successfully implemented a baseline Q&A chatbot for answering the user queries related to a specific aspect of the project Beyond 2022. The work uses Rasa open source framework to build and implement the baseline bot. This work also defines efficient data cleaning and data augmentation methods(Section 3.1.2 and 3.1.3) which are used to improve both the quality and quantity of the seed training data. Further, it also defines how these components can be reused across the development cycle of the agent. This work also defines and reasons for different Natural language processing algorithms and models (Section 3.2.2) that can be leveraged to build the Q&A type bot with high performance. This can be easily extended across multiple domains.

2. **Implementing a system for Conducting Wizard of Oz Experiment**

   Research Objective - *To design a platform for conducting a Wizard of Oz based experiment to simultaneously gather the new training data from the end user and improve the responses of the bot based on inputs of an expert user (wizard).*

   As described in section 3.5 system proposed by this work modifies the traditional Wizard of Oz technique. The proposed system is designed in such a way that allows historians(Wizard) to act as a mediator between the end users and a conversational agent,

allowing them to change and improve the actual responses from the agent based on the domain knowledge. It also allows the system at the same time to incrementally capture the additional training corpus which can be used along with the modified responses to retrain the agent to make it more efficient. The design also allows using WOz experiment to test the performance of the bot with real-time conversations with end users. Both the user interface (2) as well as expert user interfaces (3) designed within this work are very simple and intuitive allowing users to easily participate in such experiments without demanding any technical skills. The overall design of the system is highly extensible so that it can be re-used to perform the Wizard Of Oz experiment to test and evaluate the conversational agents in any domain.

3. **Methodology to store and use conversations for iterative improvement**

   Research Objective - *To design a methodology to track the end-user conversations and use them along with expert user inputs for iterative training of the agent.*

   The work uses Conversation Driven Developement startegy (25) in order to leverage the re al time conversation to mine the training data from it. The work makes use of tools like Rasa X (12, 31) which are internally designed to support such development philosophy. In addition it uses simple mechanisms on the proposed wizard interface to track the conversations with the user along with the inputs from the expert users which helps to improve the responses from the agent. All the user message along with the inputs from the expert users are stored in efficient and lightweight tracker store from where they can be easily exported for bot retraining. This work also defines a standard procedure as described in section 6 to use these stored conversations in order to retrain the agent.

4. **Proposed System Evaluation**

   Research Objective - *To evaluate the proposed system through the explicit feedback provided by the end users based on their interaction with the agent, as well as implicit feedback measures.*

   This work uses simple explicit as well as an implicit feedback mechanism to gauge the performance and effectiveness of the system. It uses explicit ratings provided by end users as well as an expert user to analyze how the responses from the bot improve over iteratively. It also uses some of the implicit measures like NLU model performance metrics, the size of the training data corpus etc. to evaluate how the proposed system helps the developer of the agents to improve the accuracy and performance of the conversational agent using actual user interactions. All these evaluation measures have been applied in the actual experimental settings defined in the chapter 4 to obtain the results further encouraging the adoption of the proposed system.

5.2 Remarks

- This works proposes a methodology based on the concept of Conversation-Driven Development (CDD) to iteratively design the conversational agent and to make it learn and adapt throughout its lifecycle based on actual user interactions.

- The proposed system design makes use of the Wizard of Oz technique but it modifies the traditional technique in such a way that allows the developers of the agent to gather the training corpus and to improve the bot responses within the same setting during the interactions.

- Along with improving agent response the proposed system also allows developer of agents to iteratively broaden the scope of the it allowing it handle varied types of queries.

- The design of the proposed system is very simple and intuitive making it very extensible to use for the development of the conversational agent across numerous domains.

- The WOz middleware architecture proposed makes this system easy and economical to adopt. Also, it works on generic REST and Webhook protocols so that it can be easily integrated with any chatbot or NLP framework.

- This technique will allow the developers of conversational agents to test the agent in the early development phase through actual interaction which can give valuable insight in terms of the efficiency of the agent at a very early stage. Also, it allows involving the domain expert at an early stage to improve its responses right from the nascent development stage.

5.3 Future Scope

5.3.1 Limitations

- The proposed approach is evaluated on a total of 10 users hence although the evaluation results are promising the efficiency of the system can be evaluated by performing the experiment on a large number of users and may be on the conversation agents in disparate domains.

- This system fully relies on an expert user to instantaneously provide appropriate and reliable answers to a user query. But the even expert users would not know the exact response to give to a user query at a particular time. Also, this work does not provide any means to suggest or to help the expert users to build responses.

