Health Agent: Conversational AI for Primary Healthcare Support Advice

Vishesh Bhagat
210010283

Supervisors: Dr Alice Toniolo, Dr Phong Le

Master of Science (MSc) in Artificial Intelligence
School of Computer Science
University of St Andrews
Academic Year 2021-22
August 2022

Abstract

Health Agent: Conversational AI for Primary Healthcare Support Advice

For common illnesses, netizens often regularly search for information on the internet, read through sophisticated medical blogs and comprehend the suggested treatment based on their capabilities. The information provided on the internet may not be easily searched, comprehended and consumed. Medical terminology's intricacy may make this information search more difficult and uncomfortable.

This report summarizes the design and implementation of a Health Agent, a proof of concept of an artificial intelligence-based conversational agent, which is trained with information about common illnesses and corresponding symptoms. Users can naturally communicate with the agent like they would do with a healthcare representative during their initial screening at a medical establishment. In a multi-turn conversation, the Health Agent seeks symptoms the user experiences and responds with one or more possible illnesses that match these symptoms. With this ready-to-consume knowledge in simplified form, users are saved from the hassle to go through complicated online documentation.

The project makes use of state-of-the-art NLP techniques like transformers and transfer learnings for handling NLP tasks involved in dialogue management and information extraction. An open-source conversation AI platform has been used for orchestrating conversational flows and tasks. The functionality of the agent is exposed using a web-based interface. A survey about the naturalness and functionality of the agent demonstrates the respondent's interest in using such agents.

Keywords

Conversational AI, NLP, Transformer, Transfer Learning, Healthcare

Acknowledgements

I would like to thank my dissertation supervisors, Dr Alice Toniolo and Dr Phong Le, for their unwavering enthusiasm regarding this project and the invaluable guidance and support given to me throughout the project. I would also like to thank them for enabling me to choose a topic of my interest and letting me explore the field of conversational AI. I am extending my thankfulness to the student services and project coordinator for being supportive during my difficult time throughout the project work.

I would also like to thank my university friends who were my study companions during my research work. The esprit de corps made this journey interesting and memorable.

I would like to express my gratitude to my family for their unconditional support and encouragement throughout my education and especially during my master's degree at the University of St Andrews, where we were not able to meet for extended periods.

Declaration of Authorship

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated.

The main text of this project report is 13,962 words long, including project specification and plan.

In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bona fide library or research worker and to be made available on the World Wide Web. I retain the copyright in this work

Date: 27th August 2022

Vishesh Bhagat

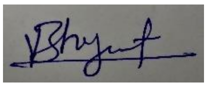


Table of Contents

List of Figures

List of Tables

Abbreviations

AI	Artificial Intelligence
ASR	Automated Speech Recognition
BERT	Bidirectional Encoder Representations from Transformers
CNN	Convolutional Neural Network
DIET	Dual Intent and Entity Transformer
DL	Deep Learning
DM	Dialogue Management
FAQ	Frequently Asked Questions
GUI	Graphical User Interface
GPT	Generative Pre-trained Transformer
HTML	HyperText Markup Language
HMM	Hidden Markov Model
JSON	JavaScript Object Notation
LSTM	Long Short-Term Memory
ML	Machine Learning
NER	Named Entity Recognition
NLG	Natural Language Generation
NLP	Natural Language Processing
NLU	Natural Language Understanding
NN	Neural Network
POS	Part-of-speech
RNN	Recurrent Neural Network
TED	Transformer Embedding Dialogue
TTS	Text to speech
Q&A	Question and Answer
YAML	YAML Ain't Markup Language or Yet Another Markup Language

1 Introduction

Conversational Artificial Intelligence is a system that primarily makes use of techniques from the natural language processing field of artificial intelligence and machine learning techniques to generate human-like interactions with the system. When combined with state-of-the-art interfaces, the interaction with these systems becomes like interaction with human beings

Kong (2021) argues that voice-based and conversation-based interactions are poised to replace the traditional web-based or command-line interfaces in many applications. Interactions with digital systems are becoming more natural. Applications of these intelligent systems are pervasive and are getting adopted rapidly.

A conversational AI agent, referred to as an agent from here onwards, is a system that uses full conversational dialogue to accomplish one or more tasks. A command-interpreting agent uses enough dialogue to interpret and execute a single command. Event classifiers agents just read the message and perform an action based on the content. These agents have taken the world by storm. There are many practical applications of these assistants, for example, enhancing customer buying experience using guided conversation (conversation agent), using natural language or voice interface to interact with devices (command interpreter) or sorting emails into folders (event classification) (Freed, 2021).

Interactions with these agents are intuitive, effective, and efficient, making the adoption of these techniques of great importance.

1.1 Background

As discussed by Palash Goyal (2018), the initial work on conversational agents date back to 1966 when ELIZA was introduced by Joseph Weizenbaum. However, the recent state-of-the-art in natural language processing and advancements in computing technologies has given a great thrust to the field of conversational AI. As per the research on insider intelligence, 72.3 per cent of the time on mobile devices was spent on smartphones. This is a significant amount of time spent on mobile interfaces, so conversational AI techniques can greatly enhance interactions between customers and organizations by making them simple and effective (Dolan, 2022). The use of advanced language models and transformers shows promising results in building conversational agents that are more than the chatbots that are based on a state machine. Therefore, this project aims to explore state-of-the-art NLP techniques to build a modern AI-based conversational agent.

In the healthcare domain, a good amount of reliable knowledge is available on the internet from authentic sources such as government websites and healthcare research groups. Despite this, general users find it difficult to navigate through the complex healthcare jargon and understand or use this information. Demands for healthcare professional has risen significantly and continues to grow. Many healthcare systems are under tremendous pressure, and this limits the number of patients that can be treated on time. Patients find it difficult to get treatment on time even for simple and mild illnesses. This delay in early treatment exaggerates the symptoms which can lead to further health complications (Laumer et al., 2019). Having a conversational agent-like system can come to the rescue of both patients and healthcare services. Users can ask agents about their health issues like how they normally interact with doctors and nurses. Users do not have to be experts in the medical field to interact with these agents. An agent will be capable of understanding a user's problem, searching its knowledge base for possible solutions and presenting a solution to the user in an understandable format (Gupta et al., 2022). The agent can be trained to handle mundane administrative tasks and reduce the work pressure on healthcare services. Advanced agents can be equipped with voice-based conversational capabilities. With such features, these agents can virtually replace any front desk present in hospitals and medical services. Agents can complete patient registration and data gathering in a timely fashion without any healthcare professional involvement. Agents can also be integrated with local pharmacies for ordering medicines and medical supplies.

Healthcare conversational agents are AI-powered conversational solutions that help users and healthcare service providers to connect easily. Agents can play a critical role in making first-line health services available to anyone who is in need. For example, users can use these agents around the clock and get their queries answered. Further, an easy-to-use interface can facilitate intuitive interactions with these agents. There are endless possibilities about what these agents can accomplish. Some of the interesting capabilities can be symptoms checker and triage, self-care advice, health risk assessment, chronic condition monitoring, appointment booking, medication reminder and tracker, and healthcare tracker in addition to other services. Thus, agents can make extensive medical knowledge available to users in need, so users do not have to wait for doctors' availability for basic illnesses.

Due to the sensitive nature of data in the healthcare domain, healthcare agencies need to be careful while making technological choices. Data protection rights have the utmost importance but not complying with the law and privacy may result in significant financial and reputational damages to organizations (U.S. Department of Health & Human Services, 2022). This project aims to explore conversational AI platforms that give developers and organizations more control and freedom over the system, its components, and data.

1.2 Objectives

In this project, the researcher seeks to build a prototype of a conversational AI assistant called Health Agent for disease and illness matching, which targets to achieve several primary, secondary and tertiary objectives.

1.2.1 Primary Objective

The primary objectives focus on the development of a basic conversational AI (like an interactive Q&A system) where the user mentions all the symptoms in a single input text and receives information on the most likely illness matching these symptoms. The following objectives aim at building a conversational agent with these capabilities:

- Prepare data for training the natural language understanding unit
- Train the natural language understanding unit. The system should be able to read question input and extract appropriate entities and intent
- Prepare scenarios for natural language generations
- Develop the natural language generation part so that the agent can generate texts for responding to the users
- Data processing for symptom and disease matching

1.2.2 Secondary Objective

The conversational agent's dialogue management can be extended for a more coherent conversation with the end users. The goal is to enhance the dialogue management of the agent with the following objectives:

- Extending the dialogue with a multi-turn exchange of information to achieve a more humanly conversation
- Gathering the symptoms of the illness in multiple dialogue turns
- Extending dialogue management system to handle out-of-scope and ambiguous questions

1.2.3 Tertiary Objective

The user's experience can be enhanced with a modern interface and better techniques of a conversational agent's knowledge representation. The following objectives aim to enhance the user's experience:

- Web-based graphical user interface (GUI) to interact with the conversational agent
- Improving conversational features by exploring techniques like knowledge representation using a knowledge graph

This thesis developed an AI-based conversational agent (Health Agent) to handle a multi-turn conversation to collect illness symptoms that the user of the system might be observing and respond with a disease name corresponding to the collected symptoms. The research was conducted to identify an appropriate conversational AI platform for building the agent. The work also involved researching the NLP language models for disease and symptom NER tasks. The most challenging part of the project was integrating the language models into the chosen conversation AI platform and developing a multi-turn conversation to collect symptoms from the user's text. A significant time was spent on generating the training data and identifying the right dataset for disease and symptom matching. Evaluation of a conversational agent can be tricky and significant efforts were required to understand conversational AI testing frameworks. Integrating the agent with a web interface was one of the exciting developments which gave the system a real-life look. All the development was done on-premises and no need arise to upload the data to any third-party platform or cloud service provider, ensuring the privacy of the data.

In the following sections, Chapter 2 will outline the current state of the art in NLP and conversational AI along with some related research work for conversational AI in the healthcare domain. Chapters 3 and 4 discuss the system requirements and development methodologies. Chapter 5 covers the ethical consideration of the project. Chapter 6 will introduce the overall design of the Health Agent including the choice of conversational AI platform and language models used for building the system. Implementation of design choices, generation of data for model training and configurations of the conversational platform are discussed in chapter 7. Chapter 8 evaluates the success of the agent and chapter 9 summarizes work along with suggestions for future work.

2 Context Survey

This chapter will review the state-of-the-art NLP techniques as well as the conversational platforms available in the market. In addition, scholarly articles related to the use of conversational AI in the domain of healthcare and the use of transformers in conversational AI are also reviewed.

2.1 Natural Language Processing

As defined by Patel and Arasanipalai (2021), *"Natural language processing (NLP) is a subfield of linguistics, computer science, information engineering, and artificial intelligence concerned with the interactions between computers and human (natural) languages, in particular how to program computers to process and analyse large amounts of natural language data."*

Freed (2021) argues that every intelligent application involving human language has some NLP behind it. Table 1 summarizes various NLP tasks and corresponding popular applications identified by the author.

Table 1: NLP Tasks and Applications

NLP Task	General Applications
Text classification	Spam classification
Information Extraction	Calendar Event Extraction
Conversational Agent	Personal Assistance
Information Retrieval	Search Engines
Question Answering System	Legal entity Extraction

Human language is a structured system of communication that involves complex combinations of its constituent components, such as characters, words, sentences, etc. Linguistics is the systematic study of language. To study NLP, it is important to understand some concepts from linguistics about how language is structured. One can think of human language as composed of four major building blocks: phonemes, morphemes and lexemes, syntax, and context. NLP applications need knowledge of different levels of these building blocks, starting from the basic sounds of language (phonemes) to texts with some meaningful expressions or context. Table 2 summarizes the building blocks of language and its applications (Vajjala et al., 2020).

Table 2: Language Blocks and NLP Applications

Blocks of Language	Applications
Context (meaning)	Summarization Topic Modelling Sentiment Analysis
Syntax (phrases and sentences)	Parsing Entity Extraction Relation Extraction
Morphemes and Lexemes (words)	Tokenization Word embedding POS tagging
Phonemes (speech and sounds)	Speech to text Speaker Identification Text to speech

2.1.1 Challenges in NLP

Language has inherent ambiguity in a general sense as well. (Perera et al., 2013) have discussed some of the common challenges that NLP faces in clinical settings and below are the challenges for most of the common NLP tasks as described by (Vajjala et al., 2020)

Ambiguity: Most human languages are inherently ambiguous. Many times, sentence has multiple meanings, and the meaning is decided by the context around the sentence. One can draw multiple meanings from the sentence "not bad" depending on the context used.

Common knowledge: Humans use common knowledge all the time to understand and process any language. One of the key challenges in NLP is how to encode all the things that are common knowledge to humans in a computational model.

Creativity: Humans are creative, and language is no exception for creativity. A range of styles, dialects, genres and variations are used in any language. Making machines understand creativity is a hard problem not just in NLP, but in AI in general.

Diversity across languages: For most languages in the world, there is no direct mapping between the vocabularies of any two languages. This makes porting an NLP solution from one language to another hard.

2.1.2 Machine Learning, Deep Learning, and NLP

The boundaries and overlaps between machine learning, deep learning and NLP are well-discussed by Mukhopadhyay (2018). He argues that AI is a subfield of computer science that tries to create systems which can do activities that would normally need human intelligence. He also defines Machine Learning (ML) as a field of AI that focuses on the creation of algorithms which can learn to do tasks automatically based on a large number of instances without the need for hand-crafted rules. Further, he defines Deep learning (DL) as a type of machine learning that uses artificial neural network designs to learn.

Mukhopadhyay (2018) also highlights that while NLP, ML and DL have some overlap, they are also quite independent fields of study. Rules and heuristics were also used in early NLP applications. However, in recent decades, ML approaches have had a significant effect on the development of NLP applications. More recently, DL has been widely developed and applied to natural language processing (NLP) systems.

2.1.3 Approaches to NLP

A variety of approaches have been identified for solving NLP tasks. As argued by (Vajjala et al., 2020), the approaches can be categorized as follows:

2.1.3.1 Heuristics-Based NLP

Early attempts at constructing NLP systems, like other early AI systems, are based on creating rules for the task at hand. This has necessitated the developers having some domain knowledge in ways to construct rules that could be put into a system. Such systems also need dictionaries and thesauruses. More extensive knowledge bases have been constructed to facilitate NLP in general and rule-based NLP in particular, in addition to dictionaries and thesauruses. Wordnet (Miller, 1995), for example, is a database of words and semantic ties that exist between them. More recently, common-sense world knowledge has been included in knowledge bases such as Open Mind Common Sense which supports rule-based systems (Singh et al., 2002). Regexes are a common paradigm for creating rule-based systems, and NLP software like StanfordCoreNLP contains a framework for developing them. Context-free grammar (CFG) is a sort of formal grammar used to model natural languages. Moreover, grammar languages like JAPE (Java Annotation Patterns Engine) may be used to model more sophisticated rules.

2.1.3.2 Machine Learning for NLP

For many NLP applications, supervised machine learning approaches such as classification and regression algorithms are widely used. The extraction of features from the text, the use of the feature representation to develop a model, and the evaluation and improvement of the model are all typical phases in any machine learning technique for NLP. Some of the commonly used ML algorithms are Naive Bayes and support vector machine (SVM) for classification tasks, hidden Markov model (HMM) conditional random field (CRF) for part-of-speech (POS) tagging (Vajjala et al., 2020).

2.1.3.3 Deep Learning for NLP

The AI field has seen a big increase in the use of neural networks (NN) to deal with complicated tasks in recent years. As human language is naturally unstructured and complicated, the use of NN models better represents the complexity of language and produces improved outcomes.

Recurrent neural networks (RNNs) are specifically intended to keep such sequential processing and learning in mind since human language is fundamentally sequential. RNNs have neural units that can remember what they have processed previously. This memory is temporal, and when the RNNs read the next word in the input as well as store and update the information at each time.

The problem of forgetting memory is a challenge that RNNs face. To address this problem, Long Short-Term Memory networks (LSTMs), a form of RNN, were developed. LSTMs get around this difficulty by ignoring irrelevant information and memorising just the parts of it that are important to the job at hand. This alleviates the burden of memorising a large amount of information in a single vector representation. Because of this solution, LSTMs have largely replaced RNNs in many applications.

Another type of network called Convolutional neural networks (CNNs) are widely employed in computer vision applications such as image classification and video recognition, among others. CNNs have also shown promise in NLP, particularly in text categorization. The capacity of CNNs to use a context window to look at a collection of words together is their major benefit (Vajjala et al., 2020).

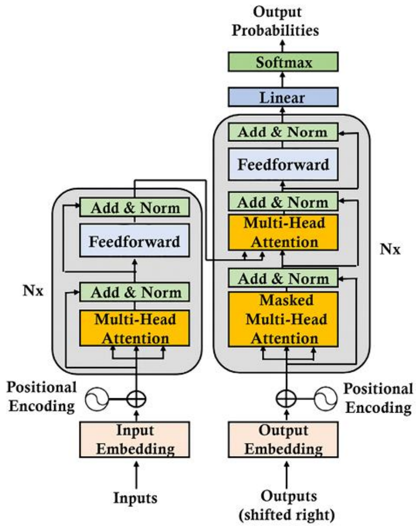
2.1.4 Transformers

This section reviews the state-of-the-art techniques called transformers used in many tasks of natural language processing. The transformer model was released in 2017 (Vaswani et al., 2017), and it provided remarkable results on machine translation tasks. In the last two years, transformer models have surpassed state-of-the-art in practically all key NLP tasks. They model the textual context, but not in the order in which it appears. Rather they prefer to look at all the words surrounding it known as self-attention and represent each word in its context when given a word in the input.

According to (Vajjala et al., 2020), the transformer's huge success has sparked the interest of numerous NLP researchers. They have created even more developed transformer-based models, including Generative Pre-trained Transformer (GPT) and Bidirectional Encoder Representations from Transformers (BERT) which are two of the most well-known and essential models. GPT is entirely made up of the decoder layer of the transformer, whereas BERT is entirely made up of the encoder layer of the transformer. The purpose of GPT is to create text that appears to be written by a human, whereas BERT's purpose is to give a better language representation to aid a variety of downstream activities (sentence-pair classification tasks, single-sentence classification tasks, question-answering (QA) tasks, and single-sentence tagging tasks) in achieving better outcomes.

Figure 1 shows the original transformer architecture proposed by Vaswani et al. (2017) which is based on a self-attention mechanism relating to different positions of a single sequence (positional encoding) to make sense of the entire sequence. Notably, the original transformer model was employed for the language translation task.

Figure 1: The Transformer Model Architecture



Note: Image sourced from Vaswani et al. (2017)

Each encoder structure converts an input sequence of tokens into a sequence of embedding vectors, often called the hidden state or context. The structure consists of two layers: multi-head self-attention mechanisms layer and position-wise fully connected fee forward layer. Each decoder structure uses the encoder's hidden state to iteratively generate an output sequence of tokens, one token at a time. The structure has an additional layer to perform multi-head attention over the output of the encoder stack (Vaswani et al., 2017).

Only the lowest level of the stack contains the embedding sublayer. The encoded input is guaranteed to remain stable through all other layers because there isn't an embedding layer in the other layers.

