Towards the Implementation of an Intelligent Software Agent for the Elderly

by

Amir Hossein Faghih Dinevari

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

Master of Science

Department of Computing Science

University of Alberta

Abstract

With the growing population of the elderly and the decline of population growth rate, developed countries are facing problems in taking care of their elderly. One of the issues that is becoming more severe is the issue of companionship for the aged people, particularly those who chose to live independently. In order to assist the elderly, we suggest the idea of a software conversational intelligent agent as a companion and assistant.

In this work, we look into the different components that are necessary for creating a personal conversational agent. We have a preliminary implementation of each component. Among them, we have a personalized knowledge base which is populated by the extracted information from the conversations between the user and the agent. We believe that having a personalized knowledge base helps the agent in having better, more fluent and contextual conversations. We created a prototype system and conducted a preliminary evaluation to assess by users conversations of an agent with and without a personalized knowledge base.

Table of Contents

List of Tables

List of Figures

Acronyms

A.L.I.C.E. Artificial Linguist Internet Computer Entity.

ACE Advanced Content Extraction.

AI Artificial Intelligence.

AIML Artificial Intelligence Markup Language.

ANA Automated Nursing Agent.

CALO Cognitive Assistant that Learns and Organize.

CKB Context Knowledge Base.

CRF Conditional Random Field.

DARPA Defense Advanced Research Projects Agency.

IE Information Extraction.

KR Knowledge Representation.

ML Machine Learning.

MUC Machine Understanding Conference.

NLG Natural Language Generation.

NLP Natural Language Processing.

NLU Natural Language Understanding.

PKB Personalized Knowledge Base.

POS Part Of Speech.

RG Response Generation.

SAR Speech Act Recognition.

SVM Support Vector Machine.

TAC Text Analysis Conference.

XML Extensible Markup Language.

Chapter 1

Introduction

Today, many assistive technologies can be found with the goal of helping the elderly. However, there are few technologies which exploit artificial intelligence to their advantage. With the increase in the elderly population, these assistive technologies can prove to be beneficial. If they can reduce the amount of work that is done by caregivers or nurses, many problems related to the growing population of senior citizens [50] can be solved. In this project, our goal is to go one step further with these technologies. We are trying to build an intelligent companion and assistant for the elderly with the aid of artificial intelligence.

1.1 Motivation

How the elderly live and the resulting impact on the health care system are some of the issues that are gaining momentum. Moreover, the growing population of seniors, how they are accommodated, and the psychological and physical effects of the modern world and evolving culture on them, are some of the reasons why more countries are investing in assistive technologies. In this section, we will discuss some of these reasons that have increased attention to this topic.

1.1.1 Population Trends

With the growing number of elderly people in developed countries, taking care of this aging population is becoming more of a focus for governments and their respective health organizations. Currently, in the United States, there are 41

million people that are aged 65 or older [16]. This number accounts for 13% of the population. Also in Canada, there are 5 million seniors aged 65 or older (about 14% of the population) [78].

There are some factors at play, resulting in this population trend in developed countries. In recent years, advancements in healthcare and medicine eliminated many causes of fatality [54]. This elimination resulted in a significant increase in life expectancy [41]. Also, there is a decline in population growth in developed countries [31]. As a result of these changes, today, compared to last century, a larger portion of the population is old. Some studies believe this trend will continue. According to the United States Census Bureau, by 2050, 20 percent of the US population will be 65 or older and from this population, at least 400,000 will be 100 years or older [16].

As a result, financial issues would arise because of having a larger number of seniors compared to younger people. Many countries with free healthcare depend on the money from taxpayers to fund the necessary infrastructure and personnel. With bigger bills related to the elderly while having a smaller portion of the population funding the healthcare system, governments and people will face many problems, unless there are changes to the current system that can make it sustainable to this population change trend.

1.1.2 Living Options for the Elderly

Based on the needs of the elderly, there can be different living options for them. They can live in nursing houses with other seniors, which can be helpful since they can keep each others company. They can stay at home by themselves if they can take care of themselves; if they need help, they can have a family member living with them or hire a caregiver that can check up on them or live with them if it is necessary.

However, there can be some issues with each of these options. In the case of living in nursing homes, one of the main issues is the insufficient number of these places and their capacity. It is most likely that they will not be up to scale with the elderly population growth. Also, the quality of services in such facilities are sometimes debated.

According to the Canada Statistics report, 92% of seniors in Canada are living in their own private places [78]. Living in private households has financial, social and health advantages [42]. Living in private houses, provided the elderly can maintain their social connections and take care of themselves, is a good option [12]. If the elderly are not autonomous in their daily activities, living in private places by themselves is not a good choice. In this case, there is a need for someone to check up on them regularly. This job can be done by a family member or a caregiver.

Another issue arises from the number of necessary caregivers that is needed to assist seniors. Because of this high demand, some countries like Canada have immigration programs exclusive to caregivers. The federal immigration plan of Canada in 2016 shows that more than one-forth of the professional immigrants is set to be caregivers [36]. By reducing the number of needed caregivers by other means, the efficiency of the healthcare system can be increased and imposed expenses to it will be lessened.

1.1.3 Companionship

Aside from the growing population and living issues, one of the most important aspects of life for everyone, especially elderly people, is companionship. Humans are social beings and companionship has become an essential part of their lives [11]. They need to have interactions with other people. With these interactions, people share different aspects of their lives with others. This need gets accentuated by getting older in a sense that social isolation and loneliness can predict declining health and poor quality of life for seniors [56].

Companionship is also considered an important part of the healing process for hospital patients. Patients in the hospital can feel lonely, which may affect them psychologically. To eliminate this problem, in some hospitals, there are companion care programs where volunteers engage with patients in small talk and social activities [17]. The elderly living alone can have the same problem. Their family members or friends may not be close enough to visit them regularly and keep them company.

People living in nursing homes usually have others around them. They

can talk and interact with each other, which is beneficial to them [85]. In the case of people living in their own places, the situation is a little bit different. If they go out they can have conversations with friends or neighbours. Some even take pets as companions. It is proven that having pets as companions can improve elderly people's mental health [30]. While having pets can be entertaining, people cannot have meaningful conversations with them. In the case of the elderly who cannot take care of themselves, having a pet is not even an option. The only people who can keep their company are friends, family members and caregivers. Sometimes the caregivers are from different countries and they cannot converse in the same language as the seniors.

1.1.4 Current Technologies

Many projects for assistive technologies have been started in different countries with different defined criteria. Among these technologies, there are robots that are designed to help the elderly with their everyday tasks. Also, there are technologies developed to monitor elderly people's health condition remotely and devices designed to remind the elderly about their medication. All of these assistive technologies can help in reducing daily responsibilities of caregivers.

In the design and proposition of these assistive technologies and devices, there are a few designers considering companionship for elderly people. With the current changes in the world, it is possible that companionship will become a greater issue than the problems that current technologies are addressing.

If we look further into intelligent assistants and their technologies, we can see the current applications of such systems are very limited. Some of the famous assistants that are widely available are Apple Siri, Google Now, and Microsoft Cortana. All of these assistants are designed in the same way to solely address commands or answer search or information retrieval related questions.

1.2 Proposed System

We propose a talking virtual intelligent agent to help the elderly. We named our system Automated Nursing Agent (ANA). This agent is in the form of an Android app and the main input method for the agent is speech. The reason for choosing speech for input is that new technologies can seem confusing to the elderly, while conversing is something that most people are used to doing. The system has a pipeline approach to processing the speech. First, it converts the speech to text and runs the text through speech act recognition and information extraction components. Then according to the type of sentence, being declarative, interrogative or imperative, it tries to come up with a reasonable response. Figure 1.1 shows the simplified pipeline for the system and some of its components.

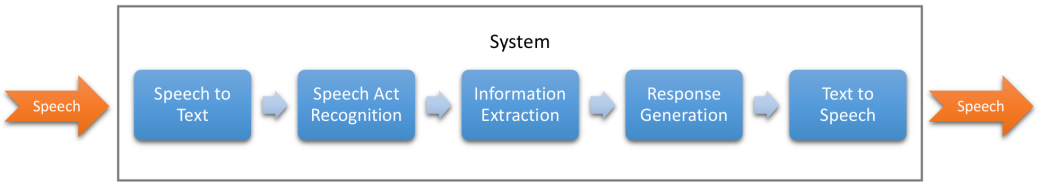


Figure 1.1: Simplified pipeline of the system

In this system, the user is not necessarily the person who starts the conversation. ANA, sometimes, according to the user's habits, may come up with questions related to their health or everyday life. The functionalities of our system can be categorized into three groups: personal assisting, companionship and nursing.

1.2.1 Personal Assistant Functionalities

ANA is able to perform many tasks that are done by current personal assistant systems. It can remind the elderly about the events they have planned, send messages to different people, create and update to-do lists, make phone calls, send emails and much more. Since ANA is using Android devices, it can have access to contacts, calendar, call logs, emails and all the other necessary information to do these tasks.

These tasks are usually in the form of commands and can be easily detected when the input goes through the speech act recognizer. To know what to do, the system needs to extract information from the input. It needs to know what to do and how to do it. If it is calling or sending an email to someone, it needs to know who the contact is. If it is an event, information like the time, date and the place of the event are needed. In the case of not having all the necessary information in the initial command, ANA can ask follow up questions. These questions can be about the person, when the name is not stated, or if there are ambiguities in the commands. These ambiguities can be entities when we have more than one entity with the same name or when the content is absent from initial command for sending an email or a message.

After completing the task, ANA acknowledges the status of the command saying if it was successful or not. This part also includes a text to speech component to convey the necessary response to the elderly.

1.2.2 Companionship Related Functionalities

One of the main goals that we are trying to achieve with this system is making it a companion to the elderly. We want ANA to converse with the elderly fluently. Although ANA has personal assistant functionalities, it is not designed to compete with the existing personal assistants. ANA is trying to make a personal connection with the elderly through listening to their stories about their lives and relating to them. Some of the implemented capabilities in ANA include telling jokes, for which ANA learns the preference of the user using reinforcement learning to choose more appreciated ones, telling stories, finding recipes, reading, and other planned capabilities.

In order to play the role of companion, ANA needs to mimic how humans behave in some of the aspects of conversations. The response to each sentence should be related to the context of the conversation. To generate appropriate responses, ANA includes speech act recognition in its pipeline to know what kind of sentence is being told. If it is a declarative statement, it needs to acknowledge the understanding of the story as the person is speaking, using declarative statements or follow-up questions related to the story, to show its

interest.

In the case of answering a question, ANA tries to answer it based on the knowledge that it gathered about the person in its personalized knowledge base. If it is a general question, it is favourable to get an answer from the system. ANA uses public general knowledge bases to answer these kinds of questions. If ANA could not answer a question, it will show the web search results for that question, which can be useful to the elderly.

ANA needs to deal with imperative statements too. If the commands or requests are somethings that can be done within the device, ANA tries to address them. If it is not able to address the command, due to limited capabilities, it should try to come up with a helpful response to the request.

With ANA, the user does not always start the conversation. The system can start a conversation if there is a long pause or a period of inactivity. It can also start conversations with the elderly about the reminders that he or she added to the system. These conversation openers can be related to the previous conversation in case of having a long pause. This kind of related conversation topics, in general, help the system act more natural.

For good reception of the proposed system, there should be user studies about the preferences of the elderly. The tone of conversations, for example, is one of the characteristics that should be set to follow the elderly's preference. Also, the voice that is going to be used in the final system is important for having better conversations with elderly. These studies should be conducted before releasing the system.

1.2.3 Nursing Functionalities

Nursing capabilities in ANA are vital. The target user base for this system is the elderly who are staying at home alone and do not have many frequent visitors. ANA should help them with the medical and health related issues that they may have.

One of the things that ANA should track regularly is the medication intake by the elderly. ANA can get related information about what medication should be taken by the elderly every day and when they should take it. It asks

questions at the right time to see if the users have their medication, and if not, remind them to take it.

Caregivers usually ask the elderly how they feel, and sometimes they keep a record of it. ANA can do the same thing by asking questions about how they feel and if they feel any different compared to the previous days. This information can be helpful later for the doctor who is checking up on the elderly.

ANA also keeps a log of what the elderly do daily. This log can be used for detecting unusual patterns in their behaviour. These patterns can be symptoms of a disease, which can be detected sooner by the help of these logs, when they are viewed by a doctor or a healthcare professional. In order to give access to this log, we also have a web interface that can be used by authorized people to access ANA.

Sometimes there may be an emergency situation involved. While it is very hard to detect such irregularities, ANA at least can recognize when the elderly is unresponsive for a period of time. In those situations, ANA can contact a family member, caregiver or an ambulance based on the defined rules.