- The experiment conducted as part of this research only involved one expert, and a single user may not be able to handle multiple users' queries efficiently this can be visible in the experimental results that the delay caused by this indirectly lowers the overall user

experience ratings/feedback. Hence the measure of the effectiveness of this method on overall user experience may not reliable which further can be tested by the extending system to multiple expert users.

- The proposed system does not address and specifies the way the system can be extended to involve multiple expert users. Also if multiple expert users are involved how coordination in response generation between those users can be handled.

5.3.2 Future Avenues

Implementing Response Review Dashboard

The current work allows to automatically capture all the user messages and input from the expert users (wizards) in form of modification in bot response. However, responses from the expert users may not be accurate or may contain some noisy text/data. In some cases, even expert users may not be able to give the appropriate answers to a query. To handle such scenarios currently, the developers adopting this system will have to manually review each of the responses in the database and filter out them.

Instead to help developers easily analyze and filter out response improvements a simple dashboard can be designed to provide developers of the agent with an easy interface to filter out responses that cannot be used in the retraining of the agent. This dashboard can provide a simple user interface that would allow developers to review the responses collected from the conversation data and in case require modify or delete it. Some simple tools like (62) can be used to automatically delete the responses that contain sensitive words or have a high negative sentiment score. This will be especially useful to avoid the issues in the training agents directly on conversation history as visible in the case of some famous conversational agents like Microsoft's Tay (38).

Extending system To A Multi-Wizard Setup

The proposed system design only allows a single expert user to connect with the system. So in case when multiple users get connected to a system simultaneously, they would experience extra dealy from a system as a single expert user may not be able to handle multiple concurrent user queries. This puts a limit on the number of concurrent end users who can connect to a system. Instead, this system can be extended to support the involvement of multiple expert users which will enable to concurrently handle a large number of end-user requests. This will further enable to conduct this experiment on a larger scale allowing to capture a huge number of conversations.

Contextual Query Handling

The chatbot response improvement technique suggested here is best suitable for the free-flowing Q&A type agent. However, in some use cases, the agent is required to generate user-specific responses based on the context within a dialog. Hence the modification will be required to adopt the proposed system for the improvement of contextual bots.

Adopting Expert Systems

In a typical scenario, a wizard or expert user may or may not know the exact answer to a user's queries according at a particular time. In such cases, this architecture can be extended by including some sort of expert database system or domain-specific Knowledge Graphs which can be used to suggest possible responses to an expert user.

Extending System Over Multiple Channels

The design of the proposed system is best suited for traditional text-based agents/bots. However, with the advancement in the field of AI and natural language understanding, voice-based bots/agents are also getting prevalent. Hence current system design can be extended to support the iterative design of the conversational agents over different channels including voice-based agents/bots.

Bibliography

[1] Daniel Jurafsky. *Dialog Systems and Chatbots*, volume 35. Pearson Prentice hall, 2017.

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

[3] Kenneth Mark Colby, Sylvia Weber, and Franklin Dennis Hilf. Artificial paranoia. *Artificial Intelligence*, 2(1):1–25, 1971.

[4] Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. A survey of available corpora for building data-driven dialogue systems. *arXiv preprint arXiv:1512.05742*, 2015.

[5] Joel Ross, Andrew Zaldivar, Lilly Irani, and Bill Tomlinson. Who are the turkers? worker demographics in amazon mechanical turk. *Department of Informatics, University of California, Irvine, USA, Tech. Rep*, 49, 2009.

[6] Scott R Klemmer, Anoop K Sinha, Jack Chen, James A Landay, Nadeem Aboobaker, and Annie Wang. Suede: a wizard of oz prototyping tool for speech user interfaces. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 1–10, 2000.

[7] Bruce Hanington and Bella Martin. *Universal methods of design expanded and revised: 125 Ways to research complex problems, develop innovative ideas, and design effective solutions*. Rockport publishers, 2019.

[8] John F Kelley. An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems (TOIS)*, 2(1):26–41, 1984.

[9] Ting-Hao Huang, Joseph Chee Chang, and Jeffrey P Bigham. Evorus: A crowd-powered conversational assistant built to automate itself over time. In *Proceedings of the 2018 CHI conference on human factors in computing systems*, pages 1–13, 2018.

[10] Ben Krause, Marco Damonte, Mihai Dobre, Daniel Duma, Joachim Fainberg, Federico Fancellu, Emmanuel Kahembwe, Jianpeng Cheng, and Bonnie Webber. Edina: Building an open domain socialbot with self-dialogues. *arXiv preprint arXiv:1709.09816*, 2017.