The self-attention which is a token-to-token operation has replaced the recurrence function that is present in RNN, LSTM or CNN. The attention mechanism will determine how each word in a sequence, including the word being processed, relates to every other word in the sequence which results in finding a deeper relationship between the words and producing better results. This layer applies three independent linear transformations to each embedding to generate the query, key, and value vectors. These transformations project the embeddings, and each projection has a unique set of parameters that can be learned. This enables the self-attention layer to concentrate on various semantic facets of the sequence. Having several heads allows the model to focus on several aspects at once and that is achieved with multi-head attention (Vajjala et al., 2020).

The positional encoding layer adds positional information to input embedding which helps in establishing relationships between words which are semantically closely related but positionally far apart in the sentence.

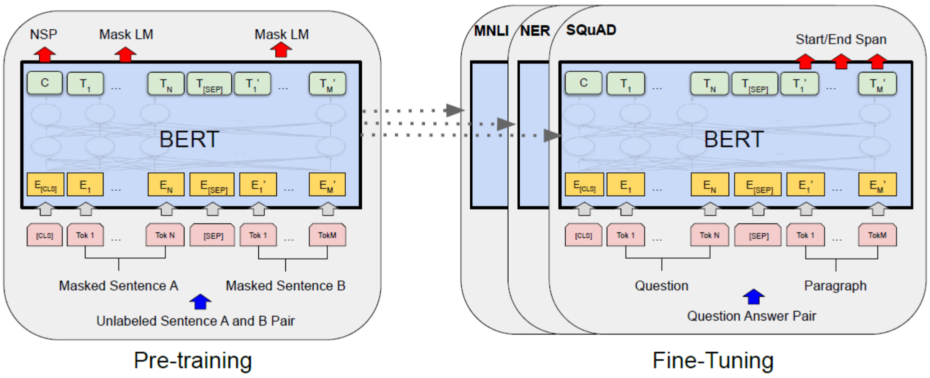
The following section will discuss one of the important transformer architectures which have applications in many NLP tasks.

2.1.5 BERT

Bidirectional Encoder Representations from Transformers (BERT) is an encoder-only architecture of a transformer. As argued by Devlin et al. (2018) BERT is pre-trained with two objectives: predicting masked tokens in texts (masked language modelling task or MLM) and determining if one text passage is likely to follow another(next sentence prediction or NSP).

BERT training has two steps: pre-training and fine-tuning. During pre-training, the model is trained on unlabelled over different pre-training tasks. For finetuning, the BERT model is first initialized with the pre-trained parameters and then these parameters are fine-tuned using labelled data. Each downstream task has separate fine-tuned models though they were initialized with the same pre-trained parameters. During fine-tuning, task-specific inputs and outputs are plugged into BERT and all the parameters are fine-tuned. Compared to pre-training, fine-tuning is relatively inexpensive (Devlin et al., 2018). Figure 2 shows the overall pre-training and fine-tuning procedures for BERT.

Figure 2: Overall Pre-training and Fine-tuning Procedures for BERT



Note: Image sourced from Devlin et al. (2018)

2.1.6 Transfer Learning

Transfer Learning (TL) is an approach in which information obtained while addressing one problem is used to solve a related but different problem. It is now a normal practice in the field of computer vision to employ transfer learning to train a CNN on one job, and then to adapt it to or fine-tune it on a different task. Because of this, the network can make use of the information that is learned in the original task. Large transformers have recently been employed in the TL of smaller downstream activities. On one hand, TL needs less training data and, on another hand, usually, not much training data is available in the conversational AI domain. TL can achieve much better performance on the limited amount of training data making training faster needing only a few training epochs to fine-tune a model for a new task. Generally, it is much faster and more efficient than the traditional ML method (Tunstall et al., 2022).

2.1.7 The Hugging Face Hub

It is a common practice to share software artefacts using a standardized platform and the same applies to transformer models. The Hugging Face (Hugging Face, 2022) provides a standardized interface to a wide range of transformer models as well as code and tools to adapt these models to new use cases. TensorFlow, PyTorch and JAX are the three main deep learning frameworks that the Hugging Face presently supports, and it makes it simple to transition between them. It also gives task-specific heads so you can fine-tune transformers on downstream tasks like text classification, named entity recognition, and question answering. Because of this, it takes less time to train and test a variety of models. The ecosystem consists of mainly two parts: a family of libraries and the Hub. The libraries provide the code while the Hub provides the pre-trained model weights, datasets, scripts for the evaluation metrics, and more.

2.2 Conversational Agents

A conversational agent is a computer programme that can converse with humans more naturally (Williams, 2018). These agents can be integrated into existing websites and applications and provide voice or text-based interactions. He identified three main components of a conversational agent which will be discussed in the next section.

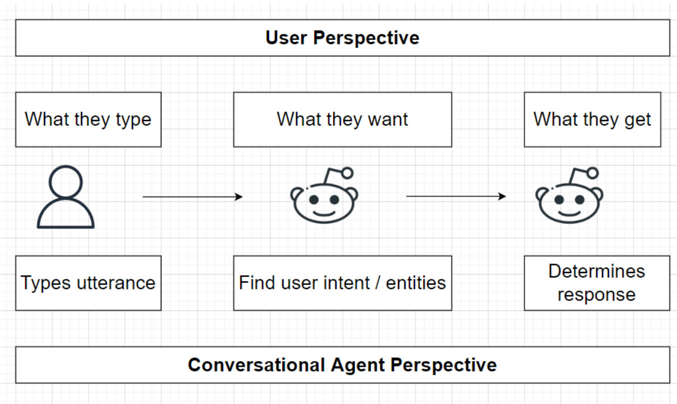
2.2.1 The Main Components of a Conversational Agent

While discussing conversational agents, it is important to understand some of the key terminologies and components that are often part of an agent. As mentioned by Williams (2018) and Freed (2021), the following are the three main components of an agent:

- **Utterance**: The user interacts with the agent through natural language. Whatever a user types in the web client are an utterance. It is also a response generated by the agent to reply to a user.
- **Response:** A response is whatever the assistant returns to the user. It can be textual or multimedia.
- **Intent:** An intent is a normalization of what the user means by their utterance or what they want to do.
- **Entity:** An entity is a noun-based term or phrase. An entity can be any important detail that your assistant could use later in a conversation
- **Slots:** Slots hold key information for performing the task the user intends to do. Slots can hold only a specific form of information and are domain dependent.

The same information is presented from two perspectives here: the user's perspective and the agent's perspective. These conversational terminologies become evident when seen from both sides. Figure 3 depicts the user and conversational agent’s different perspectives on the same element of text-based communication.

Figure 3: Conversation Perspectives



Note: Image referenced from Freed (2021)

Kong et al. (2021) categorise conversational agents based on their goals: task-oriented agents and chitchat agents. Task-oriented agents aim to do certain tasks by engaging with humans such as purchasing a flight for someone, whereas chitchat agents are more like real beings - their purpose is to answer users' messages, exactly like in natural chitchat. Some scenarios in which a conversational agent may have an advantage are hospital reception, medical consulting, online shopping customer service, after-sales service, investment consulting and bank services. A method for conversational agent development has been established to help developers follow standard practices.

Developing a conversational agent generally comprises five distinct parts, which are shown below:

- Automated Speech Recognition (ASR) to convert user speech into text
- Natural Language Understanding (NLU)to interpret user input
- Dialogue Management (DM) to make decisions on the next action concerning the current dialogue status
- Natural Language Generation (NLG) to generate text-based responses to the user
- Text-To-Speech (TTS) to convert text output into voice

2.2.2 Benefits of Conversational Agent

As summarized by Brush and Scardina (2021) and Singh et al. (2019), below are some of the major benefits of conversational agents:

- **Reduce waiting time:** Conversational agents can be scaled to meet the demand and the customer's wait time can be greatly reduced
- **Less expensive:** Training a conversational agent is less regressive as compared to training the human workforce.
- **Increase availability:** Conversational agents can be deployed to work 24/7 to cater global needs of the customers
- **Better customer experience:** Conversational agents can cater specific needs of the customer and can provide quick solutions.

2.2.3 Success Metrics for a Conversational Agent

At the outset of any conversational agent development project, success metrics must be defined. The metrics serve as a reference point for the solution and the agent's intended benefits. Singh et al. (2019) have mentioned the following points as the success metrics:

2.2.3.1 Customer Satisfaction Index

The Customer Satisfaction Index (CSI) gauges the effectiveness of customer service interactions by following up with the customer and asking them to complete a brief survey about their experience.

2.2.3.2 Completion Rate

The Completion Rate (CR) is defined as the percentage of agent's conversations that resulted in the customer's solution being addressed.

2.2.3.3 Bounce Rate

The bounce rate (BR) is the percentage of users that leave the chat after entering one or two inputs into the conversational agent. A high bounce rate indicates that the agent was not effective in engaging the user, which should be reflected in the part of the customer feedback.

2.2.4 Conversational AI Frameworks

There are two types of solutions for creating conversational agents: closed-source solutions and open-source solutions. Each of the solutions has its pros and cons.

2.2.4.1 Microsoft

Microsoft offers separate Azure Cognitive Services: Language Understanding Intelligent Service for natural language understanding and Bot Framework for dialogue and response (Microsoft, 2022). These are the cloud-hosted services.

2.2.4.2 Amazon

Amazon Lex is the primary service used for building AI assistants and integrates easily with Amazon's other cloud-based services as well as external interfaces (Amazon Web Services, 2019). This is a cloud-hosted service.

2.2.4.3 Google

Google's Dialogflow is a service by Google cloud for building Lifelike conversational AI with state-of-the-art virtual agents (Google, 2022). This is a cloud-hosted service.

2.2.4.4 IBM

Watson Assistant is IBM's AI assistant platform for building conversational AI experiences for businesses. (IBM, 2022). This is a cloud-hosted service.

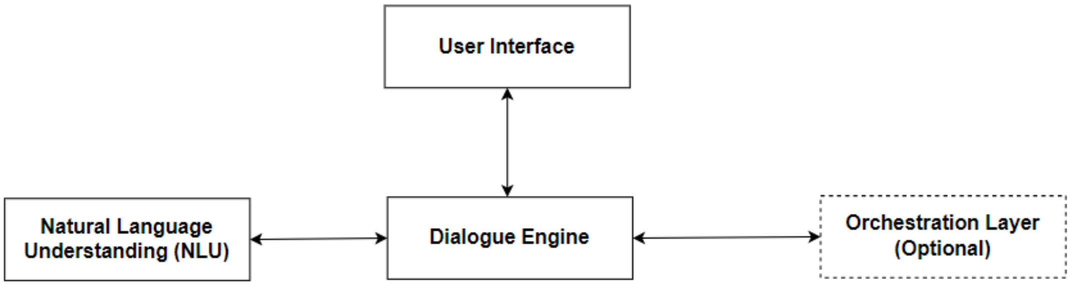
2.2.4.5 Rasa

Rasa is an open-source solution with all the industry-standard features: Built-in enterprise-grade concurrency capabilities, rich functions covering all the needs of conversational agents, rich documents and tutorials, and a huge global community. Rasa provides you with complete control over the applications that you deploy. Other platforms allow you to control the classifier by changing the training data, but Rasa allows you to customize the entire classification process (Rasa Technologies Inc., 2022).

2.2.5 Generalized AI Assistant Architecture

A generalized AI assistant consists of the following four components as described by Freed (2021). Figure 4 shows the generalized AI assistant architecture.

Figure 4: Generalized AI Assistant Architecture



Note: Image adapted from Freed (2021)

The components of the above architecture are explained further:

Interface: Used by the end user to interact with the conversation agent and can be text or voice-based

Dialogue Engine: Maintains dialogue state and predicts next action

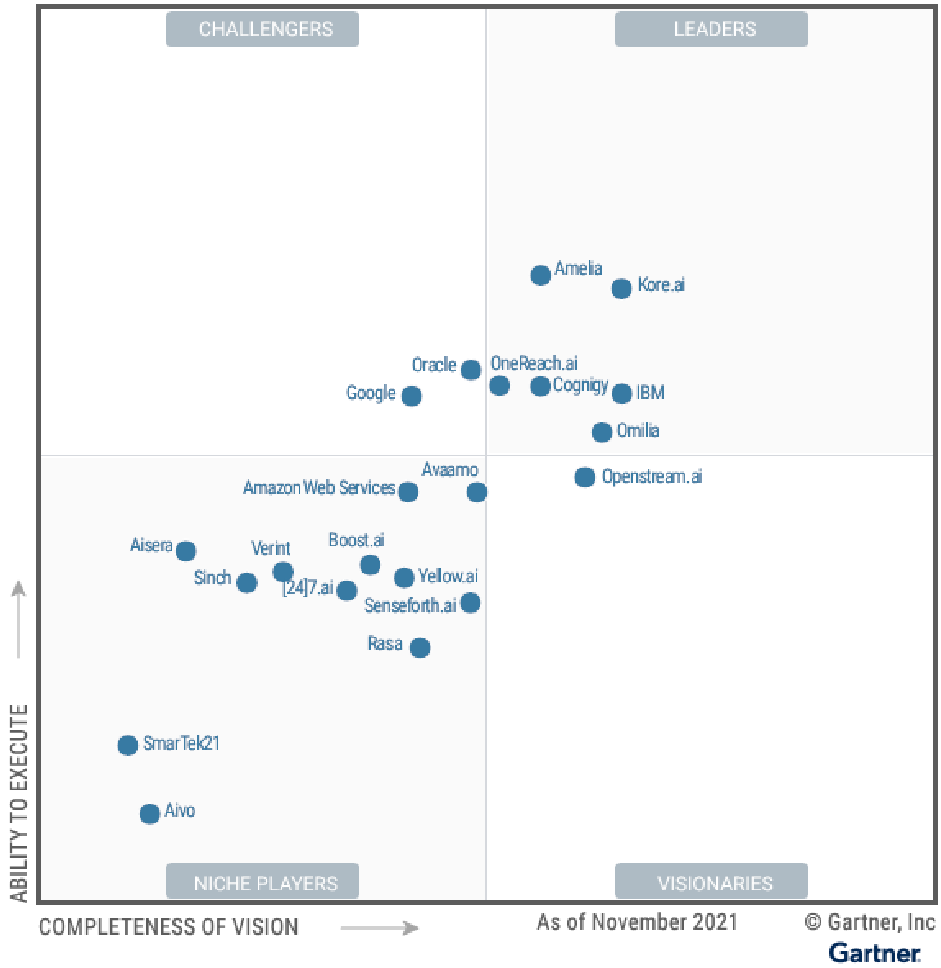
Natural Language Understanding: Used to extract meaningful information from the user's input

Orchestrator: Communicates with business APIs for driving business processes and generating the dynamic response. This component can be optional.

2.2.6 Rasa

The conversational agent market has seen increased interest and rapid adoption over the last five years. During that time, the products and platforms enabling the building of conversational agents have been extremely diverse and have evolved considerably. These solutions can be divided into two types: closed-source solutions and open-source solutions. Closed-source solutions come with a high cost, vendor lock-in, risk of data leakage, and the inability to implement custom functions. Open-source solutions have an advantage in this regard. A downside of open-source solutions is that users need to be careful in choosing the framework and make sure it is scalable and concurrent and has wide community support. Rasa (Rasa Technologies Inc., 2022) is the open-source, industry-grade conversational AI framework that meets these requirements. Many companies have successfully built their conversational agents using Rasa. Rasa has been recognized as a Niche Player in Gartner®Magic Quadrant™ (Revang et al., 2022) which is shown in Figure 5 below. For the privacy-conscious healthcare domain, the open-source nature of Rasa offers complete control over all the aspects of conversational agent development and operations.

Figure 5: Magic Quadrant for Enterprise Conversational AI Platforms



Note: Image sourced from Revang et al. (2022)

2.3 Related Work

This section will discuss existing work that makes use of the conversational agent in the healthcare domain and the use of state-of-art NLP techniques in the conversational AI field. It was observed that the existing work on the conversational agent is not using transformers and transfer learning for symptom detection. This research seeks to make use of these techniques in the conversational agent in the healthcare domain.

2.3.1 Conversational Agent for Healthcare

Athota et al. (2020) have proposed the implementation of a conversational agent for healthcare systems using AI. The author has implemented the conversational agent using Java technology and has made use of N-gram, TF-IDF and cosine similarity techniques for NLP tasks. The focus of the paper is to achieve conversational agent functionality in a question-answer format. Kandpal et al. (2020) have discussed the implementation of a contextual conversational agent for healthcare purposes. The paper focuses on the use of deep learning for achieving NLP tasks. Sheth et al. (2019) focus on the contextualization and personalization of patient data. The paper discusses how existing conversational agent systems can be extended by a whole ecosystem of the Internet of Things for better and personalized health tracking of individuals. It has mentioned the usefulness of knowledge graphs in data representation.

Nadarzynski et al. (2019) have discussed the acceptability of AI-led conversational agents in the healthcare domain, finding that there is hesitancy regarding the acceptance of conversational agent technology. The paper argues user's perspective, motivation and capabilities need to be considered during conversational agent development. Laumer et al. (2019) have explained user adoption of conversational agents for disease diagnosis.

A paper by (Rakib et al., 2021) on mental healthcare highlights the usefulness of BiLSTM (Bi-directional LSTM) and Sequence-to-Sequence (Seq2Seq) encoder-decoder architecture and has used the Bilingual Evaluation Understudy (BLEU) score for model evaluation. A paper by IIT Delhi researchers (Pandey et al., 2020) has worked on studying the Q&A support system for maternal and child health in rural India. Researchers at MDPI have studied the feasibility of developing a rule-based virtual caregiver system using a mobile conversational agent for elderly people(Miura et al., 2022).

2.3.2 Transformer for Conversational AI

A paper by Yu et al. (2020) has studied the use of a bi-directional transformer for financial service conversational agents. They have shown how the BERT model outperformed other methods for common NLP tasks like intent classification, sentence completion, information retrieval and question answering. A paper from Microsoft researchers (Damani et al., 2020) has discussed optimized transformers for FAQ answering. A paper by hugging face researchers (Wolf et al., 2020) has a detailed explanation of how transformers have reshaped state-of-the-art natural language processing. A paper published by MODUL Technology GmbH (Braşoveanu & Andonie, 2020) focuses on explaining Transformer architectures through visualizations. Gillioz et al. have provided an Overview of the Transformer-based Models for NLP Tasks.

3 Requirements specification

This chapter discusses the functional and non-functional requirements of the project. The targeted audience of the system can be a user who wants to understand medical symptoms and potential diseases associated with those symptoms. This is a prototype and should not be considered a replacement for any medical advice or diagnosis.

3.1 Functional Requirements

The system must be able to achieve the following functionalities:

- The agent should introduce itself and its purpose
- The system should be able to accept text inputs from the user
- The agent should be able to determine if patients want to discuss their symptoms or just informing about their good health
- When a patient is not well, the agent should be able to ask for symptoms, collect and try to match those symptoms with its disease dataset to determine possible disease
- The agent should be able to answer specific questions outside the current conversation context
- The agent should be able to ask for clarifications when information shared by the user is not clear

3.2 Non-functional Requirements

As an end user of the system, the patient should be able to interact with the system via easy-to-use interface. Complexities of the system should be transparent to the user. Users need not be healthcare experts to make complete use of the system. A user guide with instructions on how to operate the system should be documented.