Because of having nursing functionalities, there are ethical issues that should be addressed in our system. In the case of not listening to the recommendation of the agent, how the situation should be handled can cause ethical issues. For example, in the case of reminding the elderly about the medication that they should take, how the system should react to the user's response can be debated. Reminding the user again, contacting someone, or just skipping can be some of the proper approaches depending on the situation; for instance whether not taking the medicine is harmful, whether the medicine is hugely beneficial, etc. Also, because the system deals with health and wellness of a person, ethical liabilities for implementing such a system exist since the improper handling of a situation can put the health of the user in jeopardy. These issues should be studied and evaluated properly for handling different situations.

1.3 Problem Definition and Research Methodology

Recently, intelligent agents have become more popular. Giant technology companies like Amazon, Apple, Google, and Microsoft are investing significant resources for creating software agents also known as virtual intelligent chatter-bots. Their design is mostly query-based and they are not targeting conversations. Also, their responses are mainly in the form of declarative statements.

Our system is following another goal. Conversation is an integral part of our system. Because of the different types of the sentences that the system may encounter in a conversation, it needs to identify these types. Also based on the statement from the user, it needs to have different forms of responses. The context of the conversation is very important for having a fluent conversation too. It should remember the last mentioned entities in order to link it with pronouns further in the conversation.

There are many problems in the way of designing such a system. We need to identify the building blocks of the system. The necessary methods for implementing these components are actively progressing by current research and technology companies. We are doing a preliminary implementation of these components which may possibly not be comparable with what is the state-of-the-art.

According to the pipeline of the system, the first step is speech to text conversion, and the last one is text to speech conversion. Because of its high accuracy, we will utilize Android speech engine for the purpose of speech conversion. These are the main problems that we need to address in the design of our system:

- **Speech Act Recognition:** There is research centred on speech act recognition, but in many cases, the acts captured by them are not the ones that we need. We need to categorize statements into declarative, interrogative and imperative groups only.

- **Information Extraction** is also subjective to the application which makes it very hard to use in other systems. This component needs to be specifically designed for our system.

- **Knowledge Representation:** Extracted information should be stored in a structured way to be usable by the system for conversations. Having information in a structured way for machine interpretation is called Knowledge Representation. The problem here is devising the necessary tools with the objective of capturing all the relevant information.

- **Response Generation** can be considered the main response mechanism of ANA in conversations. The responses can be declarative or interrogative according to the context of the conversation. Having the appropriate response is crucial.

- **Question Answering:** System should be able to answer personal and general questions, which may need a reasoning on different knowledge bases. It should utilize extracted information for personal questions and public knowledge bases or the Internet for general questions.

1.4 Thesis Statement

In this work, we are identifying the necessary building blocks of a conversational smart agent and we combine them with natural language processing tools to design a system with better conversation abilities, while achieving the goals set in system design. We claim that:

> *Machine generated conversations are more natural and fluent when incorporating a personalized knowledge base that is populated with the information from the context of the previous conversations.*

To demonstrate our claim, we need to have all the problems defined in the previous section addressed appropriately in order to extract information from previous conversations and store them in a personalized knowledge base. This

structured information will be used in the context of new statements in the conversation.

1.5 Thesis Contributions

We have devised several components addressing the mentioned problems in Section 1.3. We have tried to implement and design each component in a way that it can operate as independently as possible. These are preliminary but operational modules as a proof of concept. This approach, helps us replace components easier in future implementations while keeping others intact. These components are:

- **Speech Act Recognizer (SAR:** This component helps the system identify the category of each sentence. We are using SVM as the classification method of our SAR and our evaluations show a gradual improvement in performance compared to the previous work.

- **Information Extractor (IE):** Most of the information that we have are included in utterances. However, the converted textual sentences from utterances are not understandable by a machine. We need to extract the information that is necessary from each sentence before storing them in a knowledge base. We have designed an IE system specific to our application with the help of NLP tools and handcrafted rules.

- **Knowledge Representation (KR):** Since the system relies on the information from prior conversations with users, this information should be stored in a machine-readable format. To this goal, we have created a KR encompassing a personalized knowledge base and context knowledge base to store such information.

- **Response Generator RG:** We have created a response generation mechanism that utilizes different knowledge bases. It can answer questions in the general domain and also in the PKB domain. The RG depends on rules for generating appropriate responses to each statement.

1.6 Organization of the Dissertation

For each one of the described components in Section 1.5, we have a separate chapter. Each chapter has its own related work section with a conclusion and future directions specific to that component and the current state of it. Having this explanation in mind, we have seven chapters:

- **Chapter 2.** ***Related Work:*** It includes the related work to this project that does not specifically fit under the chapters related to each component.
- **Chapter 3.** ***Speech Act Recognition***
- **Chapter 4.** ***Information Extraction***
- **Chapter 5.** ***Knowledge Representation***
- **Chapter 6.** ***Response Generation***
- **Chapter 7.** ***System Design:*** In this chapter, we describe how we created a system based on the devised components in previous chapters.
- **Chapter 8.** ***Conclusion:*** We conclude the dissertation with our findings and an overall evaluation of the final system.

Chapter 2

Related Work

Looking at the existing research for assistive technologies for the elderly, smart agents and their underlying components can be helpful in devising a better system. In this chapter, we review some of the related work relevant to our project. In the first section, we will review some of the projects with the goal of helping the elderly since our project is following the same goal.

Conversing is an integral part of an agent which relies on speech as its main method of communication. It is one of the old problems in AI that researchers have been working on since the 1960s. Chatterbots are the oldest computer programs that were trying to do such a task. Exploring how these systems work and how well they perform can help us with designing a system with better conversation ability. In the second section, we will examine some of them and popular implementation methods that are being used by developers.

Many tech companies are working on smart agents that can help people with their everyday tasks. ANA has the same responsibility but it has a different audience and a different set of tasks. These systems utilize AI and Natural Language Processing in order to answer questions, perform commands, etc. Reviewing their system design can be beneficial for designing new systems which can also incorporate some of these personal assistive functions. In the third section, we will examine some of them and their technical specifications.

In chatterbots and smart agents, there is a shared component that is necessary to deal with natural language. However, it is not necessary to design such a component from scratch since there are many Natural Language Processing

tools available for developers today. These tools help computer programs to get some information about the context and nature of natural language. By using these tools, developers created impressive software agents that can have a better understanding of the natural language input that they are getting from users. In the fourth section, we will have a quick review of those tools.

2.1 Assitive Technologies for the Elderly

Modern life has many effects on the way that people live all around the world, and the way that the elderly live is not an exception. There are technologies developed specifically to help seniors in their daily tasks. The assistive technologies contain a wide range of software and hardware products. We are going to look into some of these technologies that help the elderly with their everyday life.

2.1.1 Home Environment Systems

Designing a system to help the elderly with interacting with the environment is not a new idea. There are devices ready to deploy in the houses with extensive support. However, the tech companies and researchers are working on the idea of smart homes to replace traditional systems. Smart homes are not necessarily designed for the elderly but they can be very helpful with how the elderly interact with the environment around them [7]. Smart homes for the elderly can include many components like sensors for fall and emergency detection, smart appliances, security system, etc. One of the examples in this category is the smart house that is designed in University of Florida to ease the elderly's interaction with appliances and home environment [24].

2.1.2 Medication Reminder Tools

The importance of medication for the health of the elderly is undeniable. In the case of the elderly who live alone by themselves, medication taking can be forgotten. There are many tools that are developed to remind medication to the elderly. Some of these tools are in the form of software that can be

installed on different devices like smartphones and tablets and in some cases these tools have a simpler form like a pill bottle with a blinking light on top of it [40].

2.1.3 Assistive Robots

Assistive robots come in different sizes and purposes. The goal of these robots can vary from physical help to being a social companion. The robots helping with the physical tasks can bring different objects, do some chores, and in the bigger robots, sometimes, move the elderly. Some of these robots may have social aspects too. Social robots are designed to be a companion to the elderly who live alone. However, the culture of the country that these robots will be used in is affecting the adaptation rate of such devices. Most of these robots are actively being used in Japan since people there have a higher acceptance of devices mimicking humans [52].

2.2 Chatterbots

Chatterbots are one of the oldest concepts that have been worked on since the foundation of Artificial Intelligence. Chatterbots are also known as chatbots and artificial conversational entities. Sometimes they are referred to with the general term bots. Chatterbots are designed to have intelligent conversations with people. One of the goals in artificial intelligence was to create a program which can think and understand like a human being. Chatterbots were part of this general system that scientists were working on.

2.2.1 Understanding Ability of the Chatterbots

In the early years of artificial intelligence, people believed that if a program can have an intelligent response in the context of a conversation, we can assume that this program has an understanding of the statements. However, there was another group of people who opposed this interpretation by saying that having a basic response is not enough to conclude the understanding ability of a machine.

Since it is very hard to prove if machines think like humans or if they understand them, Alan Turing came up with a simpler criterion for evaluation of chatterbots [83]. How well a computer can imitate human intelligence became the evaluation method for chatterbots and intelligent systems. Turing devised a test for chatterbots which is widely known as the Turing test [84].

John Searle opposed Turing's definition of intelligence saying that it does not show the thinking ability in machines [70]. From there, the idea of weak and strong artificial intelligence came to life. Strong AI is considered to a system that can analyze and reason on a human level. Since Turing's test does not care about the analyzing and reasoning capabilities of intelligent systems, a system passing Turing test is considered weak AI.

Even with these debates about what is considered an intelligent system in the 1960s, today there still is no system that has passed the Turing test which was considered weak AI by Searle. This shows that how far we are from the ideals that are set in AI field.

2.2.2 Notable Chatterbots

The first well-known chatterbot was developed by Joseph Weizenbaum [89]. It was named ELIZA and published in 1966. It was mimicking an on-going conversation in a person-centered psychotherapy session. ELIZA gained a lot of attention from specialists in AI as well as the general public.

The second notable chatterbot was PARRY designed by Kenneth Colby in 1972 [20]. PARRY was simulating a paranoid schizophrenic person. Compared to ELIZA, which was just substituting keywords and phrases from human speech in its own templates, PARRY utilized a crude line of a conversation which made it a more advanced program.

Since 1990, based on the Turing test, an annual contest for chatterbots has started. Hugh Loebner and The Cambridge Center for Behavioral Studies came up with the general idea of the contest. Among the winners, there are some outstanding chatterbots. Albert One by Robby Garner won the contest in 1998 and 1999. A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) by Richard Wallace [87] is one of the most famous chatterbots which won the

contest in 2000, 2002 and 2004. Elbot by Fred Roberts won the contest in 2008 and it was very close to passing the Turing test.

2.2.3 Common Implementation Methods of Chatterbots

There are many ways to implement chatterbots. Over time, with the improvement of the AI and NLP tools, people used more sophisticated methods in their software design. They moved from using only pattern matching to using NLP tools and AI for better response generation. However, with all the improvements, there has not been a breakthrough in the functionality of chatterbots. The main implementation methods for chatterbots are listed below.

- **Pattern Matching** is one of the oldest methods for implementing chatterbots. It consists of a set of templates which can be matched with the input from the user to generate a response. ELIZA used this technique to generate responses which were mostly questions.

- **Parsing** can be done in different variations. The earlier chatterbots were looking for a set of pre-defined words in a specific order. By having this kind of parsing added to pattern matching, chatterbots were able to answer more questions. Newer chatterbots use natural language processing for grammatical parsing of the input.

- **Markov chain models:** In some chatterbots, Markov chain models were used to evaluate possible responses. In the case of having a set of plausible responses, the response with higher probability would be chosen by the chatterbot.

- **Ontologies:** In a very small number of chatterbots, ontologies are used to do partial reasoning on different concepts. OpenCyc is one of the ontologies that some developers tried to incorporate within their chatterbots. Ontologies are sometimes referred to as Semantic Networks by some chatbot systems.

- **AIML** is a method of defining patterns in chatterbots. It is written in XML form and has its own syntax. It has the ability of recursively calling

itself and also it is possible to use a wildcard (*) within the patterns. A.L.I.C.E. is using AIML as its pattern matching mechanism.

- **Chatscript** is a new alternative for AIML. It has better syntax and also has more functionalities compared to AIML. It is used in Suzette, Rosette and Rose (Winners of Loebner award in 2010, 2011 and 2014) by Bruce Wilcox and Chip Vivant (Winner of Loebner award in 2012) by Mohan Embar.

Even with integrating these methods there are not chatterbots that use the context of the conversation efficiently for creating appropriate responses. Using the information that is given to the system by the user in the conversation can help the chatterbots generate more appropriate responses.