[11] Walter S Lasecki, Rachel Wesley, Jeffrey Nichols, Anand Kulkarni, James F Allen, and Jeffrey P Bigham. Chorus: a crowd-powered conversational assistant. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 151–162, 2013.

[12] How to migrate your assistant to rasa x (the easy way). https://rasa.com/blog/how-to-migrate-your-assistant-to-rasa-x-the-easy-way/, Jun 2022.

[13] Rainer Winkler and Matthias Söllner. Unleashing the potential of chatbots in education: A state-of-the-art analysis. In *Academy of Management Annual Meeting (AOM)*, 2018.

[14] Stephan Diederich, Max Janssen-Müller, Alfred Benedikt Brendel, and Stefan Morana. Emulating empathetic behavior in online service encounters with sentiment-adaptive responses: insights from an experiment with a conversational agent. 2019.

[15] Jewish heritage network, museum chatbot. https://jhn.ngo/services/chatbot.

[16] Teaching a titanosaur to talk: Conversational ux design for field museum. https://purplerockscissors.com/blog/teaching-a-titanosaur-to-talk.

[17] Jonathan Grudin and Richard Jacques. Chatbots, humbots, and the quest for artificial general intelligence. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–11, 2019.

[18] Eun Go and S Shyam Sundar. Humanizing chatbots: The effects of visual, identity and conversational cues on humanness perceptions. *Computers in Human Behavior*, 97: 304–316, 2019.

[19] Nils Dahlbäck, Arne Jönsson, and Lars Ahrenberg. Wizard of oz studies—why and how. *Knowledge-based systems*, 6(4):258–266, 1993.

[20] John D Gould and Clayton Lewis. Designing for usability: key principles and what designers think. *Communications of the ACM*, 28(3):300–311, 1985.

[21] L Frank Baum. *The wonderful wizard of Oz*. Oxford University Press, 2008.

[22] S. Dow, B. MacIntyre, J. Lee, C. Oezbek, J.D. Bolter, and M. Gandy. Wizard of oz support throughout an iterative design process. *IEEE Pervasive Computing*, 4(4):18–26, 2005. doi: 10.1109/MPRV.2005.93.

[23] Sang-su Lee, Jaemyung Lee, and Kun-pyo Lee. Designing intelligent assistant through user participations. In *Proceedings of the 2017 conference on designing interactive systems*, pages 173–177, 2017.

[24] Indrani Medhi Thies, Nandita Menon, Sneha Magapu, Manisha Subramony, and Jacki O'neill. How do you want your chatbot? an exploratory wizard-of-oz study with young, urban indians. In *IFIP Conference on Human-Computer Interaction*, pages 441–459. Springer, 2017.

[25] Conversation-driven development. https://rasa.com/docs/rasa/conversation-driven-development/, Aug 2022.

[26] Giovanni Almeida Santos, Guilherme Guy de Andrade, Geovana Ramos Sousa Silva, Francisco Carlos Molina Duarte, João Paulo Javidi Da Costa, and Rafael Timóteo de Sousa. A conversation-driven approach for chatbot management. *IEEE Access*, 10:8474–8486, 2022. doi: 10.1109/ACCESS.2022.3143323.

[27] Lue Lin, Luis Fernando D'Haro, and Rafael Banchs. A web-based platform for collection of human-chatbot interactions. In *Proceedings of the Fourth International Conference on Human Agent Interaction*, pages 363–366.

[28] Mohit Jain, Pratyush Kumar, Ramachandra Kota, and Shwetak N Patel. Evaluating and informing the design of chatbots. In *Proceedings of the 2018 designing interactive systems conference*, pages 895–906, 2018.

[29] Septya Egho Pratama, Wahyudin Darmalaksana, D Sa'adillah Maylawati, Hamdan Sugilar, Teddy Mantoro, and Muhammad Ali Ramdhani. Weighted inverse document frequency and vector space model for hadith search engine. *Indones. J. Electr. Eng. Comput. Sci*, 18(2):1004–1014, 2020.

[30] Leigh Clark, Philip Doyle, Diego Garaialde, Emer Gilmartin, Stephan Schlögl, Jens Edlund, Matthew Aylett, João Cabral, Cosmin Munteanu, Justin Edwards, et al. The state of speech in hci: Trends, themes and challenges. *Interacting with Computers*, 31(4): 349–371, 2019.

[31] Want to improve your ai assistant? then you need rasa x. https://info.rasa.com/improve-ai-assistant-rasa-x.

[32] Conversation-driven development. https://rasa.com/blog/conversation-driven-development-a-better-approach-to-building-ai-assistants/, Oct 2021.