4 Software Engineering Process

This chapter introduces software engineering practices that were followed during the research.

4.1 Planning

The initial planning of the project was done in the first two weeks of the project work. A Gantt chart with details of tasks and allocated efforts for each task were prepared. The chart is included in Appendix A.

4.2 Development

The agile methodology was followed during the development of the Health Agent. Features were developed in incremental cycles to reach the final product. The entire development was broken into small trackable tasks and tasks were prioritized during weekly project discussions.

4.3 Tracking

Similar to sprint review meetings, a weekly meeting helped in reviewing the project's progress. Thursday meetings were used for all the critical discussions, brainstorming and tracking of the work done in the week and work items for the coming week. MS Teams channel was used for maintaining the backlog items.

4.4 Testing

The manual and automated testing strategy was applied to ensure the agent is working as per expectations.

Manual Testing: Initial Health Agent testing was performed manually using Rasa interactive command line features. After integrating the agent with a web interface, manual testing was performed using a web browser.

Automated Testing: Python unit test cases were written for testing backend service is functioning as expected. The test result summary is included in Appendix B.

4.5 Version Control

The GitHub repository was used for maintaining the code base of the project. It is a private repository and access can be provided on a need basis until it is made publicly available.

4.6 Final Artefacts

The Rasa train creates trained models in the archive format with a naming convention `<timestamp>.tar.gz`.

User interface artefacts are static and do not need compilations. Detailed instructions on using the artefacts are mentioned in the README.md hosted in the version control. The same is included in Appendix D.

5 Ethics

Ethics evaluation is done following the artefact evaluation form. Throughout the development and evaluation, no personal information was collected. The system is a prototype and is not supposed to replace any existing healthcare advice mechanisms.

The evaluation was performed in line with the artefact evaluation form by demonstrating the Health Agent to the staff and students at the University. A fictitious conversation between a user and the agent was recorded and anonymous feedback on the quality of the artefact was collected using the Qualtrics survey system.

The ethical approval document is included in Appendix E.

6 Design

This chapter discusses the system design considerations that were made during the development of the Health Agent. The Health Agent is closed domain and task-oriented conversational AI Assistance. The chapter covers conversation flow design, the architecture of the system and technical design choices made during the Health Agent development.

The proposed prototype of an AI-enabled conversational virtual agent named Health Agent is built on top of an open-source conversational AI platform called Rasa. Rasa helps develop virtual assistants which are HIPAA (U.S. Department of Health & Human Services, 2022) and has built-in privacy features. The open-source nature of the Rasa platform provides better control and transparency to the solution built on top of it.

6.1 Conversation Flow Design

The first step towards building a conversational agent is identifying the conversations that the system is supposed to handle. Below is an example of dialogue that was covered for the primary objective.

symbol represents a human interacting with the conversational agent

symbol represents the conversational agent

6.1.1 Conversation Scenario of a Healthy Patient

When conversing with a healthy user, the agent will understand that the user is healthy and will not ask further questions. The agent motivates the user to stay fit and ends the conversation gracefully.

Hello

Hello there! How are you doing today?

I am good today.

Fantastic, keep moving!!

thanks!

Good luck

6.1.2 Conversation Scenario of an Unhealthy Patient

When users start a conversation with the conversational agent and inform the agent that they are unwell, the agent proceeds with asking about symptoms observed by the user. During the conversation, instead of informing symptoms, a user asks some random and out-of-scope questions and also asks about the agent, which the Health Agent handles gracefully and continues asking about symptoms. The Health Agent then extracts symptoms from the text entered by the user and keeps informing the user about the progress that the agent is making. Health Agent then informs the user about the disease that has matched the symptoms entered by the user and then ends the conversation gracefully. Thus, the Health Agent is capable of handling context switches while maintaining the conversation flow with the user.

Hello

Hello there! How are you feeling today?

i am not feeling well today.

Can you please tell me what symptoms you are observing?

who are you

I am a Health agent and your virtual assistant. Can you please tell me what symptoms you are observing?

can you make pasta?

Sorry, I can't handle that request. Can you please tell me what symptoms you are observing?

I have neck pain, fatigue and blurred vision

Ok. I have collected all the required data. As I could understand, you have entered the following symptoms: neck pain, fatigue, blurred vision. Checking for diseases corresponding to the entered symptoms. Checking for diseases corresponding to the entered symptoms. Your symptoms are matching with these diseases: Ankylosing spondylitis

thanks!

Good luck

bye!

bye!

6.1.3 Advanced Conversation Scenario of an Unhealthy Patient

In addition to previous scenarios, with this enhancement, the Health Agent asks if the user is observing more symptoms and keep asking till the user has more symptoms to inform.

Hello

Hello there! How are you feeling today?

i am not feeling well today.

Can you please tell me what symptoms you are observing?

I have neck pain

Do you have any other symptoms?

I also have fatigue and blurred vision

Do you have any other symptoms?

No

I have collected all the required data. As I could understand, you have entered the following symptoms: neck pain, fatigue, blurred vision. Checking for diseases corresponding to the entered symptoms. Checking for diseases corresponding to the entered symptoms. Your symptoms are matching with these diseases: Ankylosing spondylitis

thanks!

Good luck

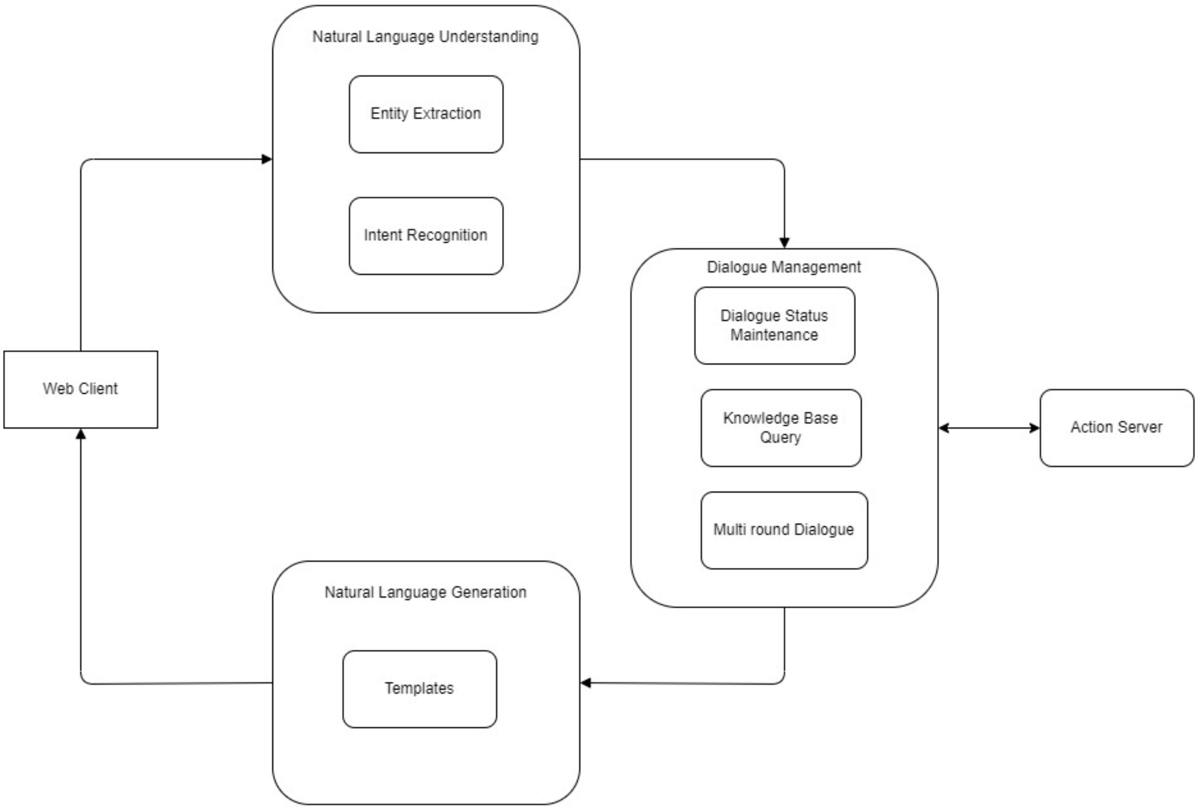
bye!

bye!

6.2 Architecture

Health Agent is a Rasa-powered conversational agent that interacts with users through a web-based interface chat widget and manages dialogue using state-of-the-art (SOTA) techniques in natural language processing. Figure 6 shows the architecture of the Health Agent.

Figure 6: System Architecture of Health Agent



Note: Image adapted from Khilji et al. (2020)

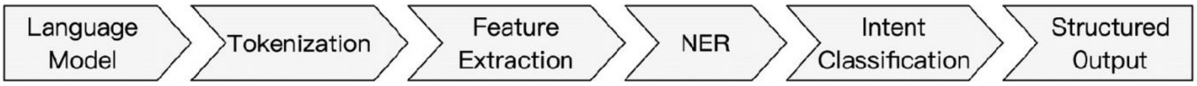
The key components of the Health Agent architecture are as follows:

6.2.1 Natural Language Understanding Unit (NLU)

In the case of the Health Agent, NLU interprets text inputs. With Rasa, NLU is a data processing pipeline that converts unstructured user messages into intents and entities. The pipeline consists of a series of configurable and customizable components.

In Rasa, the config.yml file defines the steps and components of the NLU pipeline. The pipeline takes text as input and processes it to generate intents and entities as output. Figure 7 shows typical Rasa NLU pipeline internals.

Figure 7: Rasa NLU Pipeline Components



Note: Image sourced from Kong et al. (2021)

There are different types of components that you can expect to find in a pipeline. The function of each component is as follows:

- Language model component: Loads language models that will be used by downstream tasks
- Tokenizer component: Splits text into tokens
- Featurizer component: Extracts features from token sequences
- Entity extractor component: Performs NER on featurized text
- Intent classifier component: Classifies text into different predefined intents according to context.
- Structure output Organizes the prediction results into structured data and outputs it.

6.2.2 Dialogue Management (DM)

The main task of the DM module is to coordinate and manage the whole conversation flow and is particularly important for multi-turn task-oriented dialogue. It also predicts the next best action that the agent needs to perform. Rasa provides rule-based, ML-based and hybrid models for DM. The health agent makes use of rule-based and ML-based models. Rule-based models are useful for handling strict conversational behaviour where the next action depends on the current turn. Machine learning-based models consider the entire dialogue context, the previous dialogue turns, extracted entities and slots for dialogue management. TED policy and Memoization policy are used in the ML-based approach. DM module also provides fallback capabilities in case of ambiguous intent (Rustamov et al., 2021). If the intent is not clear, DM instructs the agent to either repeat the question or send fall back response to let the user know about the ambiguity.

6.2.3 Natural Language Generation (NLG)

NLG is almost the last challenging mile in human-machine interaction. Once the next action is chosen by DM, NLG generates the text of the response to the user. In other words, NLG converts the agent's response into human-readable text. There are broadly two ways of doing this: template-based method and deep learning (DL) based method. The template-based method creates a response without much flexibility or variety. DL-based methods are capable of generating a quite dynamic response; however, it is difficult to control the quality and stability of the result.

In the case of task-oriented conversational agents like Health Agent, users need accurate and concise responses to their queries. Hence Health Agent has used a template based NLU in our architecture. A degree of flexibility can be added to the agent’s response by creating a pool of templates.

6.2.4 Action Server / Backend:

The DM module may require interactions with databases, APIs or third-party integration to get extra information to generate responses to user queries or complete the intended task. DM might be interested in implementing custom actions which might be complicated as compared to built-in response generation. For all such scenarios, Rasa provides an action server that runs custom actions for a Rasa Open-Source conversational assistant.

When your assistant predicts a custom action, the Rasa server sends a POST request to the action server with a JSON payload including the name of the predicted action, the conversation ID, the contents of the tracker and the contents of the domain. When the action server finishes running a custom action, it returns a JSON payload of responses and events. The Rasa server then returns the responses to the user and adds the events to the conversation tracker.

6.2.5 Web Client

The web client is responsible for collecting text input from the user and delivering it to Rasa NLU. It also renders the response generated by NLG and presents it to the user. Rasa open source does not provide a built-in GUI client. Developers can integrate Rasa open source with a channel or web interface of their choice.

6.3 Additional Rasa Concepts

Some of the conversational technologies are Rasa specific (Rasa Technologies Inc., 2022)and as the Health Agent makes use of these Rasa features, the same are discussed briefly here:

- **Stories:** This is training data that is used to train agents' dialogue management systems. It is represented in the form of a conversation between a user and a conversational agent. Stories are made up of intents, actions, entities and forms. Based on the stories, models can generalize to unseen conversations.
- **Rules:** Rule is also part of the dialogue management unit's training data. They cover the strict conversation path between a user and a conversational agent.
- **Slots:** The slots act as a long-term memory of the agent. Information stored in slots is generally used later in the decision-making process. Slots are generally filled using entities but not mandatorily.
- **Forms:** Slot filling is a common conversation pattern used to collect pieces of information from a user to do something. Rasa recommends the use of Rasa forms for slot filling. It is a controlled way of slot-filling from extracted entities or text (and is customizable).

6.4 Rasa Data Files

Rasa uses YAML to manage all the training data. As described by Ben-Kiki et al. (2009), YAML is a data serialization language. It is easy to read by humans, portable between languages, expressive, extensible and easy to use. The training data consists of NLU data, stories and rules.

NLU is responsible for understanding the user's intent and the information user is entering. The NLU data consists of intents that the agent can understand and sentence patterns corresponding to the intent. Rasa intents may contain patterns for entity extraction, but this is optional. NLU data defines all the intents that the agent needs to use for intent classification. It also has information about extracting entities from user text.

Dialogue management data consists of stories, rules, slots, forms and policies. Stories and Rules are the representation of a conversation between the user and assistant. The stories vaguely cover most of the conversational scenarios that are expected to be covered by the system. Dialogue management data is used to train the model that will be used to predict the next action(s) in each turn.

6.5 Rasa Config Files

Rasa configuration files hold part of the information for model training (other information is present in data files) and execution of the model. The Health Agent makes use of the following Rasa config files.

- **config.yml:** Defines components as well as policies that Rasa's trained model will use for making predictions after accepting user inputs.
- **credentials.yml:** It contains credentials for voice and chat platforms that the agent will be integrated with.
- **domain.yml:** It holds all the information that the agent needs to know. It specifies intents, entities, slots, responses, forms, actions and session configurations.
- **endpoints.yml:** It contains all the API endpoints and their configuration that the agent can use.

6.6 Rasa Policies

An important task in DM is to predict the next action based on the current state (context) of the conversation. Rasa supports machine-learning and rule-based policies, and both can be combined and used separately. During each conversation turn, policies predict the next action along with a confidence level with that action. ML-based policies consider the state of the conversation. Rule-based policies are used to handle strict conversations between agent and user. The policies are configurable, and Rasa also supports custom policies. Rasa maintains the conversation state in a tracker object.

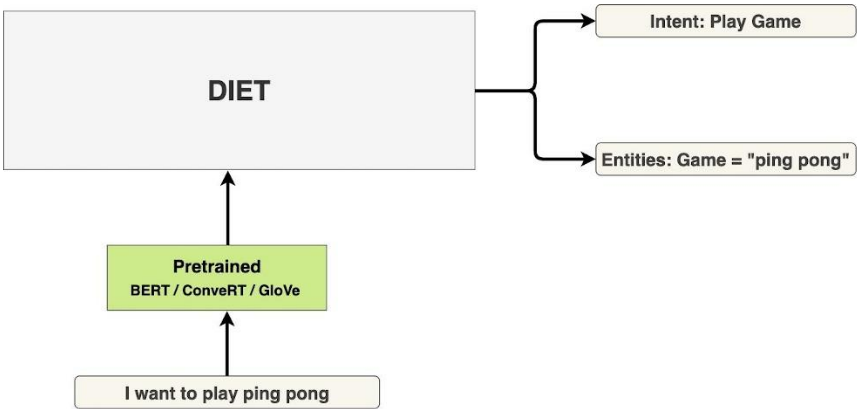
6.7 Spacy Language Models

One of the key tasks of the Health Agent is the extraction of symptoms from the user's text. Listing all the symptoms and the variations in its use is a memory and compute-intensive task. (Tarcar et al., 2019) has argued that the use of the Spacy model can be used for healthcare NER as the models are pre-trained on BioMedical data. Even with minimum additional training data, Spacy models can be tuned for better disease and symptom NER. Thus, we benefit from the transfer learning while working with even a limited amount of training data.

6.8 Dual Intent and Entity Transformer

The Dual Intent and Entity Transformer (DIET) is a multi-task transformer architecture which can be trained to handle both intent classification and entity recognition together. Various pre-trained embeddings can be plugged into the DIET. It employs a sequence model that considers word order, resulting in improved performance. Bunk et al. (2020) argue that the best set of pre-trained embeddings for DIET on NLU benchmarks has outperformed fine-tuned BERT and training DIET is six times faster. Figure 8 provides a high-level illustration of how DIET is used.

Figure 8: High-Level Illustration of DIET



Note: Image sourced from Mantha (2020)

6.9 Data Usage and Accessibility

The data used in the research is available publicly Only the researcher will have access to the raw evaluation data and this data will be destroyed at the end of the project. Aggregated data and results are published in this report.

6.9.1 Pre-trained Data

The Spacy language model used for Disease NER is trained on BC5CDR corpus and is made available under Apache License 2.0. The corpus is published by the National Library of Medicine of the National Centre for Biotechnology Information. The corpus has 6,892 diseases mentioned. More information about the corpus and datasets can be found on the website maintained by National Center for Biotechnology Information (2013)

6.9.2 Disease Dataset for Symptom Matching

The data used for symptom matching is sourced from Kaggle (M, 2020). It contains information about diseases, symptoms and medical treatments scrapped from Mayo Clinic (2022). The dataset does not contain any personally identifiable information (PII).