2.3 Smart Agents

Many tech companies are working on smart agents that can help people with their everyday tasks. These systems utilize AI and Natural Language Processing in order to answer questions, do commands, etc. Going over their system design, even with the limited information that is available about them, can be beneficial for designing new systems which can also incorporate some of these personal assistive functions. In Section 2.3.1, we will look at where smart agents come from and in Section 2.3.2, we will review some of the distinguished smart agents.

2.3.1 Definition of Software Agents and Smart Agents

The idea of software agents is one of the concepts that was defined before this century but we see its implications and capabilities today. According to a weak notion of agenthood [91], to consider a system as a software agent, it should:

- Not need interaction of user and be able to activate itself without explicit invocation (Autonomy)

- Be able to stay in a waiting condition while perceiving the context (Reactivity)
- Be able to start when it is facing a starting condition (Proactiveness)
- Be able to start other tasks, including communication, when necessary (Interfacing/Social Ability)

Nwana believed that this definition did not capture many characteristics of software agents [59], so he came up with a categorization of software agents according to their capabilities. Figure 2.1 shows different types of agents based on the attributes defined by Nwana. According to those attributes, a smart agent should have all the defined attributes.

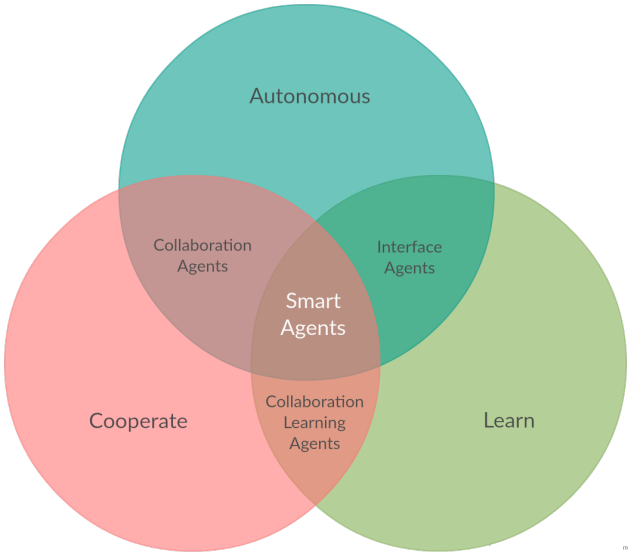


Figure 2.1: Software agent categories according to Nwana

Even current advanced personal assistants that we are using today may not meet all of these characteristics. However, we consider agents like Google Now, Apple Siri, Microsoft Cortana etc. as smart agents in this context. Software agents are not limited to personal assistants, but in the context of our research, these assistants are more relevant.

2.3.2 Examples of Smart Agents

Recently many smart agents have been released by huge tech companies and startups but only a few of them are widely adopted by the general public. In this part, we look over some of these smart agents in the order of their initial public release date.

2.3.2.1 Apple Siri

Apple Siri can be considered as the first well known and successful intelligent assistant that was made available publicly. Siri began as a startup company separated from SRI International. The technology that was used within the project shared many characteristics by the project funded by Defense Advanced Research Projects Agency (DARPA).

DARPA funded Cognitive Assistant that Learns and Organize (CALO) project within SRI International for five years to create personal assistants that can be used by armed forces. People who were actively working on CALO project created a startup to develop an intelligent assistant for smartphones. Initially, Siri became available for iPhone but later on, the startup was acquired by Apple to make it the default assistant for Apple products.

According to the interview with Tom Gruber, chief technology officer of Siri [77], at the time who was also involved with CALO project, Siri is a more task-focused assistant. It can manage a predefined limited set of tasks with the help of third party APIs. He also mentions that their goal was not about managing or browsing semantic data. Although it was not one of the priorities in the project, Siri can answer questions with the help of third-party data providers like Wolfram Alpha, Wikipedia etc.

Since Siri is using speech as the method of communication, it needs to process speech and natural language to understand commands or questions. The handling of speech in Siri is done by technologies developed by Nuance Communications. The natural language processing tools are developed by Siri team in the house but they also use some of the technologies that were developed within CALO project. Siri supports more than 20 languages with

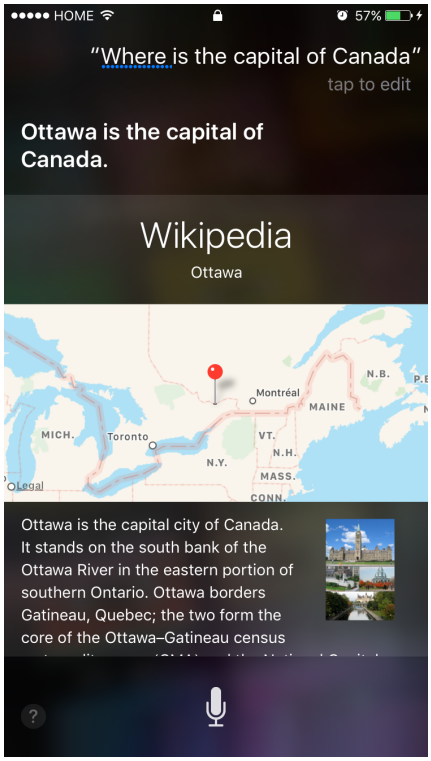


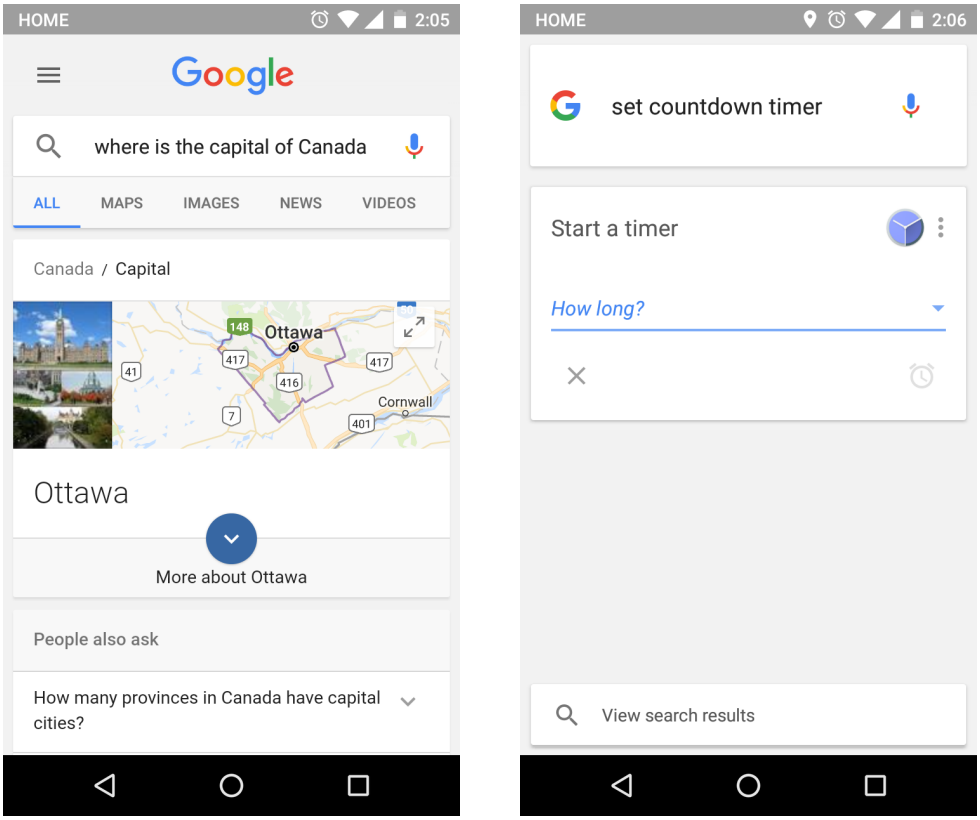
Figure 2.2: Interface of Apple Siri

different dialects.

2.3.2.2 Google Now

Google Now is an intelligent personal assistant developed by Google. It includes now cards, voice search and voice commands. Now cards are used to show relevant information to users about current or future events. Google uses emails, web search history, location history and other personal information that it gathers about users to show most relevant cards to the user. Voice search functionality and results are very similar to what is displayed on the Google website. Google search on web and mobile can find the answer to questions that are in natural language form. Voice commands incorporate device capabilities and some third party apps functionalities to give a better user experience with speech on mobile devices.

There are not many technical details available about the implementation of Google Now on devices, but like other assistants, it uses speech and natural language to communicate with users. Google is using their own propriety



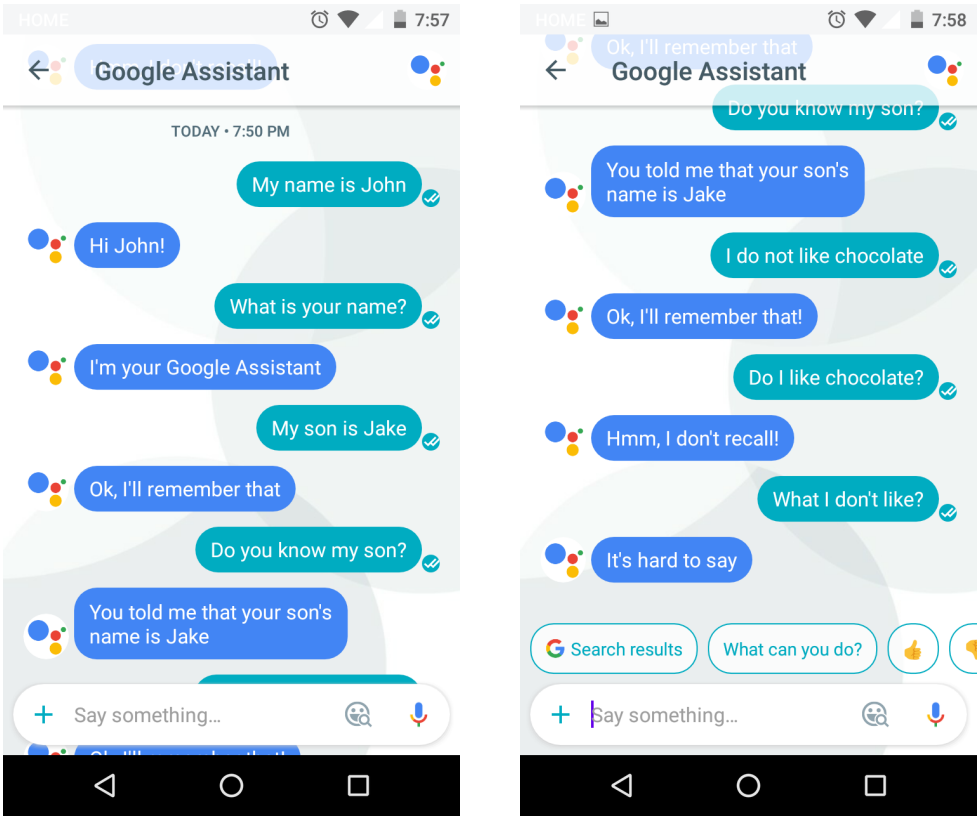
(a) Question answering (b) Setting timer

Figure 2.3: Google Now and Voice search on Android

technologies in Google Now. For speech and natural language processing, TensorFlow [3] is used to train deep neural networks. The natural language parser that is created by Google for this purpose is called ParseyMcParseface [32]. This NLP parser has higher accuracy compared to other available parsers.

To answer questions in the form of natural language, Google uses its Knowledge Graph. Initially, most of the data in Google Knowledge Graph was user contributed data from Freebase. Over the time, Google used the information that it gathered through web scraping [27] to Knowledge Graph to make it more comprehensive.