[33] Chikara Hashimoto and Manabu Sassano. Detecting absurd conversations from intelligent assistant logs by exploiting user feedback utterances. In *Proceedings of the 2018 World Wide Web Conference*, pages 147–156, 2018.

[34] Jiwei Li, Alexander H Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. Learning through dialogue interactions by asking questions. *arXiv preprint arXiv:1612.04936*, 2016.

[35] Haichao Zhang, Haonan Yu, and Wei Xu. Interactive language acquisition with one-shot visual concept learning through a conversational game. *arXiv preprint arXiv:1805.00462*, 2018.

[36] Jordan J Bird, Anikó Ekárt, and Diego R Faria. Learning from interaction: An intelligent networked-based human-bot and bot-bot chatbot system. In *UK Workshop on Computational Intelligence*, pages 179–190. Springer, 2018.

[37] Braden Hancock, Antoine Bordes, Pierre-Emmanuel Mazare, and Jason Weston. Learning from dialogue after deployment: Feed yourself, chatbot! *arXiv preprint arXiv:1901.05415*, 2019.

[38] Gina Neff. Talking to bots: Symbiotic agency and the case of tay. *International Journal of Communication*, 2016.

[39] Ethan Fast, Binbin Chen, and Michael S Bernstein. Empath: Understanding topic signals in large-scale text. In *Proceedings of the 2016 CHI conference on human factors in computing systems*, pages 4647–4657, 2016.

[40] Regina Barzilay and Kathleen McKeown. Extracting paraphrases from a parallel corpus. In *Proceedings of the 39th annual meeting of the Association for Computational Linguistics*, pages 50–57, 2001.

[41] Sam Witteveen and Martin Andrews. Paraphrasing with large language models. *arXiv preprint arXiv:1911.09661*, 2019.

[42] Kathleen R McKeown. Paraphrasing using given and new information in a question-answer system. *Technical Reports (CIS)*, page 723, 1980.

[43] Nitin Madnani and Bonnie J Dorr. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387, 2010.

[44] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*, 2015.

[45] Eda Okur, Saurav Sahay, and Lama Nachman. Data augmentation with paraphrase generation and entity extraction for multimodal dialogue system. *arXiv preprint arXiv:2205.04006*, 2022.

[46] Prithiviraj Damodaran. Parrot: Paraphrase generation for nlu., 2021.

[47] Chang Liu, Daniel Dahlmeier, and Hwee Tou Ng. Pem: A paraphrase evaluation metric exploiting parallel texts. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 923–932, 2010.

[48] Rasa open source: Open source conversational ai platform. https://rasa.com/open-source/, Jul 2022.

[49] Response selector in rasa. https://rasa.com/docs/rasa/componentsresponseselector, Aug 2022.

[50] Components. https://rasa.com/docs/rasa/components, Aug 2022.

[51] Rasa featurizers. https://rasa.com/docs/rasa/componentsfeaturizers, Aug 2022.

[52] Bagus Setya Rintyarna, Riyanarto Sarno, and Chastine Fatichah. Evaluating the performance of sentence level features and domain sensitive features of product reviews on supervised sentiment analysis tasks. *Journal of Big Data*, 6(1):1–19, 2019.

[53] Sklearn api. https://scikit-learn.org/stable/modules/classes.html.

[54] Tanja Bunk, Daksh Varshneya, Vladimir Vlasov, and Alan Nichol. Diet: Lightweight language understanding for dialogue systems. *arXiv preprint arXiv:2004.09936*, 2020.

[55] Ledell Wu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes, and Jason Weston. Starspace: Embed all the things! In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[56] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[57] Rasa test. https://rasa.com/docs/rasa/command-line-interfacerasa-test, Aug 2022.

[58] S. Dow, B. MacIntyre, J. Lee, C. Oezbek, J.D. Bolter, and M. Gandy. Wizard of oz support throughout an iterative design process. *IEEE Pervasive Computing*, 4(4):18–26, 2005. doi: 10.1109/MPRV.2005.93.

[59] Handsontable - rock-solid data grid for web applications. https://handsontable.com/.

[60] Open source natural language processing (nlp). https://rasa.com/solutions/open-source-nlu-nlp/, Jul 2021.

[61] Wari Maroengsit, Thanarath Piyakulpinyo, Korawat Phonyiam, Suporn Pongnumkul, Pimwadee Chaovalit, and Thanaruk Theeramunkong. A survey on evaluation methods for chatbots, 2019. URL https://doi.org/10.1145/3323771.3323824.

[62] Clayton Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the international AAAI conference on web and social media*, volume 8, pages 216–225, 2014.