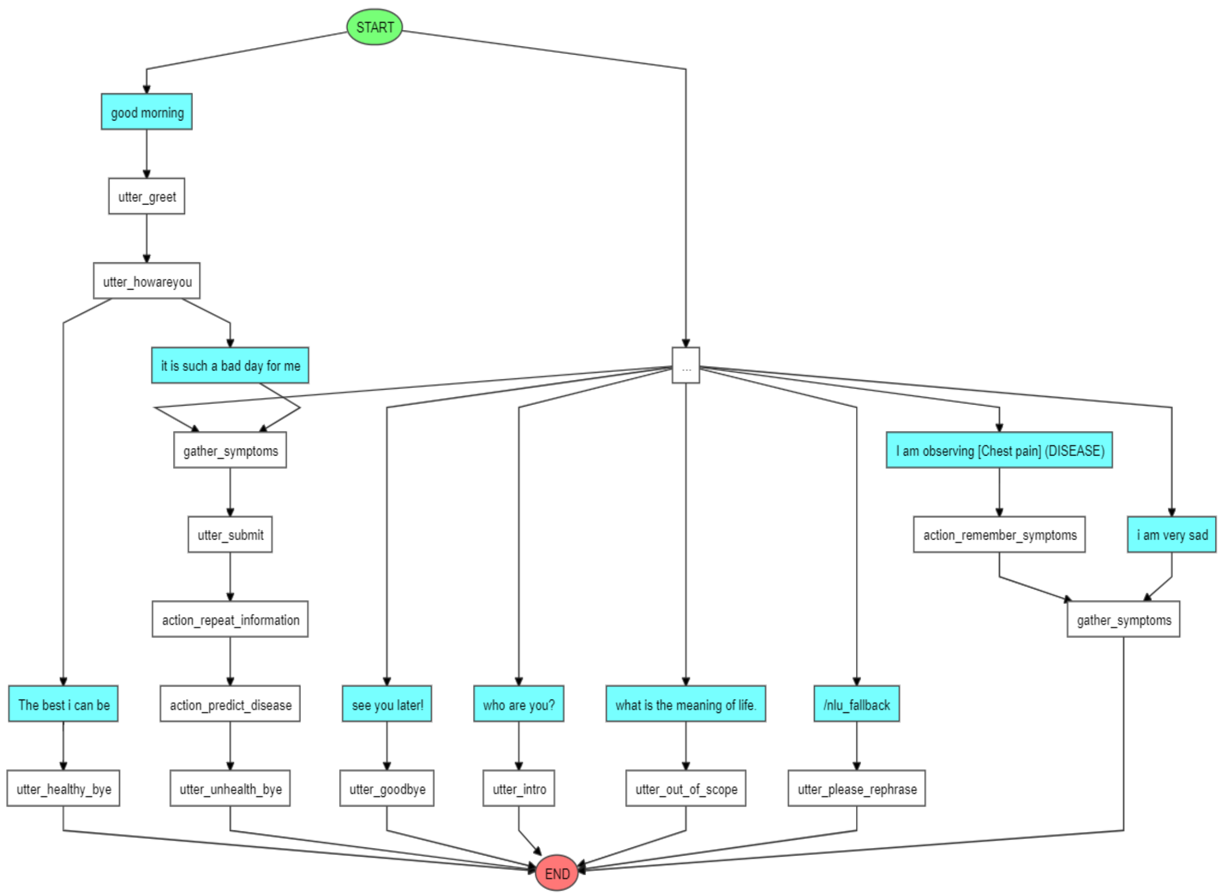
7 Implementation

This chapter elaborates on the implementation details of architecture that are presented in the design section of Health Agent.

7.1 Conversation Flows

A conversation flow consists of an exchange of information between a user and the agent. In Rasa, the flows are coded using stories and rules. Figure 9 shows the visual representation of stories that constitute conversations between a user and the Health Agent.

Figure 9: Health Agent Conversation Flow Graph



For example, the flow (leftmost) starts with the user saying hello and the Health Agent responding with a greet utterance and how-are-you utterance. When the user replies ‘I am very active today ’, the Health Agent understands the user is healthy and ends the conversation with a healthy-bye utterance.

7.2 NLU Data Generation

As a part of NLU data generation, certain intents were identified based on the expected conversation between the Health Agent and a user. This section explains all the NLU data components that have been used by the system.

7.2.1 Intents

As part of NLU data, the following intents were created and added to data files (YAML files). With the help of these intents and example sentences provided for each intent, the NLU model will be trained for a multi-class classification problem to classify each text entered by a user into categories identified by intents.

Table 3 summarizes the intents that the system can process, why the intent is created and an example showing the text that will get classified with the corresponding intent:

Table 3: NLU Intent Data

Intent	Description	Example
greet	Used when a user wants to start a conversation. At the beginning of a conversation, it is expected that it will begin with a text that has greeting intent.	Hey
goodbye	Used when a user wants to end a conversation. Whenever a user enters a text with intention of ending a conversation, Health Agent will use this intent and decide the next action.	I'm done
healthy_patient	Used when a user wants to inform the bot that he is healthy. As a user enters a text that informs the Health Agent that the user is healthy, this intent is identified and Health Agent ends the conversation with an utterance that motivates the user to stay fit.	I am perfectly fine
unhealthy_patient	Used when a user wants to inform the Health Agent that the user is unhealthy. Once the user enters the text saying that the user is not well, Health Agent will start gathering information about the user's ill health.	I am sick
inform_symptoms	Used when a user wants to inform symptoms. This intent is used to extract symptoms from the entered text.	Seems like I have a headache
bot_challenge	Used when a user wants to know about the agent. This intent is used when a user challenges the Health Agent about the agent's identity.	are you real?
out_of_scope	Used when a user tries to ask questions that are not within the scope of the Health Agent.	Can you make a pizza?

The NLU data is also used to train the model for entity extraction. Below is an example pattern that Rasa use to identify predefined entity types from the input sentence.

- I am observing **[chest pain] (DISEASE)**

As described by Kong et al. (2021), entities in the training dataset are annotated and written as the Markdown URL expression, namely **[entity value] (entity type)**.

With the help of intent data, NLU understands how to classify each text entered by the user and drives the conversation further.

7.2.2 Entities

Entities are used to extract specific information from the text entered by the user. The Health Agent has a **DISEASE** entity for extracting symptoms from the user's input text. The system has limited control over how a user can enter the symptoms they are observing. Training the NLU model by giving an exhaustive list of symptoms in the healthcare domain is a resource and time-intensive task. As explained in the transfer learning section, a pre-trained model can be used for such tasks. Neumann et al. (2019) have explained employing the spacy model for biomedical natural language processing. To benefit from this transfer learning approach, the NLP pipeline is equipped with Spacy's en_ner_b5cdr_md model for the named entity recognition task. The model is capable of extracting disease and chemical entities from a text however Health Agent makes use of only disease entities.

7.3 Response Data Generation

NLG is responsible for the generation of responses and a response is generated based on the user's input. Health Agent makes use of template-based response generation to have better control over what kind of response the system can generate.

Table 4 below summarizes the responses that the bot can generate and the purpose of each response:

Table 4: NLG Response Data

Response Template	Description
utter_greet	Whenever a user greets the agent, the system generates a greeting response as defined by this intent.
utter_healthy_bye	Whenever a user enters a text intending that the user is feeling healthy, the agent generates a response using this template.
utter_unhealth_bye	Whenever Health Agent intend to end a conversation with an unhealthy patient, the system will generate a response based on this template.
utter_goodbye	Whenever a user wants to end the conversation abruptly, the system will generate a response based on this template.
utter_ask_disease	Once the Health Agent identifies that the user is unhealthy, it will use this template for generating a response to ask the user the symptoms user is observing.
utter_repeat_information	After collecting all the information from a user, the system will use this template to display the gathered information to the user as an acknowledgement.

utter_iamabot	Whenever a user asks about the system or challenges the system's identity, this template is used to generate a response and provide Health Agent's introduction.
utter_out_of_scope	A user might be tempted to play with the agent and ask random questions. Health Agent is a task-oriented conversational agent and cannot serve random requests. This template is used to politely inform the user about the system's inability to handle random requests.
utter_please_rephrase	The text entered by a user might be ambiguous and the NLU model may assign multiple intents with similar confidence to more than one intent. In such scenarios, this template is used to ask the user to rephrase the inputs.
utter_submit	Once the Health Agent has finished collecting the data, the Health Agent keeps the user informed about the progress with text generated from this template.
action_predict_disease	This is a response generated using custom actions and it is used to display matched disease information to the user.

Following is an example of the utter_howareyou response template. As it has two texts, Rasa NLG picks one during the execution and thus can add variety to its responses.

```
utter_howareyou:
  - text: "How are you doing today?"
  - text: "How are you feeling today?"
```

7.4 Dialogue Data Generation

Dialogue management data consists of stories and rules for understanding the conversation flow between the user and the bot.

7.4.1 Stories

The Health Agent is designed to handle two major types of conversational flows. One flow handles the conversation between a healthy patient and the agent while the other handles the flow between an unhealthy patient and the agent. Both flows include the use of different intents and actions and forms(optionally). Table 5 below summarizes the flexible conversation flows that the bot can handle:

Table 5: Story Data for DM

Story Name	Description
Health patient	This story represents a Health Agent's conversation with a healthy user and does not ask the user for symptoms and in the end, motivates the user to stay fit.
Unhealthy patient informing symptoms	This story represents a Health Agent's conversation with an unhealthy user. It makes use of the Rasa forms for gathering symptoms and handling out-of-context information during the information gathering.

7.4.2 Rules

Rasa rules help in handling strict one-turn conversations. Below mentioned rules are added to Health Agent's DM model so that whenever a user seeks information that is covered by the rule, they get a reply even when the context of the conversation is different from the current flow.

Table 6 summarizes the strict conversation flows the bot can handle.

Table 6: Rule Data for DM

Rule Name	Conversational Flow
Utter goodbye anytime the user says goodbye	A user can say bye anytime during the conversation indicating the user wants to end the conversation. This rule handles such scenarios.
Utter bot information anytime the user asks about the bot	A user may wish to ask for information about the Health Agent anytime during the conversation and this rule will handle the case.
Handle out-of-scope intents	A user may get tempted to play with the agent and ask the information not intended to handle by the Health Agent. The Health Agent informs the user about its shortcomings using this rule.
Ask the user to rephrase the ambiguous text	The NLU may assign similar confidence to more than one intent to the text entered by the user and hence identify the text as ambiguous. In such cases, a fallback intent is triggered, and this rule helps in handling such scenarios.
Activate gather_symptoms	This rule triggers the activation of a Rasa form for gathering symptoms during a conversation with an unhealthy patient.
Submit gather_symptoms	This rule triggers the submission of a Rasa form for gathering symptoms during a conversation with an unhealthy patient.
Add symptom to symptoms list slot	This rule triggers the symptoms collection custom action when the user informs symptoms to the Health Agent.

7.4.3 Slots

The system has two slots which work as temporary long-term memory i.e., the memory that can hold information across the conversation.

Table 7: Slot Data for DM

Slots	Function
disease	Store a symptom (if any) informed in the last input text
symptoms	Store all the symptoms informed

7.4.4 Forms

Dialogue management also has a Rasa form called **gather_symptoms** for the systematic gathering of symptom entities. The form is activated and submitted with combined efforts of rules and stories that include the form. The form also helps in handling out-of-context questions while still maintaining the context of the conversation. Because of this feature, if a user asks some random question instead of informing symptoms (during an unhealthy user conversation flow), the Health Agent answers the random question and resumes seeking symptom information from the user.

7.5 Policies

The Health Agent makes use of three policies for dialogue management and next action prediction. This section describes these policies.

7.5.1 MemoizationPolicy

This policy remembers stories from the training data. If the current conversation matches the stories of training data, it predicts the next action based on the stories.

7.5.2 RulePolicy

This policy handles the fixed behavioural scenarios in the conversation. It makes a prediction based on the rules defined in the training data.

7.5.3 TEDPolicy

The Transformer Embedding Dialogue (TED) Policy is a multi-task architecture for next-action prediction. Its architecture consists of several transformer encoders. Several hyperparameters are available for fine-tuning the model.

7.6 Training the Model

In NLU, incoming messages are processed by a sequence of components and enriched to a machine-understandable format. As per Rasa documentation (Rasa Technologies Inc., 2022), training the NLU model is controlled by components defined in the configuration file. The model can be finetuned by configuring parameters in the configuration file. The Botfront (Botfront, n.d.) has discussed some of the pipeline optimization techniques for better intent and entity extraction.

7.6.1 NER Language Model

The Health Agent has used the ScispaCy Language model en_ner_bc5cdr_md (Neumann et al., 2019) which is a fined finer-grained NER model trained on BC5CDR dataset for diseases and chemicals. It loads pre-trained models with pre-trained word vectors in the NLU pipeline. All the other spacy components used in the pipeline relied on the Spacy structures. The model is configured to be case-insensitive.

7.6.2 Tokenizer

Tokenizers split received text into tokens. The Health Agent supports the English language for interacting with a user. Hence a tokenizer that uses English language features would be beneficial for token generation. The Spacy language model supports the English language, and the pipeline is configured with SpacyTokenizer for feature compatibility.

7.6.3 Featurizer

The NLU pipeline is configured with SpacyFeaturizer, RegexFeaturizer, and LanguageModelFeaturizer.

The disease entity extraction is performed using the Spacy NER model it is evident that the SpacyFeaturizer is required for Spacy NER to perform well.

LanguageModelFeaturizer is based on the HuggingFace pre-trained model LaBSE developed by Rasa. It creates features for entity extraction, intent classification, and response selection using the configured pre-train model. Tokenizer components should appear before featureziers (Rasa Technologies Inc., 2022).

7.6.4 Entity Extractor

The pipeline makes use of SpacyEntityExtractor. Although the spacy language model *en_ner_bc5cdr_md* can perform NER for disease and chemical entities, the system only makes use of the DISEASE dimension of the model. This use of a pre-trained model and transfer learning significantly reduced memory and computation requirements to train a model for disease NER tasks.

Dual Intent Entity Transformer (DIET) based DIETClassifier is also capable of extracting the entities however, Health Agent is focusing on the Spacy model for NER.

7.6.5 Intent Classifier

The system uses DIETClassifier for intent identification and classification purpose. DIET makes use of pre-trained word embedding made available through LanguageModelFeaturizer. DIET exposes many hyperparameters available to tune the model. As discussed by Botfront (Botfront, n.d.), some of the key parameters have been set by experimenting with the model configuration.

A FallbackClassifier is configured to handle cases where the NLU intent classification score becomes ambiguous. The threshold parameter of the classifier ensures that if no intent is classified with a confidence level greater than the threshold, a fallback intent is triggered. The ambiguity threshold parameter is used to maintain enough confidence score difference between the highest-ranked intents.

As explained earlier, NLU performs intent classification and entity extraction tasks. The following example shows the output of NLU when the text '*I have headache'* is parsed by the NLU. This is a JSON output and shows entities extracted and intent classified. Note that each intent has confidence attached to it with inform_symptoms having the highest confidence. The Spacy entity extractor does not include confidence in its output.

Example NLU Output:


```
{
  "text": "i have headache",
  "intent": {
    "name": "inform_symptoms",
    "confidence": 0.9997956156730652  },
```

```
  "entities": [
    {
      "entity": "DISEASE",
      "value": "headache",
      "start": 7,
      "confidence": null,
      "end": 15,
      "extractor": "SpacyEntityExtractor"    },
    {
      "entity": "DISEASE",
      "start": 7,
      "end": 15,
      "confidence_entity": 0.9037267565727234,
      "value": "headache",
      "extractor": "DIETClassifier"    }
  ],
  "text_tokens": [
    [
      0,
      1    ],
    [
      2,
      6    ],
    [
      7,
      15    ]
  ],
  "intent_ranking": [
    {
      "name": "inform_symptoms",
      "confidence": 0.9997956156730652    },
    {
      "name": "unhealthy_patient",
      "confidence": 9.907962521538138e-05    },
    {
      "name": "greet",
      "confidence": 4.6685065171914175e-05    },
    {
      "name": "goodbye",
      "confidence": 2.3874183170846663e-05    },
    {
      "name": "healthy_patient",
      "confidence": 1.7824922906584106e-05    },
    {
      "name": "bot_challenge",
      "confidence": 1.329962287854869e-05    },
    {
      "name": "out_of_scope",
      "confidence": 3.6354183521325467e-06    }
  ]
}
```


7.7 Action Server Implementation

As documented by Rasa (Rasa Technologies Inc., 2022) the action server acts as a backend system of the Health Agent and executes custom actions and any other business logic.

7.7.1 Custom Actions

The Health Agent is using the action server for the following two reasons:

- To collect the symptoms entered by the user
  A Rasa custom action is used to extract all the DISEASE entities identified by the NLU and add them to the ‘symptoms’ slot
- To call the backend service for identifying the disease matching the collected symptoms

This Rasa custom action sends the collected symptoms to a disease-matching service and renders the response of the service to the user.

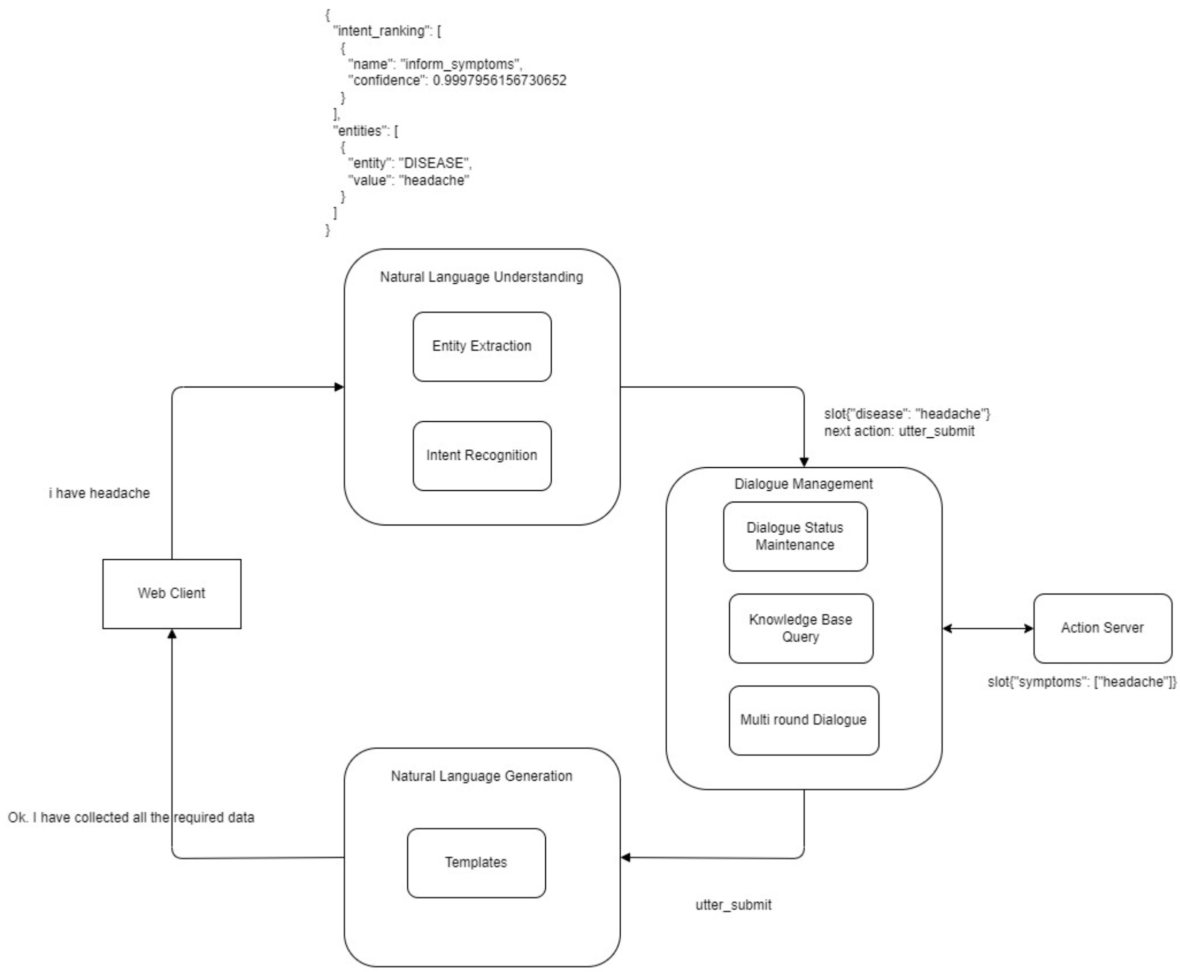
7.7.2 Disease and Symptom Matcher

Once the symptoms have been extracted by the NLU, DM makes a call to the action server for identifying the disease that matches the extracted symptoms. Disease and Symptom Matcher is a backend service and is a part of the action server. It received a list of extracted symptoms and matches it with the symptom list of diseases present in the database. It rates each matched disease based on the number of matched symptoms. The output of the service is a list of diseases that has the highest score.