Aside from Google Now, which was the main virtual assistant on Android platform, Google released a new smart agent called Google Assistant. This agent, according to Google, is more conversational compared to how Google Now was working. Google Assistant is available on few mobile devices with Android 7.1 and Google Home devices. Also, it is accessible through the new



(a) Information capture (b) Information miss

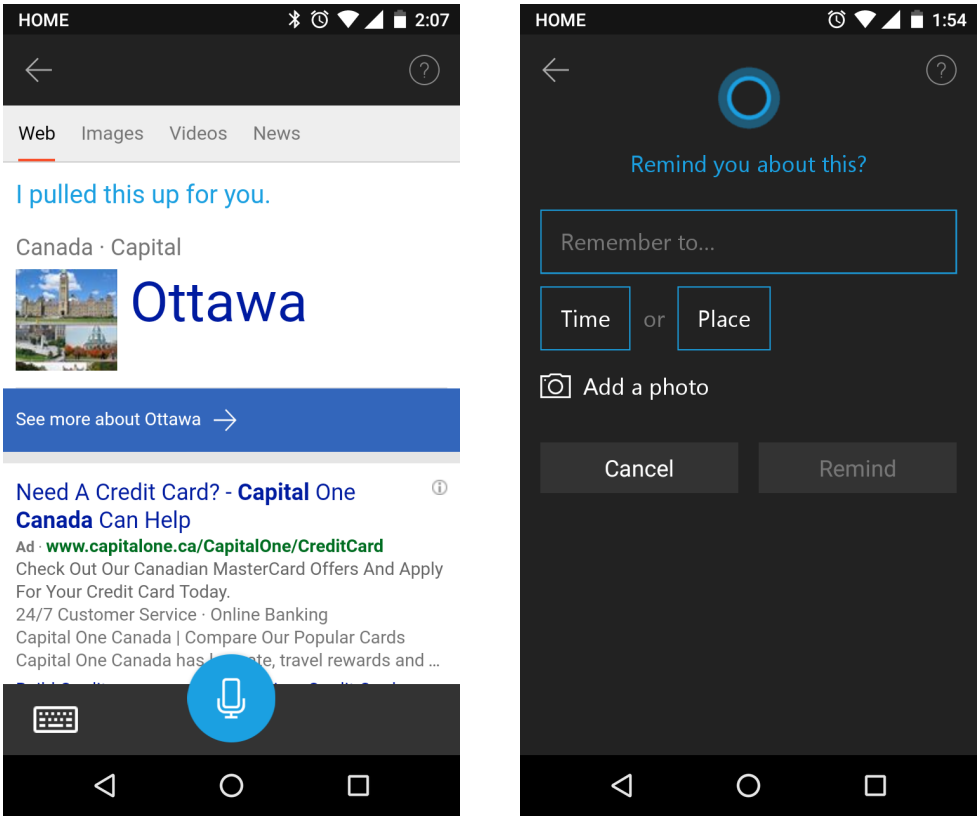
Figure 2.4: Google Assistant on Allo

messaging platform of Google called Allo.

From the tests that we have and articles from other technology news websites (e.g. Gizmodo [22]), the current state of Google Assistant is not very impressive and its functionality is very comparable to what already is available on the market. You can see screenshots of Google Assistant on Allo messaging platform in Figure 2.4 and how it can capture some information from the on-going conversation.

2.3.2.3 Microsoft Cortana

Cortana is designed by Microsoft to compete with Google and Apple in the intelligent assistant market. It was initially released on Microsoft Windows Phone 8.1 but later expanded to other mobile platforms and to desktop computers with the release of Windows 10. The functionality of Microsoft Cortana is very similar to Apple Siri and Google Now. It utilizes personal information,



(a) Qestion Answering (b) Setting a reminder

Figure 2.5: Microsoft Cortana on Android

general knowledge base and a predefined set of trigger words to process users' input. The most noticeable differences of Cortana compared to its rivals are the Notebook feature for storing current conversations and searches and extensive method of reminders not only based on the time but according to the location and the person in a currently ongoing conversation.

Microsoft borrows some of the technologies for its speech conversion and natural language processing from Tellme Networks which was acquired in 2007 and some of them are developed with the help of the company "[24]7 Inc." in partnership with Microsoft. To answer questions, Microsoft uses their knowledge base which is developed by Bing team called Satori. Although Microsoft tries to promote Cortana by making it available in as many markets that they can, there are not many differences in Cortana compared to the competition to convince people to make the switch. Also like Siri and Google Now, Cortana just gives the answer to the current request with no follow-up questions to

clarify the query in case of having disambiguations.

2.3.2.4 Amazon Alexa

Alexa is the smart agent that Amazon has integrated with Amazon Echo. The main method of interaction with this device is through speech. Amazon Alexa, like other smart agents, is triggered by some pre-defined keywords.

Alexa can do all the tasks that are usually done by other smart agents and it has an API allowing third parties to add other functionalities to Alexa. The current default functions of Alexa include weather forecast from AccuWeather, news from a variety of sources, playing music from online radios and streaming services, connecting to smart home equipment to control them with speech, accessing Wikipedia articles and Google calendar and much more.

The text to speech engine of Amazon Echo has an exceptional accuracy and natural lifelike voice compared to its counterparts due to considering natural language processing algorithms in the speech unit selection process.

Although Amazon Alexa has an advanced technology, many of the Artificial Intelligence components of the system are not built by Amazon. Many of these technologies come from the acquisitions of Yap, IVONA Software and Evi (formerly known as True Knowledge).

2.3.2.5 Microsoft XiaoIce

XiaoIce does not exactly fit in a group with the other smart agents. It is also not only a chatbot. XiaoIce is a conversation agent designed by Microsoft Asia. The current system can converse in Mandarin but there is also a Japanese version called Rinna. Microsoft is planning to release an English version of this agent. XiaoIce, like Microsoft Cortana, uses the knowledge base created by Bing search engine to do reasoning. Currently, the knowledge base of Bing contains 1 billion data entries with 21 billion relations among them.

What makes XiaoIce exceptional is its personality. XiaoIce acts like a 17-year-old girl in conversations. The conversations are usually unpredictable. However, there is no indication of the reason for their decision to create this specific persona. XiaoIce shows compassion, sympathy, and care. The system

constantly analyzes the emotional state of the conversation and adapts its responses with it. Like a 17-year-old girl, XiaoIce can get impatient, moody or angry. The emotional analysis in XiaoIce is not limited to a conversation. XiaoIce remembers the previous conversation and asks questions related to that. If you say you are sad, it may ask questions about how you are feeling the next day.

XiaoIce does not only respond to texts. It can also analyze pictures and audios as well and detect its context. This detection is not like other cognitive systems which respond with what it sees or hears; instead, XiaoIce adds a comment or opinion about what it encounters.

XiaoIce has a huge user base which makes it hard to tailor personalized conversations for each user. On the other hand, it is most likely that the agent had a very similar conversation with another user. XiaoIce uses the ongoing conversations to build its response system and learns from ongoing conversations to generate more appropriate and better responses.

There are many articles written about XiaoIce and how it works. Particularly, the New York Times ran an article [53] with an example of a conversation that is done in Chinese with translation to English [57]. This conversation highlights some of the details that we have described here.

To measure the performance of XiaoIce, the developers came up with the idea of Conversation Per Session (CPS). CPS indicates the average number of turns in a conversation with the agent. This number can be an indicator of how well the agent is performing since the user continues the conversation. In conventional smart agents that are available on smartphones, this number is between 1.5 and 2.5. XiaoIce, after a period of time, reached the CPS of 23 which is impressive compared to other smart agents and even the most advanced chatterbots.

In 2015, XiaoIce had 22 million registered users and an average user had interactions with the system 60 times a month. Rinna (Japanese edition of XiaoIce) was released on Line social platform and it currently has 2.2 million followers.

Due to the huge number of users, sometimes the XiaoIce project is referred

to as the largest Turing test in history, but the most interesting part is that against the common perception of the general public, the users do not care that they are talking to a machine and they are mostly happy with the conversations that they are having with XiaoIce [88].

2.4 Natural Language Processing Tools

Natural language is the way that humans use language to communicate with each other. Since computers are not able to understand natural language, there is a processing step involved to make the natural language usable in computerized systems. In Section 2.4.1, we will give a better definition of natural language processing in the context of computer science. In Section 2.4.2, we will review the different processes that are usually done on texts to derive various information from them and in Section 2.4.3, we will look at some of the tools that are developed to help with the task of natural language processing in the form of text.

2.4.1 Definition of Natural Language Processing

Natural Language Processing (NLP) is mainly concerned with the interactions of humans and computers in the form of natural language. NLP is considered an interdisciplinary field of research between computer science and linguistics. NLP research is also intertwined with artificial intelligence and computational linguistics.

The idea of NLP is as old as the idea of artificial intelligence itself. When Alan Turing talked about his idea of intelligent systems, the way he described them and how they should be tested was in a way that made understanding of natural language an essential part of them. However, the designed test limited the understanding capability of these intelligent machines to text format. Today, natural language processing includes written language as well as images and speech.

One of the first applications of NLP was the methods that have been used by Chatterbots to interact with users. Machine translation was also one of the

first tries at NLP. However, the attempts at developing systems with natural language abilities were not successful at the beginning. NLP was resurrected with the emergence of Machine Learning, which became successful because of the increasing computational power in computer systems.

NLP includes many tasks depending on the input method. In the case of having images, Optical Character Recognition (OCR) converts images with written words on them to text. For the speech, speech recognition is one of the important tasks that is being worked on continuously. Natural language understanding (NLU) and natural language generation (NLG) are more general tasks of NLP. Each one of these tasks has subtasks within them which are necessary to handle the tasks.

2.4.2 Natural Language Processing on Textual Content

Many of the tasks in NLP are designed to work on textual content. Search engines, chatterbots, smart assistants and many more applications depend on the correct result of NLP tasks. One of the tasks that is necessary for all of these applications is natural language understanding. There are many subtasks for natural language understanding but some of the most important ones are:

- **Parsing:** To understand each sentence, the grammatical structure of a sentence can be quite helpful. Grammatical parsing creates a parse tree of a given sentence that can be used to determine the meaning of the sentence. The difficulty of parsing is in the ambiguities that can occur in natural language which sometimes can be unclear to a human too.

- **Part of speech tagging:** With part of speech tagging, the function of each word in the sentence can be determined. In the English language, some words can be used in sentences with different parts of speech. For example, "book" can be considered a noun or a verb depending on how it is used in a sentence. This information can be helpful in understanding the meaning of a written sentence.

- **Word sense disambiguation:** For the words with more than one meaning, this subtask tries to find the most relevant word sense according to the context of the sentence or text.

- **Information extraction:** In this subtask, the goal is to extract semantic data from text. IE includes many subtasks within itself. Named entity recognition, relationship extraction and coreference resolution are some of the information extraction subtasks of IE that are listed here.

- **Named entity recognition:** The process for extracting proper names, such as people, places and organizations, is called named entity recognition. This process can be easier in the languages with capitalization for proper names. The goal of NER is not only detecting named entities but also classifying them to see in which category they belong.

- **Relation extraction:** This subtask deals with extracting relations between named entities or nouns from a chunk of text.

- **Coreference resolution:** This subtask tries to connect different words to their related entities. Pronoun resolution is a form of coreference resolution.

Other useful tasks that are usually done on the text are automatic summarization, machine translation, natural language generation, sentiment analysis and information retrieval.

2.4.3 Notable Natural Language Processing Tools

Since natural language processing tasks does not differ significantly from one application to another, there are a set of tools designed to help developers design their applications without worrying about many technical details that are involved in different tasks of NLP.

There are three famous general NLP tool sets that are open and widely adopted by software developers. Developers choose the most suitable toolkit according to their programming language, the inclusion of necessary tasks in that toolkit, etc.

- **Stanford CoreNLP** is a set of tools developed by the Stanford Natural Language Processing Group to help with tasks of NLP. CoreNLP includes stemming, tokenization, part of speech tagging, named entity recognition, parsing, dependency parsing, sentiment analysis and many other tasks. CoreNLP is written in Java and compared to other tools in this category, can be considered as a faster option. It supports languages other than English and also has models for these languages. It also has tools to create your customized models for each one of the tasks. The methods used in CoreNLP are mostly state-of-the-art and have high accuracy.

- **NLTK** or Natural Language Toolkit is a set of libraries and its development is community driven. The project was started at the University of Pennsylvania as a software tool to help teach NLP to students. NLTK is actively being used in research and commercial projects. NLTK is written in Python and, compared to Stanford CoreNLP, is slower. It supports tasks like stemming, tokenization, part of speech tagging and semantic analysis. Like CoreNLP, many tools in NLTK are state-of-the-art because of the active development and community contribution.

- **Apache OpenNLP** is also a set of tools for NLP that is being developed with the support of Apache Software Foundation but it is not actively developed and maintained like its counterparts. OpenNLP is also written in Java and it is faster than NLTK; but in contrast to NLTK, some of the components and NLP methods in OpenNLP are dated. It also supports the main tasks that are necessary for natural language understanding.

There are other task specific NLP tools that may outperform these toolkits (e.g. Google Parsey McParseface) but they do not have the convenience of having all the necessary components within the same package. Moreover, using different NLP tools for each task may affect performance of the final product or may be heavy on the needed resources for NLP processing.

Chapter 3

Speech Act Recognition

Speech Acts are defined in linguistics as vocal acts that help identify the goal and intention of the speaker in the sentences. There are many studies on the different categorization of speech acts. Speech acts also have various applications in computer science. However, unlike linguistics, the goal of speech act recognition is to help machines understand what the speaker or writer means in order to create an appropriate response. In response generation in computers, some of the categories defined in the linguistics may seem unnecessary since speech acts are not the only tool that is being used by computers in order to understand natural language. In Section 3.1, we will review some of the notions in linguistics that define and categorize speech acts. In Section 3.2, we will learn about some of the works that are done in computer systems for identifying speech acts in different contexts. In Section 3.3, how speech act recognition works in our proposed model will be discussed, and finally, in Section 3.4, we will conclude this chapter with our findings and possible future directions to continue this work.

3.1 Speech Acts

The beginning of Speech Act Theory can be traced back to the lectures of John Langshaw "J. L." Austin in Harvard University as part of the William James Lectures Series. Austin's lectures were published later by the university under the title "How to do things with words" [10]. Austin's perspective on how sentences convey meaning was a very novel idea at the time where many

language philosophers were trying to simplify the complexities in the meaning of sentences. According to Austin, people try to do things with their words and statements. His idea of actions by utterances was the cornerstone of Speech Act Theory which was refined later on by his students and other philosophers and linguists.

3.1.1 Austin's Theory

According to J. L. Austin, Speech Acts can be considered in three different levels: Locutionary Acts, Illocutionary Acts, and Perlocutionary Acts. Locutionary Acts are the acts of saying something (Utterances). Illocutionary acts, however, are the acts that are performed by saying something like asking, answering or giving information on a topic. Perlocutionary acts are further effects of a statement like persuading or convincing someone to do something, whether this effect is intended or not. For example, let us look at three different levels of speech act in the sentence "It is snowing". Locutionary act is the utterance of this sentence. Illocutionary act is making this statement with the intention of providing information or the declaration that I make. Perlocutionary act can be the effects that this sentence can have on the audience. This effect can vary based on the context of situation like looking outside of the window when you are home or even leaving work earlier because of this declaration.

Locutionary acts were studied in linguistics before Austin's theory. Also, perlocutionary acts can be unpredictable since there are many factors that can contribute to the outcome of them. Illocutionary acts, on the other hand, were a new idea and can be considered as the main concept in speech act theory. Illocutionary acts, based on Austin's theory, consist of two parts, illocutionary forces and propositional content. Illocutionary forces specify the type of action and propositional content specifies the details of the action. Based on the illocutionary forces, Austin came up with 5 categories for illocutionary acts: Verdictives (Giving a verdict about something), Excretives (Exercise a right, e.g. appointing), Commissives (Commit the speaker to do something), Behavatives (Have something to do with attitudes and behaviours, e.g. apolo-

gizing) and Expositives (Have something to do with how an utterance fits into a conversation or argument, e.g. clarifying).

3.1.2 Searle's Improvements

John Searle, one of Austin's students, extended and refined his work on speech act theory [69]. According to Searle, two parts of illocutionary actions that were defined by Austin, are not enough for categorization of illocutionary acts. He added rules for performance of illocutionary acts and with new definitions, changed the categories of speech acts:

- **Representatives:** In this type of act, the speaker asserts the truth of a proposition. Some of the verbs that can convey such actions are affirm, believe and conclude.
- **Directives:** These acts try to make a listener do something. This can be achieved by having verbs like request, dare or ask.
- **Commisives:** By these actions, a speaker commits to doing something in the future. Pledge, promise and swear are some of the verbs that can carry a commisive act.
- **Expressives:** Speaker expresses his or her attitude toward something by this kind of act. Verbs like congratulate, appreciate and regret can express some of the attitudes.
- **Declarations:** Speaker alters the external condition or status of an object or situation by the utterance of a statement with this type of act e.g. "I now pronounce you man and wife".

The work on speech act theory continued with other concepts like indirect speech acts [68] which was proposed by Searle himself and interdisciplinary ideas like dialog acts [79] which were proposed to be used in automatic conversation agents.

3.2 Speech Act Detection

The beginning of Artificial Intelligence was marked by the idea of creating machines that can be as smart as humans. The criteria of knowing if a machine is intelligent or not were defined by different metrics. One of the famous metrics for evaluating different systems was Turing test. Turing test directly relies on the capability of an intelligent system in responding to different statements that it encounters and how much that system can mimic human behaviour, facing a similar statement. Because of this criterion for identifying an intelligent system, many methods were devised to understand the input which is in the form of natural language. One of the methods that was used at the time was speech act identification.

The theoretical foundation of speech acts, which was built by the 1960s, gave researchers the idea of using that theory in creating their intelligent systems. At the time, people who were involved in Linguistics also helped computer scientists in creating AI. Even John Searle himself helped with some of the theories in the AI field. However, the computation power at the time was not enough for statistical processing that is necessary for identifying speech acts in sentences.

The first attempts for speech act identification by computer systems were initiated in the 1990s with the help of Markov models. But the idea of initial speech act categories that was proposed by linguists was not that useful for computer scientists and their intended applications. Stolcke et al. [79] were one of the first groups dealing with speech acts in computer systems with their own definition of speech acts. They tagged the switchboard telephone dataset. Their tagging categories were called dialog acts and it included 42 different labels. They have used hidden Markov models in predicting the dialog act of a sentence by looking back in the conversation with the known dialog acts.

Other works that try to identify speech acts are mostly done on textual data from emails, message boards, Internet forums and twitter. The work presented in [63] tries to classify message board posts into four speech act categories with the help of support vector machines. The acts that they have

used are commissive, directive, expressive and representative. They have a set of hand-crafted features for the classification task. Cong et al. [21] try to find questions and answers in the forum posts. They have used labelled sequential patterns in the text that they have created from POS tags and words in the sentences as features.

There are also works that are trying to find speech acts in emails. Cohen et al. [19] attempt to classify emails into four classes of requests, commitments, proposals and reminders. They have used Ngrams and POS phrases to classify them with SVM. The authors of [43] also try to do the same thing with some small modifications. Shrestha and McKeown [75] also worked on emails but instead of four classes, they classified them into questions and answers. POS bi-grams, beginning Ngrams and ending Ngrams are their features.

Li et al. [46] try to identify questions in tweets. Their work was more challenging since misspellings are frequent on twitter and they needed to handle this noise in their data. They extracted frequent subsequences in the text with the help of Prefix Span Algorithm. They have used supervised and unsupervised learning methods with various features to find tweets with questions.

The works that we have listed here and many other works that try to tackle speech act detection are not that comparable with each other since they are working on very different data with different objective. Even the acts that they try to classify differ from one work to another. The model that we are presenting in the next section is a gradual improvement from the work of Quinn and Zaiane [64], which has the same objective as ours.

3.3 Our Model

In our model, we are trying to identify speech acts of each given sentence to our system. Like other works that we have discussed in the previous section, the original categories that were defined by Searle are not appropriate for our application. Our goal of identifying speech acts is to help smart agents, that interact with people through natural language, get a better perspective of what each sentence conveys. Therefore, instead of those five categories we are using

the following three categories:

1. **Declarative:** This type of sentence is used for expressing different opinions and feelings, and explaining what is happening. In general, all the sentences that have Representatives, Declarations, Expressives and Commisive speech acts are considered Declarative.

2. **Interrogative:** Questions are the main type of sentences that we are trying to identify in this speech act category. However, some of the requests and commands can be expressed as questions. In the current categorization, we consider them as an interrogative sentence as well. Questions fit in the Directives category of speech acts but not all the sentences in the Directive category can be considered interrogative.

3. **Imperative:** These sentences are used for requesting or commanding someone or something to do or act based on the contents of the sentence. Imperatives also fit in the directive category of Searle's categorization of speech acts.

Our model, unlike other related models that we have discussed in the previous section, deals with text without punctuation. Many of the speech act identification tasks work with textual data, which usually comes with relevant punctuation. Punctuation can help with precise identification of questions in most of the cases. If the punctuation is missing from the text, some of these methods may perform poorly.

Another issue that systems that work with textual data may face is misspellings. In smart agents that deal with natural language, a preferable method of communication between users and agents is speech. Since speech to text usually does not contain misspellings, we can assume, in the sentences that we will have as input in our model, we will not have any misspellings.

Our task can be considered a classification problem. In Section 3.3.1, we will define our feature set for classification task and the classifier that we have used for this task. In Section 3.3.2, we will analyze the dataset for the training of our model and the results that we have obtained from our experiments.

3.3.1 Feature Set and Classification

In speech act identification tasks, there is a set of features that are usually being used for classification. Since our model relies on the output from speech to text engines to get the textual content of each statement, we may miss some of these features that are used by other works. One of these features is punctuation which we do not have in the output of speech to text engines. However, there are other features like presence of specific words, POS tagging and Ngrams that we can also get from our dataset. This set of features is based on the previous work of Quinn and Zaiane on the same task. The set of features that we have in our classification task includes:

- **Binary Features:** There are some specific words that can help in identifying speech acts. The binary features that we have used show the presence of such words or conditions in the sentence. The binary features that we have tested are: verb being the first word of a sentence, having a wh-word in the beginning of the sentence, the presence of question words, interrogative tag phrases and common declarative phrases.

- **POS Ngrams:** We have used a set of POS Ngrams that each sentence will be checked against. The reason why this can be a helpful feature is that many sentences with a certain speech act may share the same grammatical structure. For example, in many requests, the pattern of "VB + DT + NN" (e.g. Open the door) can be seen repeatedly. To create a set of discriminative POS Ngrams we went through the dataset and with the help of part of speech tagging, we made a list of all possible Ngrams with a high frequency of appearance. According to the work of Quinn and Zaiane [64], the most effective Ngrams that can be used are bigrams and trigrams. Also, based on their recommendation, all the Ngrams that appeared more than 3 times are added to the POS Ngrams list.

- **Word Ngrams:** In addition to POS Ngrams, word Ngrams can also be helpful in identifying speech acts. There may be a sequence of words that

appear several times in one of the categories of speech acts. Identifying the most common sequences in the dataset and using them as a feature in our classification task can be beneficial.

- **Word Clusters:** According to the previous works of [82], word clusters can help in classification acts. Usually, the words that belong in the same cluster, share similar semantics. Quinn and Zaiane showed that using such a cluster can be beneficial in speech act classification tasks [64]. First, we find the cluster of each word in the sentence from a 1000 class cluster; this cluster showed better results compared to the others in [64]. We use all the clusters of a sentence as a feature.

Another feature that could have been helpful is dependency links and constituency parsing but from the results of the work done by Quinn and Zaiane [64], they did not improve the results. Therefore, we will skip these two features as well.

For the classification, we are using SVMlight library [39] with the Java support library from [81]. The kernel in our classification task is linear. For each class, we created an SVM classifier and the class of speech act is then chosen by the highest objective function value.

3.3.2 Experiments

In order to make the results comparable to the ones from the previous work done by Quinn and Zaiane, we tried to use the same data with similar criteria in our experiments. In Section 3.3.2.1, We are going to discuss how the data is collected and what are its characteristics. In Section 3.3.2.2, we will define the evaluation metrics of our experiment and in Section 3.3.2.3, we will present the results of our experiment.

3.3.2.1 Data

We are using the same data from [64]. This data includes synthetic data that were tailored to the specifications of a smart agent, getting speech as input. Also, data includes sentences that were extracted from different online sources

like fact or news websites. None of these sentences contains any punctuation related to speech acts. The training set and test set include 1044 and 232 sentences respectively.

3.3.2.2 Evaluation Metrics

For evaluating the performance of our classifiers, we are using precision, recall, and f-score as our criteria. Precision, recall and f-score of each class is calculated separately for each experiment with an overall accuracy, including all the results of all the categories. Overall accuracy is defined as:

$$A = \frac{\sum TP_i + \sum TN_i}{\sum TP_i + \sum TN_i + \sum FP_i + \sum FN_i}$$

It is worth noting that a sentence will be considered as a true negative case for a category, if it does not belong to that category and is not classified as being in it, regardless of the classification output. We have used features in various combinations to show the effectiveness of including them in the final classifier.