7.8 Conversation Data Exchange

This section briefly explains data exchange between the Health agent and a user. Figure 10 shows data flow during one dialogue turn of communication between the agent and a user. When a user enters 'I have headache' in the provided web interface to communicate with the Health Agent, NLU extracts intent and entities from this message. This information is passed to the DM unit which uses extracted entities for slot filling using Rasa core as well as Rasa custom actions. The DM then predicts utter_submit as the next action and NLG usage template for the utter_submit response to generate response text 'Ok. I have collected all the required data and the text is displayed to the user using the web interface.

Figure 10: Conversation Data Exchange



Note: Image adapted from Khilji et al. (2020)

7.9 Data Pre-processing

The disease data obtained from Kaggle (M, 2020) required pre-processing so that it can be used as a backend database for symptoms and disease matching.

The original dataset contained the following columns:

- name
- link
- symptoms
- causes
- risk_factor
- overview
- treatment
- medication
- home_remedies

Of these, 'name' and 'symptoms' were considered for further processing. The symptoms column required text processing to remove non-alphanumeric characters and punctuation removal. An additional column for the symptom list was created by extracting symptoms from the 'symptoms' column of the dataset. The Spacy en_ner_bc5cdr_md mode was employed for the disease NER (Neumann et al., 2019) from the symptoms.

7.10 Web Client

This section discusses the interface component of the Health Agent. A well-designed interface should be considered for a providing superior experience to its users. To facilitate easy interaction between users and Health Agent, a chat widget made available by Dialogue Technologies Inc. (2021) was integrated into an HTML page. Python's built-in HTTP server is used for hosting the HTML page

The following figures are screenshots of an example interaction of a user with the Health Agent. Figure 11 shows the interaction of a healthy user with the agent.

Figure 11: Health Agent's Conversation with a Healthy User

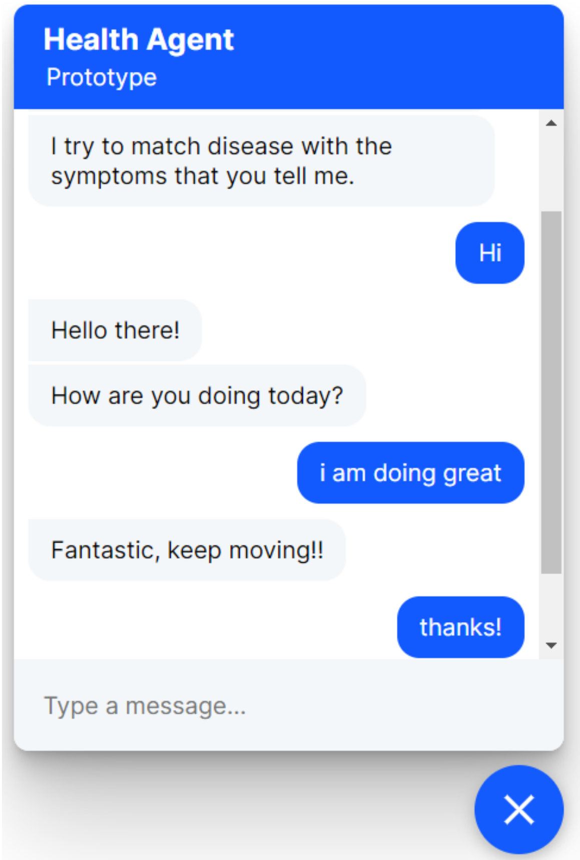
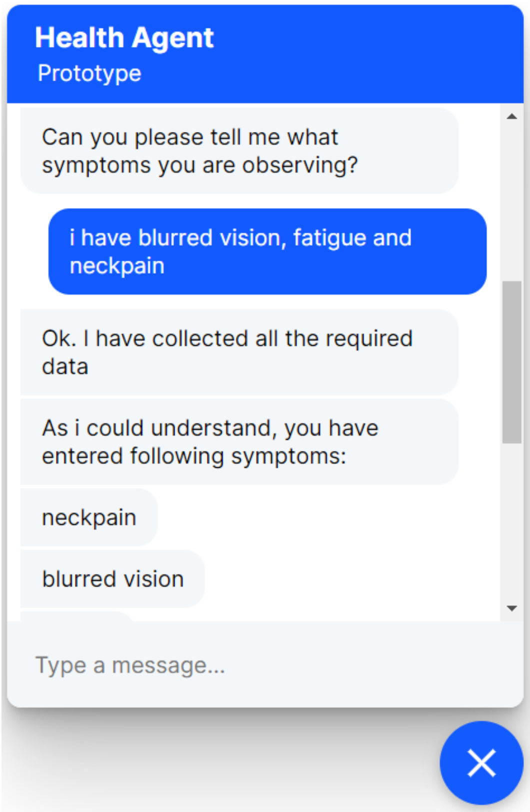


Figure 12 shows part of the conversation between the Health Agent and an unhealthy user. The Health Agent asks the user about symptoms they are observing two and the user responds with symptoms. The agent then repeats the extracted symptoms.

Figure 12: Health Agent's Conversation with an Unhealthy User



7.11 Software Libraries

The core of the Health Agent is implemented in python language with the help of the following libraries. The interface is implemented in JavaScript.

7.11.1 Python

The Health Agent is developed on the python interpreter version 3.8 provided by Anaconda Distribution.

7.11.2 Rasa Open Source

The health agent is developed using the open-source conversational AI framework developed by Rasa using Rasa version 3.2.1. The open-source nature provides complete control over building a conversational agent. As described by Rasa Technologies Inc. (2022), Rasa Open Source supplies the building blocks for creating virtual assistants. Rasa Open Source provides the following three main functions:

7.11.2.1 Natural Language Understanding

NLU turns the text entered by the user into text that can be understood by the Rasa pipeline for intent and entity extraction. NLU is a category of machine learning that analyses freeform text and turns it into structured data. (Rasa Technologies Inc., 2022)

Training NLU model needs NLU data which consists of representative sample data for identifying intents and entities. NLU components are configured in the NLU pipeline and are explained in further sections.

7.11.2.2 Dialogue Management

The dialogue management model drives the conversation flow based on user inputs and model configuration. Training DM required dialogue data consisting of the stories, rules, slots and policies. The training data is discussed in further sections.

7.11.2.3 Integrations

A website or app integration is an important part to deliver the ultimate user experience. Rasa provides REST endpoints called connectors to connect the backend with custom interface channels.

7.11.3 Transformer

The transformer model made available by Hugging Face (2022) was used for word embedding.

7.11.4 Spacy

The spacy library called SciSpacy is used for disease entity extraction (Neumann et al., 2019).

7.11.5 Pandas

The pandas library is used for pre-processing the dataset and generating the CSV database to be used by Health Agent for disease matching.

7.11.6 PyCharm

PyCharm IDE was used for all the python-based development.

7.12 Code Structure

The Health Agent is built by adapting the development practices suggested by Rasa. The code structure of the health agent is included in Appendix F.

8 Evaluation

For the NLU and DM models, the two important tasks are intent classification and entity extraction. As explained by Géron (2019), the main performance metrics for classification tasks are based on the following measures:

- True Positive (TP): model correctly predicts the positive class
- True Negative (TN): model correctly predicts the negative class
- False Positive (FP): model incorrectly predicts positive class
- False Negative (FN): model incorrectly predicts negative class

$$precision = \frac{TP}{TP+FP}$$ equation (1)

$$recall = \frac{TP}{TP+FN}$$ equation (2)

$$F_1 = 2 \times \frac{precision \times recall}{precision+recall}$$ equation (3)

$$Accuracy = \frac{TP+TN}{TP+TF+FP+FN}$$ equation (4)

Precision is the proportion of positive predicted class that is actually correct while recall is the proportion of positive classes that were identified correctly. To completely assess a model's efficacy, you must consider both precision and recall. Unfortunately, accuracy and recall are often at odds. Intent extraction is a multi-class classification.

8.1 Experimental Setup

Table 8 details the configuration of the system that was used for conducting the evaluation experiments.

Table 8: Experimental Setup

Specification	**Details**
System Type	Laptop Running Windows 10
Processor and Cores	Intel(R) Core (TM) i5-8265U CPU @ 1.60GHz 1.80 GHz
Memory (RAM)	20.0 GB
Operating System	Windows 10 Home Single Language (Version 21H2)
Python Version	Python 3.8
Python Environment Manager	Conda Environment
Development IDE	PyCharm 2022.1.3 (Professional Edition)

8.2 Data Validation

There should not be inconsistencies in the domain, NLU data, story data or rule data. Rasa data validation utility ensures there are no mistakes between domain, NLU and story data. Rasa CLI provides a utility to validate the training data. Figure 13 shows the rasa data validation output.

Figure 13: Rasa Data Validation

```
2022-08-16 05:01:42 INFO     rasa.validator  - Validating intents...
2022-08-16 05:01:42 INFO     rasa.validator  - Validating uniqueness of intents and stories...
2022-08-16 05:01:42 INFO     rasa.validator  - Validating utterances...
C:\Users\developer\anaconda3\envs\HealthAgent\lib\site-packages\rasa\shared\utils\io.py:98: UserWarning: The utterance 'utter_ask_disease' is not used in any story or rule.
2022-08-16 05:01:42 INFO     rasa.validator  - Story structure validation...
Processed story blocks: 100%|██████████| 2/2 [00:00<00:00, 128.39it/s, # trackers=1]
2022-08-16 05:01:42 INFO     rasa.core.training.story_conflict  - Considering all preceding turns for conflict analysis.
2022-08-16 05:01:42 INFO     rasa.validator  - No story structure conflicts found.
```

If there is any inconsistency between rules and stories, it gets validated in the training data and training gets aborted. No data issues were reported during the training.

Note: The warning in the figure can be ignored as the utterance mentioned in the warning was invoked from custom actions which reside outside the rasa pipeline.

8.3 Training Model

NLU model was performed using a shuffle and split data strategy. The NLU training data was split with an 80/20 split of train/test data. The following commands splits NLU data into test and train datasets.

```
rasa data split nlu --random-seed 43
```

The NLU and DM models were trained as follows:

```
rasa train
```

The training results in generating model artefacts in the directory named models in the *.tar.gz* format.

8.4 Evaluating NLU Model

It is important to validate how the model is performing on the unseen data. The trained model was tested using the holdout dataset as follows:

```
rasa test nlu --nlu train_test_split/test_data.yml
```

Table 9 represents intent classification metrics. The mode confused to classify goodbye intent with healthy_patient intent and greet intent for 3 instances.

Table 9: Intent Classification Metrics by Intent

Intent	Train/Test Data Point Count	precision	recall	f1-score	Confused with	Confusion Count
greet	18/5	1	0.8	0.889	goodbye	1
goodbye	46/11	0.846	1	0.917		
inform_symptoms	31/8	1	1	1		
out_of_scope	24/6	0.833	0.833	0.833	goodbye	1
unhealthy_patient	14/4	1	0.75	0.85	healthy_patient	1
bot_challenge	18/4	1	0.75	0.857	out_of_scope	1
healthy_patient	20/5	0.833	1	0.909		

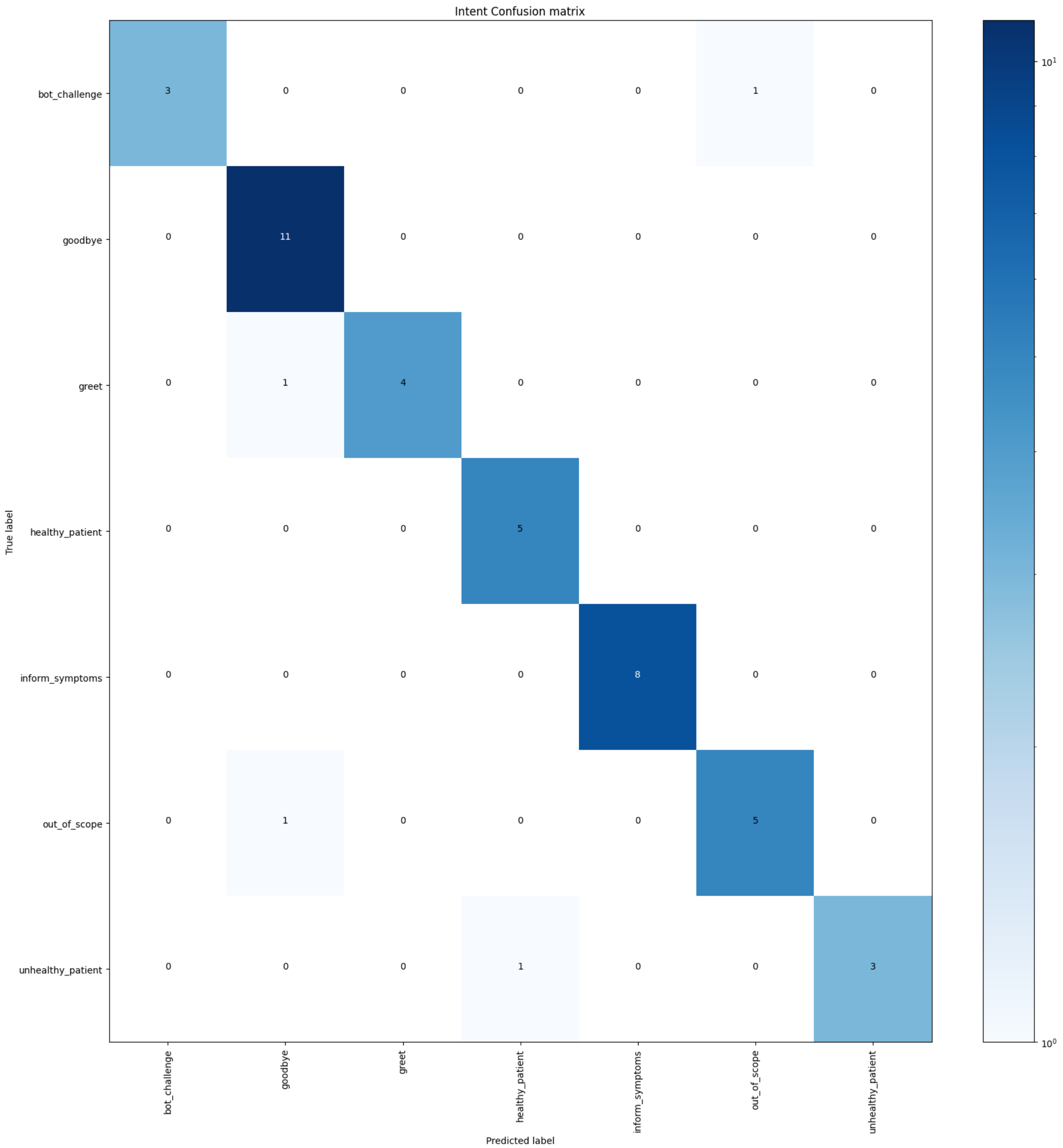
Table 10 shows the overall intent classification metric of the NLU model.

Table 10: Intent classification Overall Metric

Metric	Score (weighted avg)
F1	0.905
Precision	0.918
Recall	0.907
Accuracy	0.907

The intent classification confusion matrix in figure 14 shows which intents were mistaken for others.

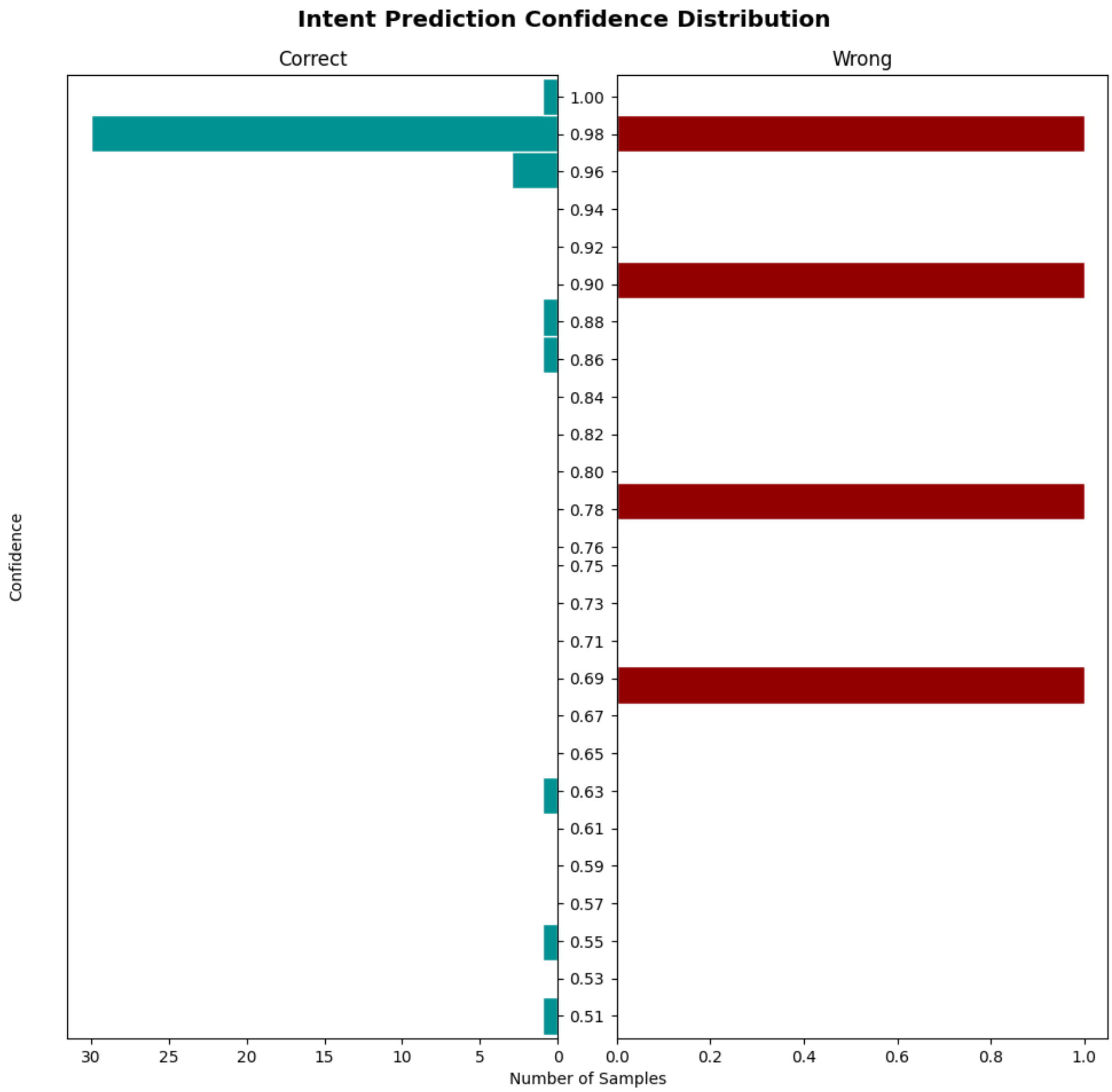
Figure 14: Intent Confusion Matrix on Test Data



The histogram in figure 15 shows confidence in all the intent predictions with correct and incorrect predictions being displayed in blue and red bars.

Note: Please note the scale of the number of samples for Correct and Wrong predictions. They are not the same and hence the careful reading of the histogram is required.

Figure 15: Intent Prediction Confidence Distribution on Test Data



8.5 Evaluating Dialogue Model

Dialogue management consists of stories, rules and policies governing the next predicted action. The model is evaluated based on how well it could predict the next action.

Table 11: Next Action Prediction Metric by Action

Action	precision	recall	f1-score
gather_symptoms	1	1	1
action_listen	1	1	1
utter_greet	1	1	1
utter_submit	1	1	1
action_repeat_information	1	1	1
action_predict_disease	1	1	1
utter_howareyou	1	1	1
utter_healthy_bye	1	1	1

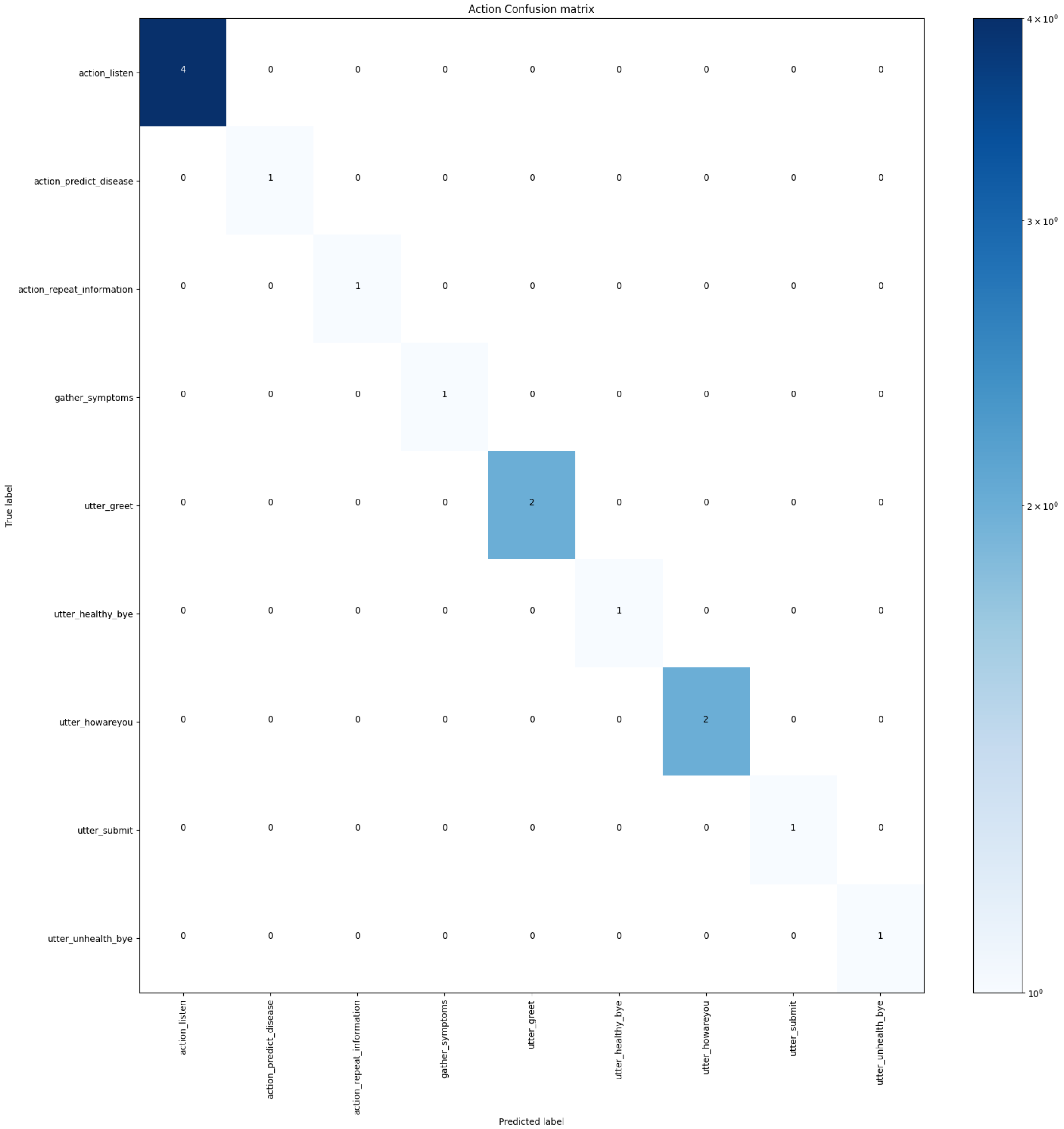
Table 12: Action Prediction Overall Metric

Metric	Score (weighted avg)
F1	1
Precision	1
Recall	1
Accuracy	1

As the Health Agent have well-defined actions, the Rasa prediction policy performed well giving the best results.

Figure 16 shows the action prediction confusion metrics on test data.

Figure 16: Action Confusion Matrix



8.6 Conversation Flow Tests

Rasa offers a test framework that can be used for constructing test stories. These test stories can then be used to test the trained model and gain confidence in the trained model's performance in a variety of scenarios. The following figure and metrics show how the Health Agent is performing against the test stories developed for assessing the agent.

Figure 17 shows the intent confusion metrics of the test stories evaluation.

Figure 17: Intent Confusion Matrix on Test Stories

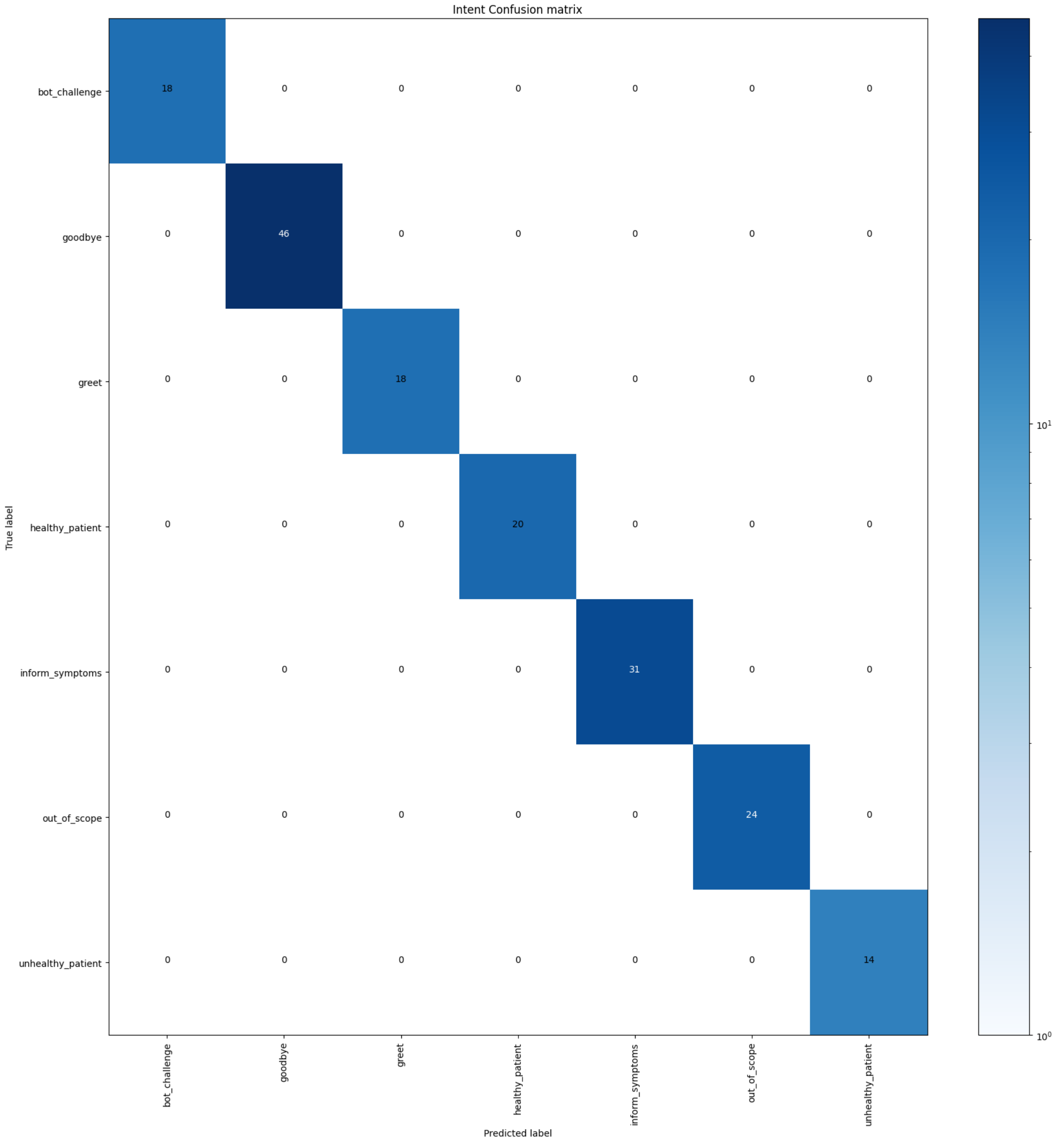


Table 13 shows that all the test stories were successfully tested with the correct intent being identified

Table 13: Intent Classification Metric

Metric	Score (weighted avg)
F1	1
Precision	1
Recall	1
Accuracy	1

Figure 17 shows that all the intents from the test stories were correctly predicted with an overall good confidence level

Figure 18: Intent Prediction Confidence of Test Stories

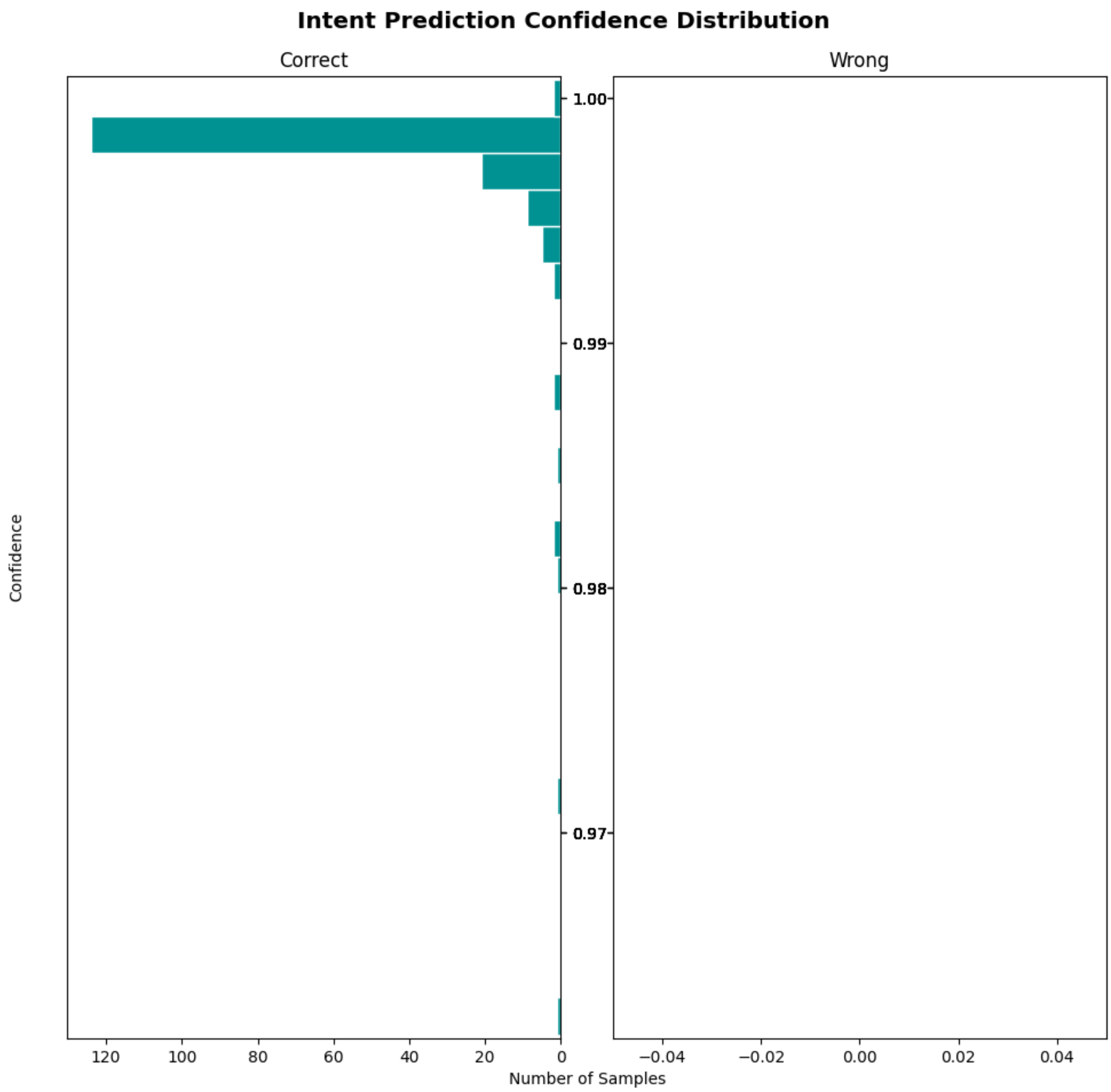
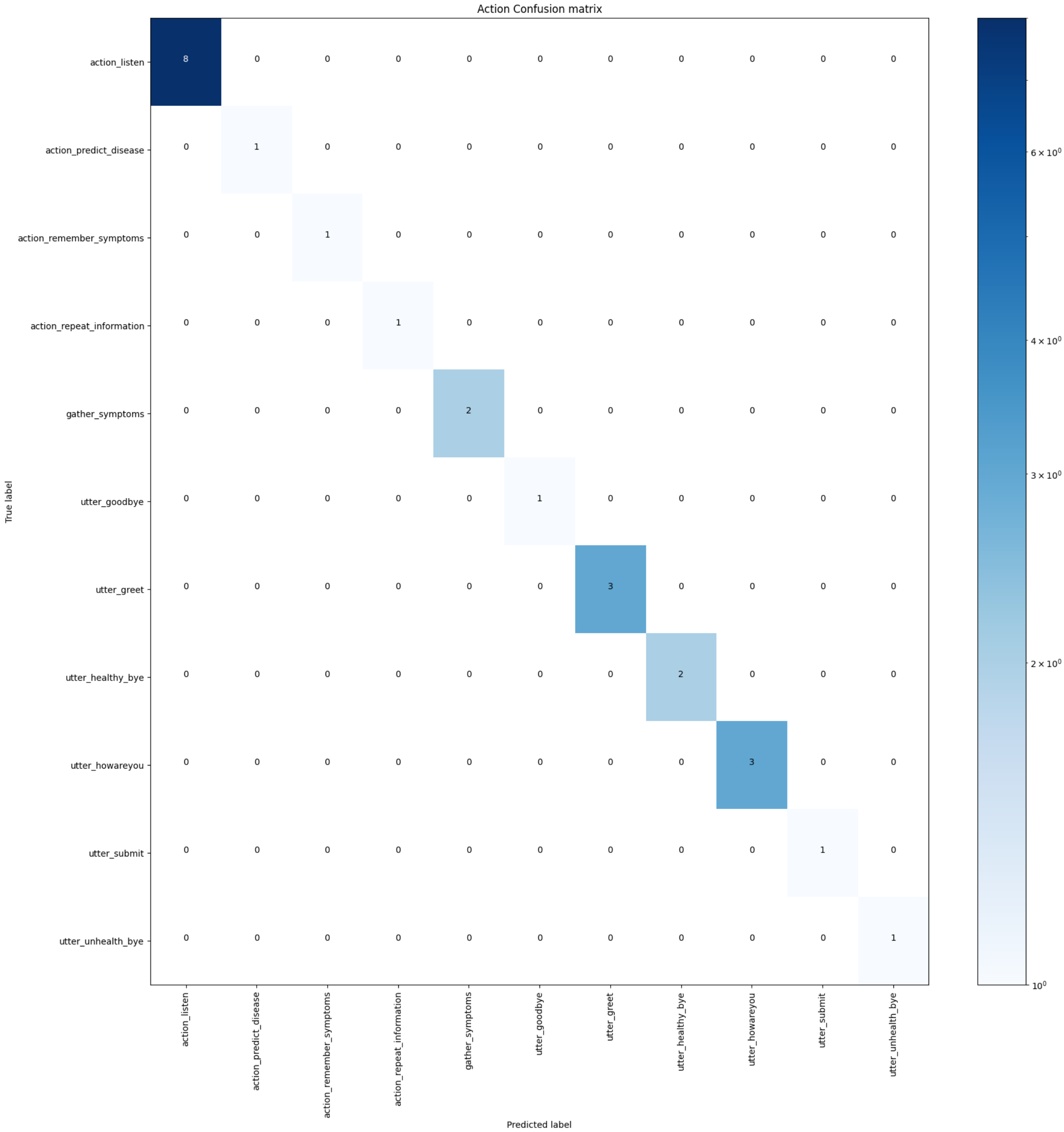


Figure 19 shows that the DM predicted the next actions for all the test stories correctly.

Figure 19: Action Confusion Matrix



8.7 Interactive Mode Validations

It was important to make sure that Rasa forms and slot filling in the dialogue management system are working as expected. Rasa's interactive mode execution helped in getting insights into the form and slot filling and troubleshooting. The validations are attached in Appendix C.

8.8 Survey Responses

A survey was conducted to get users' feedback about the naturalness of the Health Agent's conversation and the functionality of the system. The survey was hosted on the Qualtrics survey system provided by the University OF St Andrews and the survey link was shared with the respondents. The screenshots of two representative conversation flows were provided for the respondent's feedback. The survey was anonymous and voluntary and no personal information was collected during the survey. The content of the survey is provided in Appendix G. For the user's convenience, the Health Agent is referred to as a chatbot in the survey.