3.3.2.3 Results

The results of our experiment are listed in Table 3.1. We have tested each feature individually to see how they perform alone. Then we have combined them in order to find if there is any performance gain in combining them. The combination of POS Ngrams, word Ngrams and brown clusters improves the classification performance. However, by adding binary features to this combination, while we see an increase in performance of interrogative and imperative classifiers and overall accuracy, the performance of the declarative classifier is affected adversely. To mitigate this, we have modified the binary features that we have chosen for classification. From all the binary features, we only used "having verb or wh-word in the beginning of the sentence". The reason for this modification is some of the features could have been in all the three categories of classification and merely knowing that a sentence contains one of those words is not a good indicator for speech act classification. For example, question words like "did", "can" and "will" can be used in both

Table 3.1: Classifiers performance with different feature combinations

P: Precision, R: Recall, F: f-score, A: Overall accuracy

	Declarative			Interrogative			Imperative			
Feature Set	P	R	F	P	R	F	P	R	F	A
POS	64.7	77.5	70.5	71.9	60.5	65.7	85.5	83.5	84.5	82.8
Words	50.0	84.5	62.8	90.2	48.7	63.2	73.2	61.2	66.7	76.1
Cluster	67.6	67.6	67.6	69.1	61.8	65.3	74.2	81.2	77.5	80.5
Bin + POS	65.2	84.5	73.6	80.3	64.4	71.5	91.1	84.7	87.8	85.3
Bin + Words	52.9	90.1	66.6	84.4	50.0	62.8	87.9	68.2	76.8	79.3
Bin + Cluster	61.5	78.8	69.1	74.6	61.8	67.6	85.9	78.8	82.2	82.2
POS + Words	67.8	85.9	75.7	81.0	61.8	70.1	85.7	84.7	85.2	85.1
POS + Cluster	73.1	80.3	76.5	79.7	67.1	72.4	84.4	89.4	86.8	86.2
Bin + POS + Words	67.4	84.5	75.0	79.4	65.8	71.9	88.7	83.5	86.1	85.3
Bin + POS + Cluster	71.9	83.1	77.1	85.1	75.0	79.7	91.6	89.4	90.5	88.5
Words + Cluster	73.2	73.2	73.2	83.9	61.8	71.2	73.3	90.6	81.0	83.9
Bin + Words+ Cluster	62.8	83.1	71.5	80.0	63.2	70.6	88.5	81.2	84.7	83.9
POS + Words + Cluster	78.2	85.9	81.9	85.5	69.7	76.8	84.8	91.8	88.1	88.5
Bin + POS + Words + Cluster	74.4	85.9	79.7	87.3	72.4	79.1	89.6	91.8	90.7	89.1
Mod. Bin + POS + Words + Cluster	75.6	87.3	81.0	89.2	76.3	82.3	90.6	90.6	90.6	89.9

interrogative and declarative sentences. Declarative phrases that were chosen to help with classification also have such words like “okay”, “good” and “right”. The performance of the system is improved compared to the previous work of Quinn and Zaiane [64]. They used POS+Words+Cluster to obtain an overall Accuracy of 82 and the following results: Declarative P=75; R=83; F=78; Interrogative P=88; R=70; F=78; Imperative P=86; R=92; F=89.

3.4 Conclusion and Future Work

In this work, we tried to identify speech acts in sentences to help conversation agents to better understand the users’ inputs. In the experiments and system design, we relied on the contextual content of the input. However, the text of a sentence does not contain all the necessary information that is needed to identify speech acts, especially when there is no punctuation available. The

intonation of the speaker is very important in the utterance of a sentence and the act it conveys. For the future, since our work is designed for conversation agents, the speech itself can be used for intonation analysis to complement the textual classifier.

As it was explained in [64], we are looking into each sentence alone without knowing the prior conversation or sentences. It may be useful to look into utilizing past sentences in the conversation for predicting speech acts.

Chapter 4

Information Extraction

Information Extraction is the process of extracting structured data from unstructured data like text documents to make data usable by machines. Natural language processing is considered to be one of the necessary means for information extraction. A conversational agent, like other systems that deal with natural language, needs information extraction for getting information about the context of the conversation. In this work, we have devised an information extraction method specific to our necessary criteria of knowledge, which is limited by the design of our knowledge representation.

First, we will define information extraction and the tasks involved. Then in Section 4.2, we will examine some of the information extraction methods that are being used in different knowledge domains. In Section 4.3, we will discuss our method of information extraction and in Section 4.4, we will discuss some of the findings that we discovered in the process of devising our current information extraction method. In Section 4.5, we will conclude this chapter with some future directions to the current model.

4.1 Definition

Information extraction (IE) can be defined as the process of identifying specific classes of entities, events and relations from data that is in the form of natural language. It also includes extraction of attributes related to each one of these classes. The extracted information is predefined by the user in the form of templates. Sometimes information extraction is referred to as slot filling since

it is trying to fill different fields in the information templates. Also, because of storing data in a structured way, IE is considered as a database population method. In this section, we will look into classic tasks of IE, the general architecture of IE systems and their evaluation criteria.

4.1.1 Classic Tasks of Information Extraction

Information Extraction usually has four types of tasks:

- **Named Entity Recognition (NER):** The objective of this task is to identify entities such as organizations, persons, place names, temporal expressions, numerical and currency expressions etc. Extracting relevant attributes for entities is also considered part of NER task. Such attributes for a person can be gender, profession, title, nationality, age etc.

- **Coreference Resolution:** This deals with identifying multiple mentions of the same entity in a document. These mentions can be in the form of Named (Referring to International Business Machines as IBM), Pronominal (Referring to John with he), Nominal (Referring to Google as the company) and Implicit (Does not occur in English).

- **Relation Extraction:** This is the task of identifying different relations between entities from a document from a set of predefined relations. Since the relation in a document can be unlimited, this task deals with a fixed set of relations.

- **Event Extraction:** In this task, the goal is identifying different events from a text with structured information about that event. For event extraction, identifying multiple entities and relations is necessary. Event extraction is considered the hardest task among these four classic tasks.

Although these classic tasks deal with single documents, more recent works are trying to cross reference such information from multiple documents.

4.1.2 Architecture of IE Systems

The design of an IE system is dependent on the intended domain of knowledge that the system is designed to work with. However, there are common components that can be found in almost every IE system which have the same responsibility. In this regard, the architecture of an IE system can be split into two sections, domain-independent parts and domain-specific parts. The domain-independent operating parts are:

- **Meta-data analysis:** In this step, the document title, body of the document, structure of the body and the date of the document are extracted.
- **Tokenization:** In this part, the text is segmented into word-like units for further processing.
- **Morphological Analysis:** Morphological properties of each token are extracted in this step. Information such as part of speech, potential word form (lemma) and other morphological tags based on POS, such as tense of verbs and mood, are extracted.
- **Sentence/Utterance Boundary Detection:** The text is segmented into a sequence of sentences or utterances for further processing.
- **Common Named Entity Extraction:** In this step, domain- independent named entities are extracted such as temporal expressions, numbers and currencies.
- **Phrase Recognition:** Local structures like noun phrases, verb groups, prepositional phrases etc. are extracted in this step.
- **Syntactic Analysis:** The dependency structure (Parse tree) of a sentence is computed based on sequence of lexical items and small structures e.g. phrases.

These processing steps are done before the main tasks of IE. The main tasks of IE are more domain-specific. In addition to common NER, there is

another domain specific NER which should be done on the text. However, in some systems, these two NERs are combined to make a single component in the system.

4.1.3 IE Systems Evaluation Criteria

The performance of IE systems can be calculated precisely with precision and recall metrics based on the input. When there is a need for a balanced view of the performance, F-measure, which is based on precision and recall, can be used. Another metric which is specifically used with IE systems is Slot Error Rate (SER). SER is defined as:

$$SER = \frac{\#incorrect + \#missing}{\#key}$$

in which, $\#incorrect$ is the number of incorrect system responses for filling slots in the templates and $\#missing$ is the number of slots in templates which are not filled by the system. $\#key$ denotes the number of all slots in the templates that the system is supposed to fill.

4.2 Related Work

One of the reasons of great attention to the topic of Information Retrieval and Extraction from early years of natural language processing is the government-sponsored competitions for designing such systems. These competitions resulted in the fast progress of IE Systems. We will have a quick look at these competitions and their criteria in Section 4.2.1.

Early information extraction systems can be traced back to the 1970s. From a chronological perspective, IE systems over time followed different approaches to system design and functionality. These approaches can be generalized in Knowledge Engineering approach and Trainable IE Systems. We will discuss these approaches and famous IE systems following them in Section 4.2.2 and 4.2.3 respectively.

4.2.1 Information Extraction Competitions

One of the reasons for the advancements and significant interest in information extraction can be traced back to the competitions which began in 1987. The most famous information extraction competition was initiated by Defense Advanced Research Projects Agency (DARPA). It was called Message Understanding Conferences and each competition had an explicit objective for information extraction [33]. These competitions included information retrieval as well. There were seven MUCs in total with topics of naval information messages, terrorism in Latin American countries, joint ventures and microelectronic domains, news articles on management changes, and satellite and missile launch reports. The information extraction in these competitions was in the form of template filling. These templates, over time, became more complex and in the later competitions, the systems had to deal with multilingual documents as well.

After MUCs, in 1999, the National Institute of Science and Technology (NIST) started the Advanced Content Extraction (ACE) program with the similar objective of supporting the development of automatic content extraction technologies for automatic processing of natural language in text from various sources [26]. ACE tasks were more complicated than MUCs and The Linguistic Data Consortium was the developer of annotation guidelines, corpora and the other linguistic resources to support the program.

MUCs and ACE contributed to further development of information extraction systems by making available all the data from the tasks of each competition. Developers of IE systems use this data to evaluate the performance of their systems. In addition to MUCs and ACE, there are other venues with a similar goal. The Conference on Computational Natural Language Learning (CoNLL) organized a competition for language-independent named entity recognition, and the Text Analysis Conference (TAC) has a dedicated track named Knowledge Base Population with its related tasks.

4.2.2 Knowledge Engineering Approach

In the early IE systems, information was detected from the context of the text and extracted by Knowledge Engineering practices. Knowledge Engineering (KE) in this context is considered as the process of creating linguistic knowledge, by the human expert, in the form of rules and patterns for the information extraction, according to the knowledge in the specific domain. KE is done by reviewing corpus to find patterns and also by the intuition of KE designer. Some of the famous IE systems based on this approach are ATRANS System for extracting information from money transfer messages between banks [51], JASPER with the intention of extracting information from corporate earning reports [2], and SCISOR with the goal of extracting facts from online messages about mergers and acquisitions [37]. There are other examples that are done in the domain of security [44, 45].

The problem with these earlier systems was that they were not that easy to port to new languages or to add new scenarios. Finite State Machines helped in overcoming these obstacles to some extent and helped in developing IE systems in the general domain. One of the most famous systems designed with finite-state machines is The FASTUS which works with English and Japanese [34]. The FASTUS is one of the top scoring systems in MUC competitions and it is developed by SRI International. SPPC is another finite-state based IE system that works with German text [58]. Finite-state-based formalisms helped further with this approach. Today, finite-state-based formalisms are being used with advanced general IE systems like GATE [23], SProUT [28] and ExPRESS [62].

In more advanced systems, dependency parsing is also used with KE. As a middle ground, some systems used dependency parsing in combination with more shallow processing methods like rule-based and finite-state based information extraction. One of these systems is LaSIE-II which was one of the top-scoring systems in MUC-7 [35]. IE2 [5] and REES [6] are other examples of such systems. IE2 achieved the highest scores in almost all of the IE tasks in the MUC-7 competition and REES is a system which tries to extract more

than 100 relation types and events from the text in the domains of business, finance and politics.

It is worth noting that despite the fact that the knowledge engineering approach is an old method for information extraction and time-consuming, some of the systems that are designed with this approach have extraordinary performance.

4.2.3 Trainable IE Systems

In the 1990s, there was a trend of modular design for IE systems which allowed the separation of IE engine and the knowledge of the domain that is needed by the system. However, even with the separation of IE engine design in IE systems, the knowledge engineering was very time-consuming. To help with this issue, Machine Learning (ML) techniques were used to reduce the amount of work that is necessary for porting an IE system from one domain to another. One of the approaches with this goal is data annotation instead of knowledge engineering. The annotated data would be used later on with supervised ML methods to train an IE system in a specific domain. Hidden Markov Models is one of the supervised methods that has been used within IE systems and Nymble is a NER system that is designed based on it [13]. Another supervised method which performs better with IE tasks, especially NER, is Conditional Random Fields. CRFs are good at modelling local dependencies [72].

Most state-of-the-art IE systems are using trainable models for different tasks of IE. However, for some of the tasks like tokenization, nothing more than a finite-state machine is necessary.

4.3 Our Method

Information extraction in our work is an essential part of how our system operates. It is necessary to get proper information from the conversation in order to populate the knowledge base, answer questions, address commands and generate appropriate responses. Unlike textual information extraction, we are usually dealing with short sentences which depend on previous sentences to

convey a message or information. Because of this nature of our data, we need to look for other methods than the traditional ones in the literature, especially for tasks like coreference resolution. It is also worth mentioning that similar systems that deal with conversations use their own proprietary methods which are not available to the public.

In our information extraction system, we are using Stanford CoreNLP for the preliminary steps that we have described in general architecture of IE systems. Then, we are using an open information extraction system, named entity recognizer with our specific additions, gender resolver and rule-based information extraction. We are going to look into each one of these secondary tasks separately in the coming sections. It is worth mentioning that the way we deal with information extraction tasks in our agent is more in line with the knowledge engineering approach of IE systems design.

4.3.1 Open Information Extraction

An open information extraction system can help us acquire relations from a sentence. These relations can be helpful to get an understanding of a sentence, specifically when we want to get information about an entity. The Open IE system that we are using is developed by the Stanford natural language processing group based on the work of Angeli et al. [4]. This Open IE system extracts relation tuples from plain text. You can see an example of Stanford Open IE output in Figure 4.1.

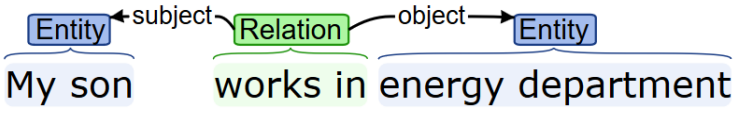


Figure 4.1: An example of a triple extracted by Stanford Open IE

However, as Open IE systems seem to be the best way to deal with information extraction tasks, the way that they are designed and the schemes that they are using may cause losing some information in the process, which is more important in the context of the conversation. In order to compensate for such situations, especially in the cases where the information necessary to populate

our knowledge base may get ignored, we need to have other mechanisms as well.

4.3.2 Named Entity Recognition

Named entities can help with extracting much of the information that we need to populate the knowledge bases. Named entities can refer to a person, location, organization etc. In our work, we are using Stanford Named Entity Recognizer which is implemented using linear chain Conditional Random Fields (CRF) models [29]. Stanford NER provides seven classes of named entities. These classes are Location, Person, Organization, Money, Percent, Date and Time. Using the results of Stanford NER can help us with getting information like event date, time and place.

In addition to NER, we are using Regular Expression Named Entity Recognizer (RegexNER). Stanford RegexNER is a pattern-based named entity recognizer that can use a set of defined patterns for named entities to check within the sentences. We added professions, relationships, drugs, symptoms and other information that can be checked with pattern matching to RegexNER patterns. Some examples of each class that we have added to RegexNER can be seen in Table 4.1. Stanford NER and RegexNER create a unified output for sentences. They specify classes for tokens in a sentence. Listing 4.1 shows an example of a unified output of Stanford NER and RegexNER.

Listing 4.1: Output sample of Stanford NER

```
[My, grandson, Johnny, is, 15]
[O, RELATION, PERSON, O, NUMBER]
```

Table 4.1: Added entities to Stanford RegexNER

Entity Type	Attribute	Examples of Keywords and Tag Phrases
PERSON	PROFESSION	Student, Professor, Doctor, Accountant, etc.
PERSON	POSSESSION	Car, House, Mansion, Boat, etc.
PERSON	RELATION	Son, Daughter, Niece, Friend, Neighbour, etc.
MEDICAL	FEELING	Uncomfortable, Pain, Sore, Sick, etc.
MEDICAL	SYMPTOM	Fever, Rash, Headache, Blurry visioin, etc.
DRUGS	DNAME	Tylenol, Vicodin, Lexapro, etc.

4.3.3 Gender Resolving

Knowing people's names in the conversation is not enough for connecting different information especially multiple sentences that use pronouns to refer to different people (coreference resolving). We need to determine the gender of a person to know which pronoun is referring to whom.

We have devised a 3-step method for determining the gender of a person. First, we look into the relationship of the person since some of these relations specify gender like sister, brother or son. If the relationship is not specified or it is not specific to a gender (e.g. cousin), we will look into pronouns that can refer to the same person in the same sentence. If we could not get gender information from relationship or pronouns, we will check the name with Genderize.io to get the gender of the name. Genderize.io supports a wide range of names in 89 languages which makes it a better choice than self-created databases.

4.3.4 Rule-Based Information Extraction

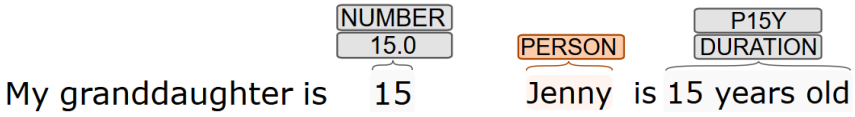
We have rules for extracting each specific class of entity and related attributes. Extracting some of these entities and attributes are simple such as extracting 'Person' entity using a general NER while some of them are more complicated. In this section, we will look into two relatively complicated attribute extraction rules for extracting age and relationship for 'Person' entity. Other rules follow very similar patterns as these two rules.

4.3.4.1 Rules for Extracting Age

Finding age from a sentence can seem a trivial task, however, there are many ways that age can be stated in a sentence. In order to get the age from a sentence, we need to consider different ways that it can be expressed.

For extracting age from a sentence, first, we need to check if there is a number that can possibly express the age of someone in the conversation. NER detects mentions of the age with two classes of NUMBER and DURATION (When age is stated as "27 years old"). Figure 4.2 shows the examples of NER

output when a sentence contains information about age. Words categorized into these two classes have the potential of referring to the age of someone but it can refer to other things rather than age. To make sure that the sentence contains information about the age we do the following checks:



(a) Number class (b) Duration class

Figure 4.2: NER output for sentences with age mention

1. Is the number the only word in the utterance? If it is, did the system ask about someone's age?

2. Is the number in the object of the sentence? If it is, does the object refer to the number? (Figure 4.2)

3. Is the number in the subject of the sentence? If it is, does the subject refer to a person? (Figure 4.3)

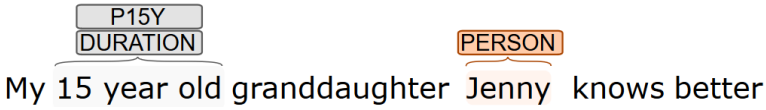


Figure 4.3: A sentence with age mentioned in the subject

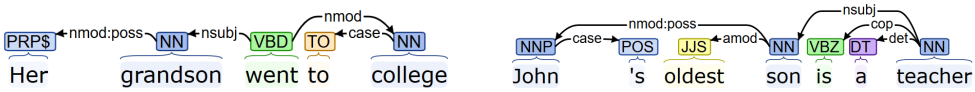
If the answers to a pair of these questions are yes, then the number is considered as the age of the mentioned person. If the person is mentioned with a pronoun in the sentence, CKB can help with finding the entity related to that pronoun for updating PKB.

4.3.4.2 Rules for Extracting Relationship

We extract relationships from the sentences to connect the persons who the user has mentioned before in order to know to who the user refers to when he or she uses relationships only to indicate a person. To extract relationships, we have added keywords to RegexNER. If a relationship is mentioned in a

sentence, the unified output of NER will tag the referring word as RELATION class.

With using NER output, we can detect the mention of a relationship, but it does not give us any information about who the person is related to. In talking about relationships, there is usually a possessive expression which gives this information to the listener. To get this information from our IE system, we check the result of dependency parsing of the sentence. Dependency parsing usually can show the dependency between relationship and the possessive expression. In Figure 4.4, The dependency parsing is shown for two sentences. However, this representation is not easy to use in IE systems. Stanford Parser has a specific representation called Universal Dependencies (Shown in Listing 4.2 and 4.3) which can help in identifying links easier. Universal Dependencies are in the form of pairs with the type of dependency.



(a) Sentence with possessive pronoun (b) Sentence with possessive expression

Figure 4.4: Dependency tree of two sentences with relationship mention

Listing 4.2: Universal dependencies for sentence in Figure 4.4a

```
nmod:poss(grandson-2, Her-1)
nsubj(went-3, grandson-2)
root(ROOT-0, went-3)
case(college-5, to-4)
nmod(went-3, college-5)
```

Listing 4.3: Universal dependencies for sentence in Figure 4.4b

```
nmod:poss(son-4, John-1)
case(John-1, 's-2)
amod(son-4, oldest-3)
nsubj(teacher-7, son-4)
cop(teacher-7, is-5)
det(teacher-7, a-6)
root(ROOT-0, teacher-7)
```

Our IE system references relationships on the pairs that we get from dependency parsing. If there is a name or possessive pronoun referring to the

relationship, there will be a dependency pair with the type of "nmod:poss" which links the pronoun or name to the relationship. In some cases, the parser may connect possessive expression and relationship in two steps (Figure 4.5 and Listing 4.4).

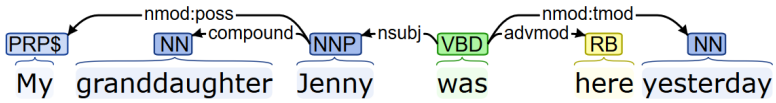


Figure 4.5: A sentence with relationship mentioned in the subject

Listing 4.4: Universal dependencies for sentence in Figure 4.5

```
nmod:poss(Jenny-3, My-1)
compound(Jenny-3, granddaughter-2)
nsubj(yesterday-6, Jenny-3)
cop(yesterday-6, was-4)
advmod(yesterday-6, here-5)
root(ROOT-0, yesterday-6)
```

4.4 Discussion

The system that we have described here is specifically designed to work with our conversational agent which is paired with the knowledge representation that we will describe in Chapter 5. In order to evaluate the performance of our system, we need a dataset similar to conversations that we would expect, and the sentences should be tagged with the template of our knowledge representation. We do not have a dataset with that specifications to have a proper evaluation of our system.

Checking the performance of our system with the datasets of MUCs or TACs does not seem relevant since their templates vary from what we are extracting. Also, the parts our IE system can fill from their templates are mostly done by Stanford CoreNLP, which has state-of-the-art performance. This kind of evaluation will not help us in identifying the shortcomings of our system but the performance issues of CoreNLP package. However, as an alternative method for the evaluation of the extracted relationships, a method similar to the work of Celikyilmaz et al. can be used [18].

The best way to do an evaluation of how our IE system works is to have the system operate for some time to gather data from the users, then have that data annotated according to our template and then compare the result of our IE system with the annotations of data. This way, since we are designing the IE system intuitively, we may be able to find patterns that we did not anticipate in the initial design of our system.

4.5 Conclusion and Future Work

In this chapter, we reviewed concepts of information extraction and IE systems. We also looked at the different approaches for designing them, and then we represented our own model. We have used intuition for designing extraction rules for most of our entities and attributes. However, it is better to use a dataset for finding the patterns that can help with information extraction. As a future work, after gathering enough data with our system, we can use that data to create better rules for information extraction which can help with better populating of our knowledge bases.

As another direction for the IE, we can use trainable models instead of knowledge engineering. By using trainable models, we can add support for new languages with less modifications to the core functionalities of our IE method.

Chapter 5

Knowledge Representation

Knowledge representation (KR) can be considered a way of representing information in a machine-readable format. It can help machines understand what is going on around them by viewing the information, drawing conclusions and relating different information from representation. Even though we try to come up with a general definition for Knowledge Representation, it is usually defined in the boundaries of a system and the role that KR will play in its functionality.

In this chapter, we are trying to define KR for a conversational agent that is designed to interact with the elderly and assist them. Since the current implementation of the system is a proof-of-concept, the KR that we have designed may have some limitations.

In order to have a better understanding of KR, we are going to look into different roles and definitions of KR in Section 5.1. In Section 5.2, we will review some of the related works that tried to add KR to conversational agents and chatbots, and in Section 5.3, we will define the role of KR in our system and how we want to implement it. In Section 5.4, we present our conclusion in implementing the intended KR for our system.

5.1 Definition

According to Davis et al., Knowledge Representation should be defined according to the role it is playing in an intelligent system [25]. They considered five different roles for KR. According to them, KR is:

1. **A Surrogate:** In this role, KR tries to replicate the real world internally for an agent. The reason is most of the things that agents need to reason about exist externally. When an intelligent entity (e.g. a person) tries to think about creating something, it thinks about the parts that physically exist but instead of having them, he or she knows the characteristics of each part, hence the expected outcome of combining them. For an agent, to do a similar thing, it needs to have such an understanding of the outside world. In a simpler form, with the help of KR, the agent can know the consequences of an action by thinking about it, instead of acting. Although having a perfect surrogate of the world is the best way to understand it, it is unachievable.

2. **A Set of Ontological Commitments:** In this role, KR is believed to be imperfect, hence we have a set of realities that we will include in KR while ignoring others. For designing KR in this role, these limitations should be decided in the design process. KRs in this role are usually domain specific.

3. **A Fragmentary Theory of Intelligent Reasoning:** In this role, KR is designed to specifically help with intelligent reasoning processes. The reasoning decides what should be included in the KR, and KR is just a small part of the complex reasoning process.

4. **A Medium for Efficient Computation:** In this role, KR is a means of computation. Reasoning in machines is considered as a computational process. Since KR is used in the reasoning process, it affects the computation efficiency. Because of this, many methods of KR with the intention of computation have explicit methods of defining knowledge (e.g. Frames).

5. **A Medium of Human Expression:** In this role, KR is a tool that we use for expressing the world to the machines. This role of KR is necessary if we want to tell the machine about the world. In this role, KR is a tool for us in order to communicate with machines.

These roles and their definitions share some characteristics but at the end, what matters is our expectation of KR and what we want to do with it. In our case, KR have characteristics of the second, fourth and fifth roles more that the other two.