8.8.1 Survey Design

Each of the questions can be answered with one of the six choices:

Strongly Agree, Agree, Neutral, Disagree, Strongly Disagree, Prefer Not to Answer

Questions about the naturalness of the conversation with the chatbot:

1. The conversation with the chatbot sounded natural and smooth.
2. The chatbot could handle out-of-context interactions gracefully.
3. The chatbot kept the user informed about the progress and the conversation was engaging.
4. The chatbot politely acknowledged its shortcomings.
5. The conversation ended gracefully and sounded natural.

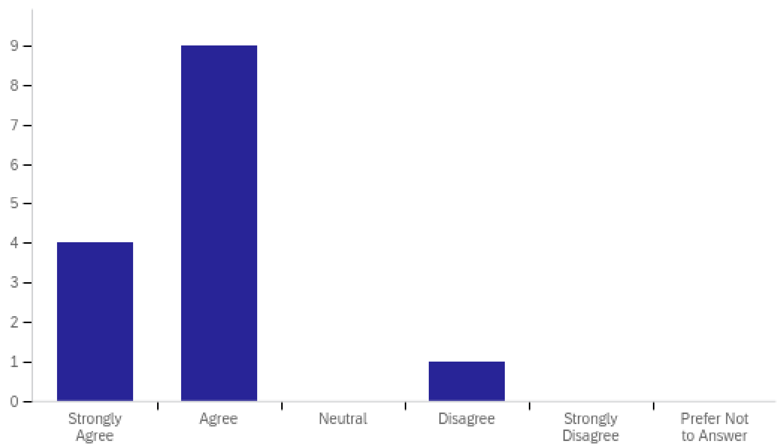
Questions about the functionality of the chatbot:

1. When asked, the chatbot introduced itself and the introduction was clear.
2. The chatbot clearly understood the user's intentions (healthy/unhealthy user, informing symptoms, asking about the chatbot).
3. The chatbot sought appropriate information from the user.
4. The chatbot recognized the symptoms entered by the user.
5. The chatbot appropriately responded to the user's question.
6. The displayed disease names were matching with the entered symptoms.
7. The disease names were displayed appropriately.

8.8.2 Survey Responses

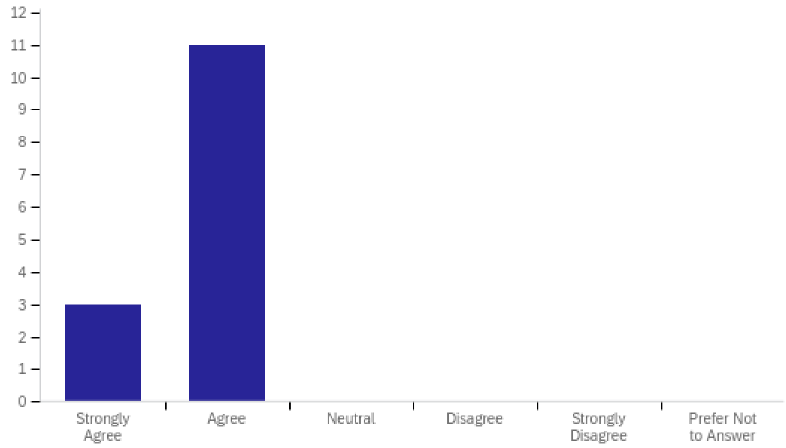
A total of 14 responses were received for the survey. The following figures provide insights into the user response.

Figure 20: The conversation with the chatbot sounded natural and smooth



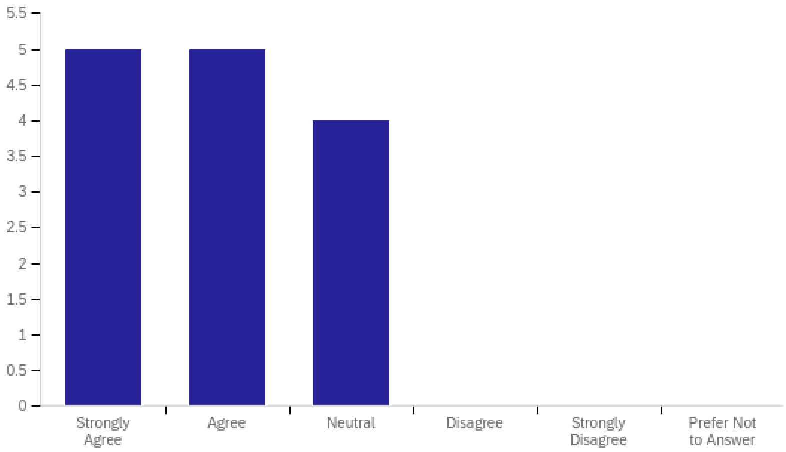
Observation: Respondent found chatbot conversation smooth and natural.

Figure 21: The chatbot could handle out-of-context interactions gracefully



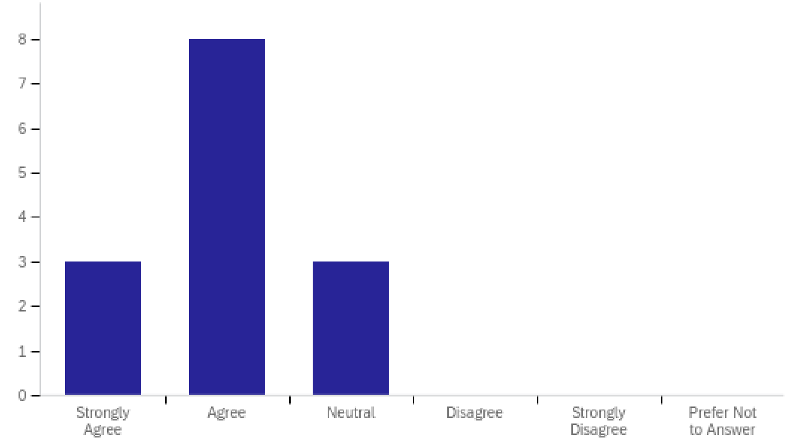
Observation: Respondent liked the bot's out-of-context handling

Figure 22: The chatbot kept the user informed about the progress and the conversation was engaging



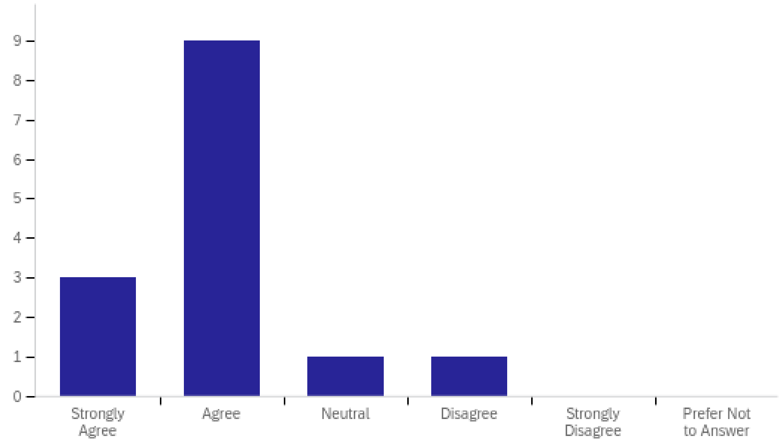
Observation: Respondent could partially agree that the chatbot conversation was engaging.

Figure 23: The chatbot politely acknowledged its shortcomings



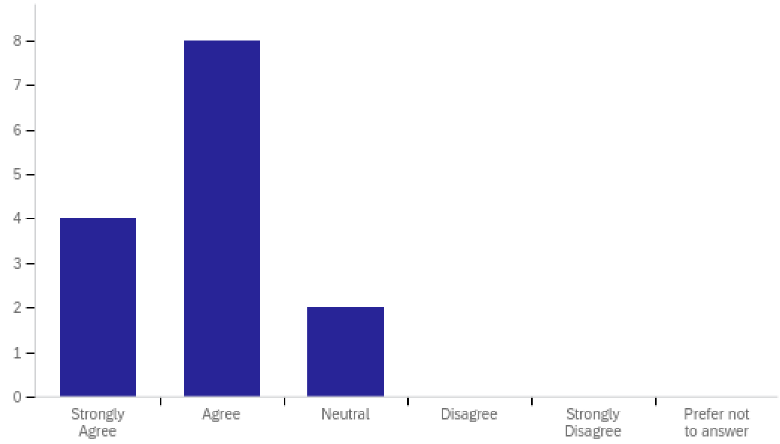
Observation: Respondents agreed that the chatbot acknowledged its shortcomings.

Figure 24: The conversation ended gracefully and sounded natural



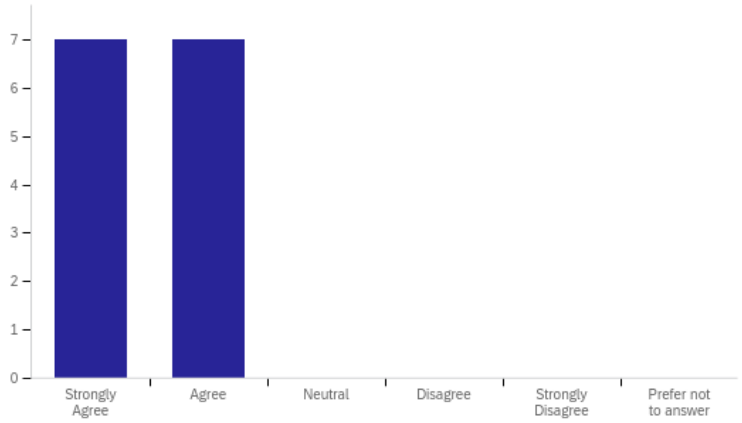
Observation: Respondents found that the conversations ended gracefully.

Figure 25: When asked, the chatbot introduced itself and the introduction was clear



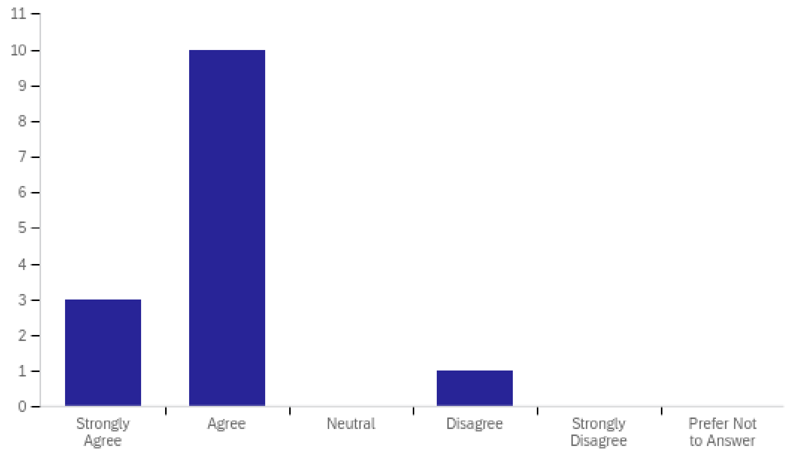
Observation: Respondents found that the chatbot's introduction was clear.

Figure 26: The chatbot clearly understood the user's intentions



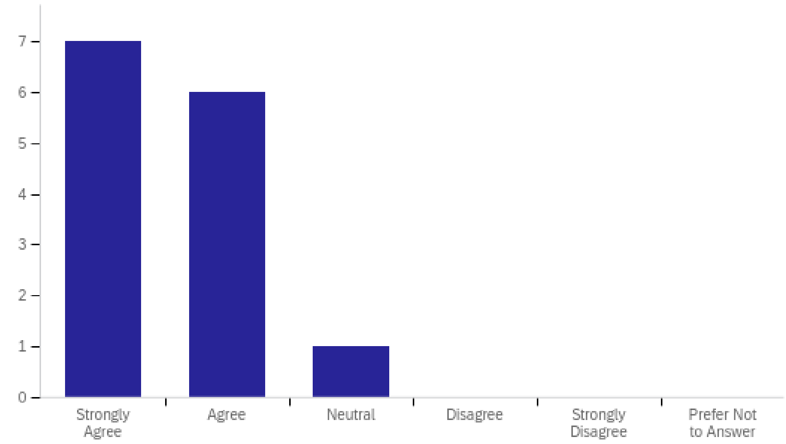
Observation: The respondent agreed that the chatbot clearly understood the user's intentions.

Figure 27: The chatbot sought appropriate information from the user



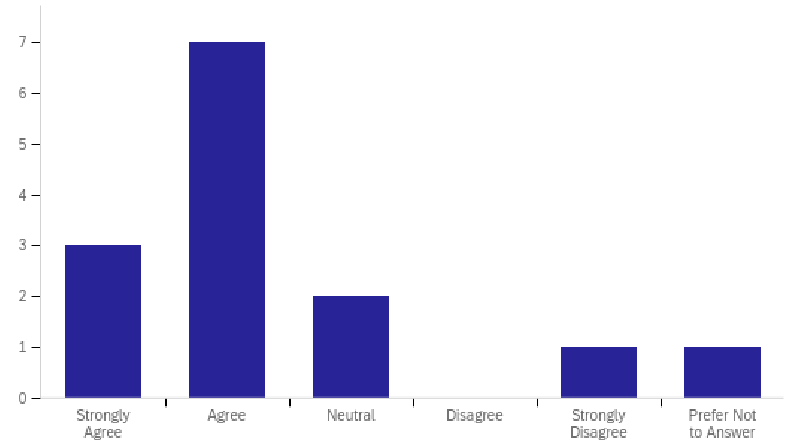
Observation: Respondents agreed that the chatbot sought appropriate information from the user.

Figure 28: The chatbot recognized the symptoms entered by the user



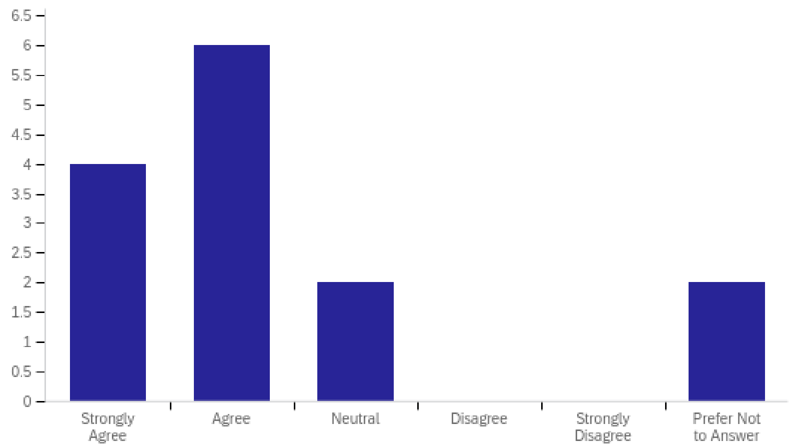
Observation: Respondents agreed that the chatbot could recognize symptoms from the text entered by the user.

Figure 29: The chatbot appropriately responded to the user’s question



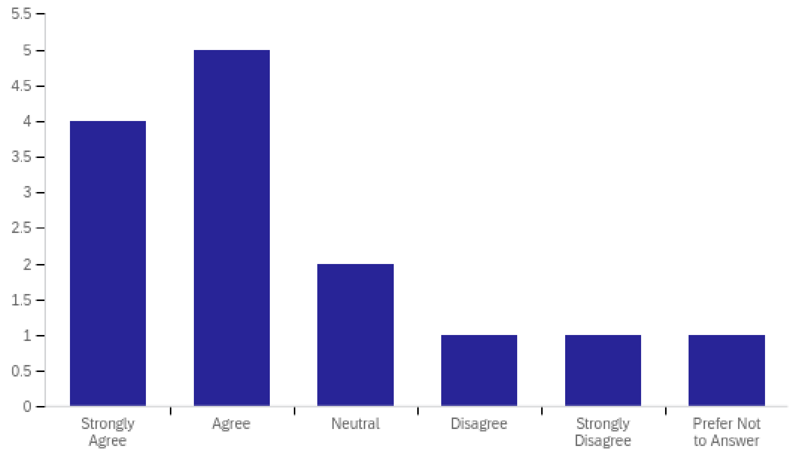
Observation: Respondents found the chatbot’s response to the user's question was appropriate.

Figure 30: The displayed disease names were matching with the entered symptoms



Observation: Respondents found the predicted disease was matching with entered symptoms.

Figure 31: The disease names were displayed appropriately



Observation: Respondents could not agree on the disease name display format.

Response to question: What features can be improved?

Below are the notable improvements as suggested by respondents:

- Knowing the user better by profiling the user's pre-existing health condition and lifestyle habits.
- Supporting the predicted disease with the confidence level of the prediction
- Integrating chatbots with government systems like NHS so that health agencies can be alerted in case of emergencies.
- Improving text in the bot's response
- Providing disease treatment information
- Asking more questions to collect more information about the symptoms

Response to question: What feature of the system is most interesting?

Below are the notable responses for the interesting feature of the system:

- Symptom extraction

- Handling of out-of-context questions
- Disease matching feature
- Detecting the health conditions of the user
- Smooth and natural responses

Response to question: Do you have any other feedback or comment?

Below are the notable responses:

- The chatbot can reduce the workload on the health professionals
- The chatbot could be integrated with the NHS system using the patient's NHS number
- It is dangerous to create systems which make definite pronouncements without either making clear either the reasoning process or even a reason that the user should trust the system

9 Conclusions and Future Work

9.1 Conclusion

The primary focus of this paper was to develop a prototype of goal-oriented conversational AI assistance for Healthcare advice. Users were able to interact with the agent via a web page integrated chat widget. During the multi-turn interactions, the Health Agent is capable of differentiating healthy and unhealthy users based on the text entered by the users. While interacting with a healthy user, the Health Agent do not ask about symptoms. When the agent realises that the user it is interacting with is unhealthy, it asks for the symptoms observed by the user. During this dialogue, Health Agent can handle questions which are not in the context of symptom collection, answer those questions and resume the symptom-gathering task. Once the symptoms are collected, the agent keeps the user updated with the progress the agent is making and finally presents the disease name(s) matched with the illnesses. The agent maintains the conversation context and with its natural responses, the conversation with the agent feels natural.

The Health Agent is developed using all open-source libraries. The use of Rasa open source provided all the control over the data and privacy aspects of the conversational agent. Health Agent achieved success in entity extraction for disease symptoms with the use of pre-trained Spacy language models with minimum fine-tuning to the original model. Without the use of the pre-trained model, it would be computationally expensive to train the model on disease and symptom data. The model can be further improved with extensive training and tuning. The use of a transformer for intent classification was successful as pre-trained embeddings greatly enhanced the contextual text-matching power of the model. It was observed that even with less conversational data, the agent could perform reasonably well in handling unseen text and classifying them correctly.

9.2 Suggestions for Future Work

This section addresses some of the potential enhancements to the current work done during the research.

Rasa model can be better fine-tuned using hyperparameter optimization techniques like Hyperopt - Distributed Asynchronous Hyper-Parameter Optimization. A variety of policy configurations can be compared by training models for different configurations and the best-performing model can be chosen for the final implementation of the agent.

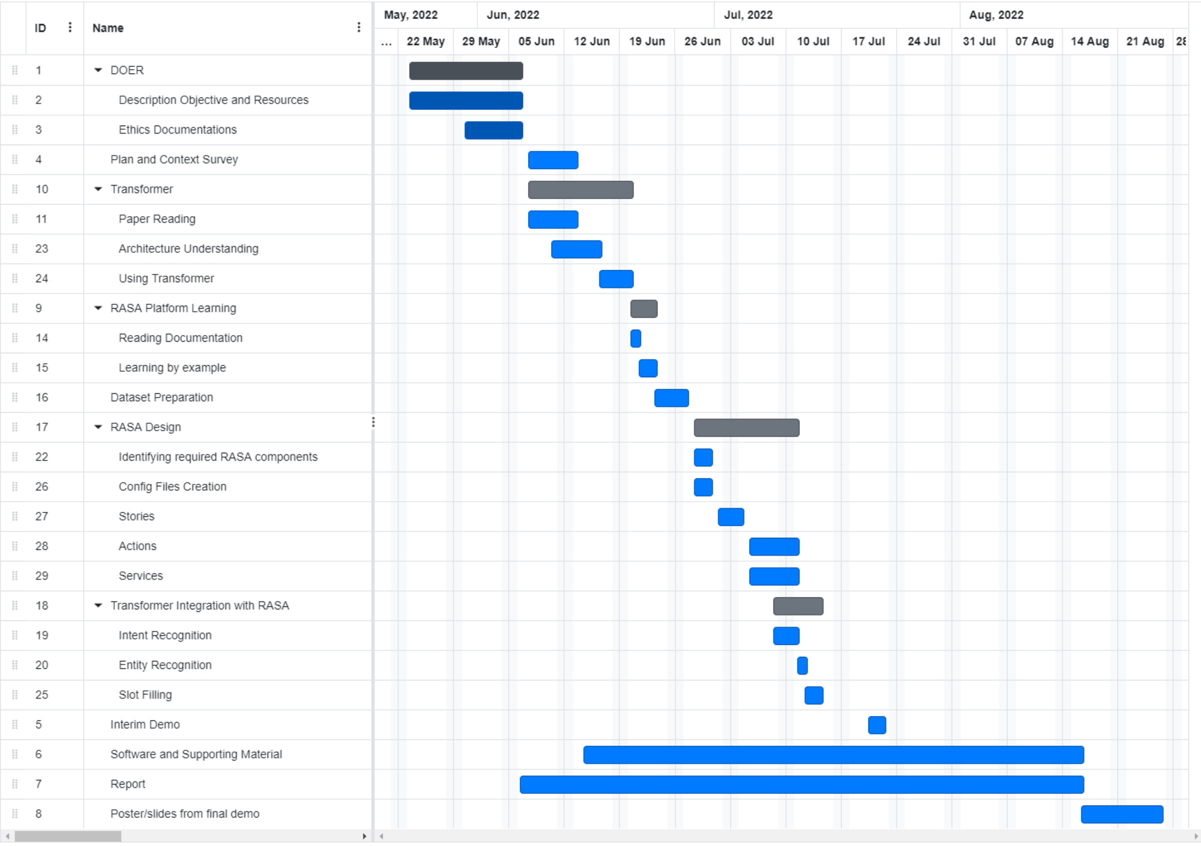
The Health Agent could achieve all the primary objectives and part of the secondary and tertiary objectives. Due to time constraints, collecting symptoms in multiple dialogue turns was partially achieved and could not be integrated into the main code. The feature can be developed by using a Boolean rasa slot indicating if the user has finished informing symptoms and when the slot is set then the Health Agent can stop collecting symptoms and move to the next action.

The responses from the survey indicate that users liked the interactions with the Health Agent, and it was observed that respondents were looking for some more advanced features. As suggested by respondents, it would be useful to add the functionality of getting treatment information for a particular disease in case the user wishes to know treatment details. Some respondents suggested that it would be informative if Health Agent can present the confidence it has in disease prediction. Maintaining the user’s profile with Health Agent and using the user’s existing health status and other key features can also provide better disease predictions. The disease and symptom matching service can be improved by employing a machine learning predictive model that can accept symptoms and some other extra features maintained in the user profile as its inputs and predicting possible diseases with their probabilities.

The NER model used for the disease was found to miss some of the tokens which are closely related to the actual symptom. It is believed that the model can be further customized to include those additional closely related word tokens as disease entities in the model. Additional capabilities for Handling spelling variations & typos can enhance the model performance and customer experience.

Appendix A: Project Timeline

Figure 32: Project Timeline



Appendix B: Test Report

```
=======================test session starts =======================
collected 6 items
tests/test_actions.py::Test::test_filter_fu PASSED
[ 16%]
tests/test_actions.py::Test::test_get_matching_disease PASSED
[ 33%]
tests/test_actions.py::Test::test_string_to_list_duplicate PASSED
[ 50%]
tests/test_actions.py::Test::test_string_to_list_empty PASSED
[ 66%]
tests/test_actions.py::Test::test_string_to_list_unique PASSED
[ 83%]
tests/test_actions.py::Test::test_string_to_list_upper PASSED
[100%]
=======================6 passed in 0.79s =======================
```

Appendix C: Dialogue Management and Slot Filling Validations

Rasa provides CLI `rasa interactive` for interactive learning of rasa. This CLI provides interactive insights into Rasa's NLU and DM predictions. Below is the output from Rasa interactive shell showing how slots and forms are used during the conversation.

Figure 35 shows that the model has extracted disease entities from the user's text and filled the slots named 'disease' and 'symptoms'. It also shows the active form 'gather_symptoms'

Figure 33: Rasa interactive CLI Output

Appendix D: User Guide

- Clone the repository to a local drive.
- Change the directory to the root of the cloned repository.
- Make sure all the python dependencies from requirement.txt are correctly installed.
- Download and install the Spacy language model
  ```
  pip install https://s3-us-west-2.amazonaws.com/ai2-s2-scispacy/releases/v0.5.0/en_ner_bc5cdr_md-0.5.0.tar.gz
  ```
- Follow the instructions mentioned below to access the Health Agent:

Train Rasa model

To train a model to be used by the agent, use the following command:

```
rasa train
```

Rasa server

To run the server, use the following command:

```
rasa run --cors "*"
```

The --cors "*" command is used to solve the cross-origin resource sharing (CORS) problem between the client and Rasa servers.

Run Action server

Run the following command to start the Rasa action server:

```
rasa run actions
```

Web client-server

Run the following command to start the web client for serving HTTP requests:

```
python -m http.server
```

Accessing the Health Agent:

This will start an HTTP-based server in the local 8000 port.

Visit `http://localhost:8000` in a browser to access the Health Agent.

Appendix E: Ethics Documents

UNIVERSITY OF ST ANDREWS
TEACHING AND RESEARCH ETHICS COMMITTEE (UTREC)
SCHOOL OF COMPUTER SCIENCE
ARTIFACT EVALUATION FORM

Title of project

Conversational AI for Primary Healthcare Support Advice

Name of researcher(s)

Vishesh Bhagat

Name of supervisor

Alice Toniolo

Self audit has been conducted **YES** ☒ **NO** ☐

This project is covered by the ethical application CS15727.

Signature Student or Researcher

Vishesh Bhagat

Print Name

Vishesh Bhagat

Date

06-06-2022

Signature Lead Researcher or Supervisor

Print Name

Alice Toniolo

Date

06.06.2022

Appendix F: Code Structure

The code structure of Health Agent is as follows.

```
📁 .HealthAgent/
├─📁 actions/
├─📄 config.yml
├─📄 credentials.yml
├─📁 data/
├─📁 db/
├─📄 domain.yml
├─📄 endpoints.yml
├─📄 github-markdown.css
├─📄 index.html
├─📁 notebooks/
├─📄 README.md
├─📄 requirements.txt
├─📄 showdown.min.js
├─📁 tests/
└─📄 webchat.js
```

The purpose of each file listed above is explained in the implementation section.

The purpose of each folder is as follows:

- actions: holds the code for the action server and backend of the Health Agent
- data: holds the training data for NLU and DM models
- db: holds the processed CSV file for disease and symptom matching
- notebooks: holds Jupyter notebooks used for disease data pre-processing
- tests: holds Rasa tests used for conversation testing

Appendix G: Survey Content

Healthy Agent responding to a healthy user:

During this conversation, the Health Agent understands that the user is healthy and does not ask for any further information.
It wishes the user good health and ends the conversation.

```
Your input ->  hii
hey there!
How are you feeling today?
Your input ->  i am feeling energetic
Great, stay fit!
Your input ->  thanks
Good luck!
```

Health Agent responding to an unhealthy user:

During this conversation, the Health Agent understands that the user is ill and asks for symptoms. The user challenges the agent's identity and asks random questions and then the user enters the symptoms.
The chatbot answers with the disease name matching the symptoms it understood and ends the conversation.

```
Your input ->  hello there
Hello there!
How are you doing today?
Your input ->  i am not feeling well today
Can you please tell me what symptoms you are observing?
Your input ->  who are you?
I am a Health Agent and your virtual health guide.
I try to match disease with the symptoms that you tell me.
Can you please tell me what symptoms you are observing?
Your input ->  can you make a pizza?
Sorry, I can't handle that request.
Can you please tell me what symptoms you are observing?
Your input ->  ok. I have neck pain, fatigue and blurred vision
Ok. I have collected all the required data
As i could understand, you have entered following symptoms:
neck pain
blurred vision
fatigue
Checking for diseases corresponding to the entered symptoms.
Your symptoms are matching with these diseases:
Ankylosing spondylitis
Your input ->  thanks
Good luck!
Your input ->  bye
Bye!
```

References

Amazon Web Services. (2019). *Amazon Lex – Build Conversation Bots*. Amazon Web Services, Inc. https://aws.amazon.com/lex/

Athota, L., Shukla, V. K., Pandey, N., & Rana, A. (2020, 2020). Chatbot for Healthcare System Using Artificial Intelligence. 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO),

Ben-Kiki, O., Evans, C., & Ingerson, B. (2009). Yaml ain't markup language (yaml™) version 1.1. *Working Draft 2008*, *5*, 11.

Botfront. (n.d.). *Better intent classification and entity extraction with Dietclassifier pipeline optimization*. Dialogue Technologies Inc. Retrieved July 10 from https://botfront.io/blog/better-intent-classification-and-entity-extraction-with-diet-classifier-pipeline-optimization

Braşoveanu, A. M., & Andonie, R. (2020, 2020). Visualizing transformers for nlp: a brief survey. 2020 24th International Conference Information Visualisation (IV),

Brush, K., & Scardina, J. (2021). What is a Chatbot and Why is it Important? https://www.techtarget.com/searchcustomerexperience/definition/chatbot#:~:text=Chatbots%20can%20improve%20lead%20generation,can%20engage%20with%20the%20leads.

Bunk, T., Varshneya, D., Vlasov, V., & Nichol, A. (2020). Diet: Lightweight language understanding for dialogue systems. *arXiv preprint arXiv:2004.09936*.

Damani, S., Narahari, K. N., Chatterjee, A., Gupta, M., & Agrawal, P. (2020). Optimized Transformer Models for FAQ Answering. In (pp. 235-248). Springer International Publishing. https://doi.org/10.1007/978-3-030-47426-3_19

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dialogue Technologies Inc. (2021). *GitHub - botfront/rasa-webchat: A feature-rich chat widget for Rasa and Botfront*. Dialogue Technologies Inc. https://github.com/botfront/rasa-webchat

Dolan, S. (2022). *How mobile users spend their time on their smartphones in 2022*. Insider Intelligence. Retrieved June 29 from https://www.insiderintelligence.com/insights/mobile-users-smartphone-usage/#:~:text=Mobile%20usage%20statistics,digital%20media%20time%20per%20day

Freed, A. (2021). *Conversational AI*. Manning Publications Co.

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. " O'Reilly Media, Inc.".

Gillioz, A., Casas, J., Mugellini, E., & Khaled, O. A. (2020). Overview of the Transformer-based Models for NLP Tasks.

Google. (2022). *Dialogflow*. Google Cloud. https://cloud.google.com/dialogflow

Gupta, V., Sood, A., & Singh, T. (2022, 2022). Disease Detection Using RASA Chatbot. 2022 International Mobile and Embedded Technology Conference (MECON),

Hugging Face. (2022). *Hugging Face Hub documentation*. Hugging Face. https://huggingface.co/docs/hub/index

IBM. (2022). *IBM Watson Assistant - Virtual Agent*. IBM. https://www.ibm.com/uk-en/products/watson-assistant

Kandpal, P., Jasnani, K., Raut, R., & Bhorge, S. (2020, 2020). Contextual Chatbot for Healthcare Purposes (using Deep Learning). 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4),

Khilji, A. F. U. R., Laskar, S. R., Pakray, P., Kadir, R. A., Lydia, M. S., & Bandyopadhyay, S. (2020, 2020). Healfavor: Dataset and a prototype system for healthcare chatbot. 2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA),

Kong, X., Wang, G., & Nichol, A. (2021). *Conversational AI with Rasa: Build, test, and deploy AI-powered, enterprise-grade virtual assistants and chatbots*. Packt Publishing Ltd.

Laumer, S., Maier, C., & Gubler, F. (2019). Chatbot acceptance in healthcare: Explaining user adoption of conversational agents for disease diagnosis. Proceedings of the 27th European Conference on Information Systems (ECIS),

M, P. (2020). *Diseases dataset*. Kaggle Inc. . https://www.kaggle.com/datasets/priya1207/diseases-dataset

Mantha, M. (2020). Introducing DIET: state-of-the-art architecture that outperforms fine-tuning BERT and is 6X faster to train. https://rasa.com/blog/introducing-dual-intent-and-entity-transformer-diet-state-of-the-art-performance-on-a-lightweight-architecture/

Mayo Clinic. (2022). *Mayo Clinic - Mayo Clinic*. Mayo Foundation for Medical Education and Research (MFMER). https://www.mayoclinic.org/

Microsoft. (2022). *Azure Bot Service – Conversational AI Application | Microsoft Azure*. Microsoft Retrieved June 1,2022 from https://azure.microsoft.com/en-us/services/bot-services/

Miller, G. A. (1995). WordNet. *Communications of the ACM*, *38*(11), 39-41. https://doi.org/10.1145/219717.219748

Miura, C., Chen, S., Saiki, S., Nakamura, M., & Yasuda, K. (2022). Assisting Personalized Healthcare of Elderly People: Developing a Rule-Based Virtual Caregiver System Using Mobile Chatbot. *Sensors*, *22*(10), 3829. https://doi.org/10.3390/s22103829

Mukhopadhyay, S. (2018). *Advanced Data Analytics Using Python With Machine Learning, Deep Learning and NLP Example*. Apress.

Nadarzynski, T., Miles, O., Cowie, A., & Ridge, D. (2019). Acceptability of artificial intelligence (AI)-led chatbot services in healthcare: A mixed-methods study. *DIGITAL HEALTH*, *5*, 205520761987180. https://doi.org/10.1177/2055207619871808

National Center for Biotechnology Information. (2013). *Rezarta Islamaj Dogan's Research Page*. National Center for Biotechnology Information. https://www.ncbi.nlm.nih.gov/CBBresearch/Dogan/DISEASE/

Neumann, M., King, D., Beltagy, I., & Ammar, W. (2019). ScispaCy: Fast and Robust Models for Biomedical Natural Language Processing. *ArXiv*, *abs/1902.07669*.

Palash Goyal, S. P., Karan Jain. (2018). *Deep Learning for Natural Language Processing: Creating Neural Networks with Python*. Apress.

Pandey, A., Mutreja, I., Brar, S., & Singh, P. (2020, 2020). Exploring Automated Q&A Support System for Maternal and Child Health in Rural India. Proceedings of the 3rd ACM SIGCAS Conference on Computing and Sustainable Societies,

Patel, A. A., & Arasanipalai, A. U. (2021). *Applied Natural Language Processing in the Enterprise*. " O'Reilly Media, Inc.".

Perera, S., Sheth, A., Thirunarayan, K., Nair, S., & Shah, N. (2013). Challenges in understanding clinical notes: Why NLP engines fall short and where background knowledge can help. Proceedings of the 2013 international workshop on Data management & analytics for healthcare,

Rakib, A. B., Rumky, E. A., Ashraf, A. J., Hillas, M. M., & Rahman, M. A. (2021). Mental Healthcare Chatbot Using Sequence-to-Sequence Learning and BiLSTM. In (pp. 378-387). Springer International Publishing. https://doi.org/10.1007/978-3-030-86993-9_34

Rasa Technologies Inc. (2022). *Introduction to Rasa Open Source*. Rasa Technologies Inc. Retrieved June 5 from https://rasa.com/docs/rasa/

Revang, M., Mullen, A., & Elliot, B. (2022). Magic Quadrant for Enterprise Conversational AI Platforms. https://www.gartner.com/doc/reprints?id=1-28XJAIXS&ct=220128&st=sb&__hstc=123545108.8bdd386aec815b71400a88628eca825d.1651495774530.1655382928568.1655389108359.89&__hssc=123545108.1.1655401502084&__hsfp=2908620477&hsCtaTracking=36211c5b-db91-4993-bc0a-88e4cc10d0bd%7C162fc67a-d304-406c-a5db-6a392893e120

Rustamov, S., Bayramova, A., & Alasgarov, E. (2021). Development of Dialogue Management System for Banking Services. *Applied Sciences*, *11*(22), 10995. https://doi.org/10.3390/app112210995

Sheth, A., Yip, H. Y., & Shekarpour, S. (2019). Extending Patient-Chatbot Experience with Internet-of-Things and Background Knowledge: Case Studies with Healthcare Applications. *IEEE Intelligent Systems*, *34*(4), 24-30. https://doi.org/10.1109/mis.2019.2905748

Singh, A., Ramasubramanian, K., & Shivam, S. (2019). *Building an enterprise chatbot: Work with protected enterprise data using open source frameworks*. Springer.

Singh, P., Lin, T., Mueller, E. T., Lim, G., Perkins, T., & Li Zhu, W. (2002). Open Mind Common Sense: Knowledge Acquisition from the General Public. In (pp. 1223-1237). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-36124-3_77

Tarcar, A. K., Tiwari, A., Dhaimodker, V. N., Rebelo, P., Desai, R., & Rao, D. (2019). Healthcare NER models using language model pretraining. *arXiv preprint arXiv:1910.11241*.

Tunstall, L., Werra, L. v., & Wolf, T. (2022). *Natural Language Processing with Transformers* (Revised Edition ed.). O'Reilly Media, Incorporated.

U.S. Department of Health & Human Services. (2022). *Health Insurance Portability and Accountability Act of 1996 (HIPAA)*. U.S. Department of Health & Human Services. https://www.cdc.gov/phlp/publications/topic/hipaa.html

Vajjala, S., Majumder, B., Gupta, A., & Surana, H. (2020). *Practical Natural Language Processing*. O'Reilly Media.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Williams, S. (2018). *Hands-On Chatbot Development with Alexa Skills and Amazon Lex: Create custom conversational and voice interfaces for your Amazon Echo devices and web platforms*. Packt Publishing Ltd.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., & Funtowicz, M. (2020). Transformers: State-of-the-art natural language processing. Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations,

Yu, S., Chen, Y., & Zaidi, H. (2020). A Financial Service Chatbot based on Deep Bidirectional Transformers. *arXiv preprint arXiv:2003.04987*.