5.2 Related Work

We have looked into similar systems that are trying to use KR to their advantage. However, most of these systems are creating or utilizing a general knowledge base that they plan to incorporate within the conversations with all the users. In our case, in addition to a general knowledge base, we will build a knowledge base for each user with the information extracted from conversations. Personalized Knowledge Base is not just for creating responses but for personalized and tailored ones.

In the first chatbots, there were not any elements of knowledge but as the technology and AI improved, more chatbots utilized knowledge bases and knowledge representation in their response generation process. One of the chatbots that used knowledge representation was ALICE [87]. Although AIML is considered as a markup language for pattern matching and response generation, the AIML templates that were used within the ALICE represents the knowledge of the chatbot in the form of question and answer pairs. CSIEC is a chatbot designed for English learners which utilizes NLML as knowledge, which is very similar to AIML in ALICE [38].

Based on ALICE, other chatbots were created by merging the capabilities of AIML with Cyc ontology [55, 61]. OntBot is another chatbot which tries to benefit from ontologies [1]. Also, there is an architectural guide for developing chatbots with reasoning capabilities from ontologies [9].

There are cases of using semantic web for creating responses for chatbots and conversational agents related to the questions that fit into general knowledge domain [8]. Pilato et al. suggested a system design that would have different knowledge sources as modules in its KR and according to the domain and user behaviour, would select the most appropriate knowledge base [60].

5.3 Our Representation

Knowledge representation definition in our work is a bit different from the general definition. KR in our work is going to be used in a conversational agent that will be focused on the elderly. We do not need our system to understand every concept or perform complex inferences on the stored information. The knowledge domain that we want to use in our KR is limited to the topics that may come up more often than the others in the conversation. Also, for general knowledge or public knowledge, there are other knowledge bases with their own inference engines that can get integrated into our system. These simplifications in our knowledge representation help us in developing our agent faster without worrying too much about handling the information in the conversations.

Three different components can be considered for our knowledge representation. We have a Personalized Knowledge Base (PKB), a Context Knowledge Base (CKB) and a simple Inference engine. PKB is a knowledge base that stores all the extracted information from the conversations for future uses like response generation. CKB is a temporary knowledge base for on-going conversations, and Inference Engine is used for simple inferences that may be deemed necessary for having appropriate responses. We are going to look into each one of these components separately and how they are implemented.

5.3.1 Personalized Knowledge Base

The knowledge that is most frequently needed in a conversation with the elderly is usually centred around personal issues, family relations and events involving them. It is crucial to include such information in the responses that are generated by the agent.

The extracted information from conversations is stored in a structured sense that can be directly used by the system for question answering and response generation. This information can be divided into four categories:

- **Personal and Health Information:** We are trying to keep all the information about the elderly that can be useful in the future, not only for response generation but also for health and well-being related issues.

This information can be about what they like or dislike, what they have done recently, how they are feeling, the medication intake and other information.

- **People's Information:** In conversations with the elderly, they often talk about other people and family members. A human listener would remember some of these names and their information. It is necessary that we try to do the same with the help of our PKB. We keep people's names, how they are related to the elderly, how old they are and some other information that can come up in the course of a conversation.

- **Event Information:** Remembering the past or coming events can be also helpful in a conversation, especially for coming events where an agent can remind a person of when it is going to be. We store information like the event location, people who are attending, and time and date of the events.

- **Information triples:** We are storing the information that is extracted from the conversation which may not fit within the defined attributes. This information can be stored in the form of triples.

The stored information is not limited to what we have categorized. There are more details involved in each one of these categories that we will discuss more in the implementation section. We also keep a complete log of conversations with the elderly which can be used later.

5.3.2 Context Knowledge Base

Pronoun resolution can help a lot with understanding conversations and textual contents. In NLP tools designed for information extraction from text, pronoun resolution is usually done by looking for named entities or nouns in the previous sentences; however, this process is not the same in the conversations. Systems dealing with conversations usually get a sentence at a time which may not include the noun related to the pronoun. In these cases, the system needs to look for the nouns and entities in the previous sentences to

know to whom or to what the pronoun refers. Doing this kind of search when the system encounters a pronoun can be difficult and inefficient.

Instead of looking for nouns or named entities in previous sentences, we store nouns and named entities that the system is detecting in each sentence which can be referred to in the coming sentences. We keep last known masculine entity, last known feminine entity, last known object and last known location. When there is a pronoun in a sentence without mentioning the reference, it will be easier to look up what the system has caught for that particular kind of entity instead of checking previous sentences.

5.3.3 Inference

Since our knowledge does not contain the information that needs complex inferences, we do not have a complex inference engine either. The inferences that may be necessary in the context of the conversation, considering the knowledge that we are storing, are not that many. The one that we have added to our current implementation of our knowledge representation is relation inferences. Relation inferences may be necessary in the case of referring to a person with relation to the speaker or someone known by the system. An example of such inference is shown in Figure 5.1 . There can be other places in our knowledge that might need inference, but we do not include them in the current model.

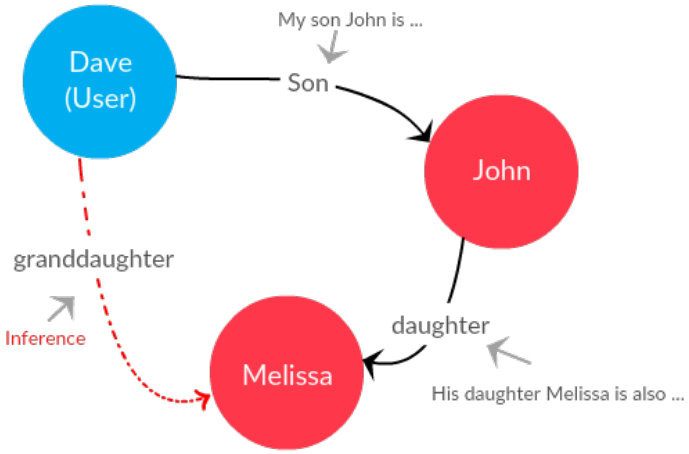


Figure 5.1: An example of relation inference

5.3.4 Implementation

The current implementation for our specific knowledge representation is done on a traditional relational database. Since we have a predefined set of attributes related to each entity in our knowledge base, using a relational database will not have any downsides. We have eight tables in each database representing required knowledge bases for each user. These tables are:

- **People:** It includes information about the people that a user has talked about during the conversations. The table for people includes the following columns: Name, Age, Gender, Likes, Dislikes, Profession, Possessions, Relation, Relation-to, Birthday, Location and Miscellaneous.

- **Events:** For the events involving the user or the others, we have this table to keep information. The columns in this table are location, start date and time, end date and time and attending people.

- **Medical:** In this table, we keep on-going medical conditions of the user that we have extracted from the conversation. The medical information can be about how the person is or the symptoms of an illness. We have a simple separation of feelings and symptoms according to some keywords and key-phrases. This table has the following columns: Type of record (Condition/Symptom), trigger word for extraction of the record, the sentence that the record is extracted from, and Timestamp.

- **Drug intake:** The information in this table is the records of the drugs that the person takes during the day. We keep the following information about each record: Drug name, Response keyword, Interpretation of the system, Response sentence and Timestamp. Usually, the system gets this information in response to a question generated by the system. Response keyword is for keeping which words or phrases have been used by the system to decide what the user's response was and the interpretation of the system keeps the record of what system has decided based on the response keyword.

- **Drugs and Conditions:** This table is used for keeping what drugs or conditions the system should monitor. The information for conditions can be in the form of keywords while the drugs should have name and the intake frequency.

- **CKB:** We use this table for keeping information of the context knowledge base. We keep the last two known entities from each category with their timestamp.

- **Triples:** We keep the triples extracted by Open Information Extraction system in this table. In addition to triples, we keep the sentence and timestamp of the record as well.

- **Sentences:** We keep a record of all the sentences that have been said by the user or generated by the system. This table includes the sentences as well as the timestamps of their utterance.

The inference engine implementation is done based on pre-defined rules. The inference will be done only on the relations that can be inferred with high certainty, eliminating many inferences that may not be correct. An example of such an inference is shown in Figure 5.2.

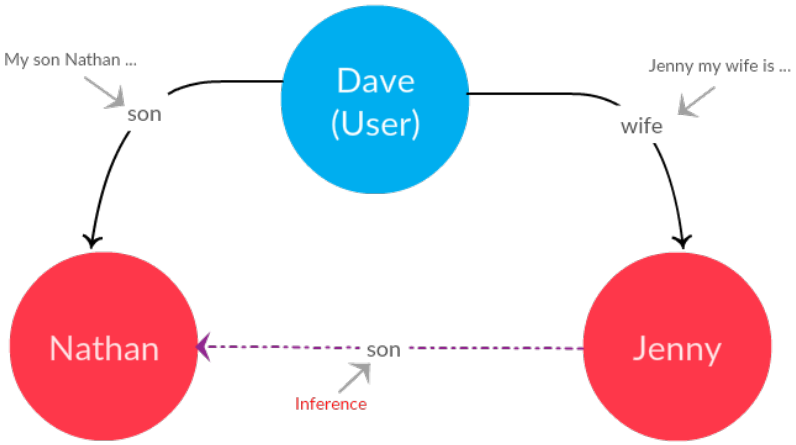


Figure 5.2: A relation inference which is not necessarily sound

5.4 Conclusion

The current state of knowledge representation allows us to develop a smart agent for elderly care. However, we can have better knowledge representation if we switch to semantical models instead of using relational databases. Switching to that kind of knowledge base can be considered as a future improvement to what we have.

Chapter 6

Response Generation

Response generation in our system is vital in the interactions between the user and the system. The generated responses should be in the context of conversation while maintaining human-like nature of natural language. Natural Language Generation (NLG) is the research field that concerns with the similar problems. NLG is content and application dependent, which means the process and the method that is used for an application may completely differ from one system to the other.

In Section 6.1, we will explain the classical definition and the processes of NLG. In Section 6.2, we will review common approaches to natural language generation in the context of the conversation. In Section 6.3, we will examine the research related to our intended implementation of NLG and in Section 6.4, we will discuss our current method with its components. We will conclude this chapter in Section 6.5.

6.1 Natural Language Generation

Natural Language Generation is the field of research that deals with the creation of text or responses in the natural language form. In contrast with Natural Language Understanding, which concerns mostly about ambiguity, NLG is about decision making. In creating a natural language response, there are many decisions involved. The decision-making process for NLG is categorized by Reiter and Dale as six dependent tasks [65]. These tasks are:

- **Content Determination:** In this task, the content that should be

included in the message is selected from the input or the data sources of the system. The selected information, regardless of its origin, should be in a structured way which can differentiate entities, relations etc. Usually, in this step, a list of viable contents is selected.

- **Document Structuring (Discourse Planning):** After selecting the content, it is important to present them in a sensible order. Document Structuring imposes this order to the text. In the case of creating a paragraph, it is important that which contents should be mentioned before the others.

- **Sentence Aggregation:** In this task, after planning the structure, we group the ordered messages, to create sentences or paragraphs. It is not always necessary to group these messages since each one of them can be expressed separately, but aggregation makes the final output more natural and fluent.

- **Lexicalization:** Based on the selected content, in this step, we choose the most appropriate words and phrases that can explain the chosen messages in the intended domain. Although these words can be hard coded as well for each type of entity and relation, which can make this task trivial.

- **Referring Expression Generation:** In this task, the words or phrases are selected in order to distinguish the different entities of the intended domain. This task is very similar to lexicalization, but it is formalized as a discrimination task which makes sure that enough information about the domain is conveyed in the message to avoid confusion or ambiguity.

- **Linguistic Realization:** In this task, the rules of grammar are applied to the processed content to create an output which is syntactically, morphologically and orthographically correct. In this process, other words can be added to the message.

For generating a document in the natural language form, each one of these

steps should be planned and implemented carefully. For the responses generated for a conversation, since we will not have long textual contents in each response, some of these tasks can be skipped or combined with the others. Also, the complexity of the implemented components in the context of a conversation will be lower. In the next section, we will review the utilized approaches for bringing NLG to conversational agents.

6.2 Different Approaches to NLG in the Context of Conversation

The conversational agents or chatbots use different methods for creating appropriate responses to each message that they are getting from the users. However, all of these methods can be categorized into two general approaches of retrieval-based and generative. We will examine each one of these approaches in the next two subsections and their advantages and disadvantages in the last subsection.

6.2.1 Retrieval-based Methods

Systems that are based on retrieval approach usually have a set of predefined responses. According to the input that they are getting from users, the system uses a selection method to match the input to one of the responses in the predefined set. This selection method can be a simple pattern matching, heuristic, or even classifiers based on machine learning.

There are many chatbots that utilize pattern matching as their main method of response generation. The very early chatbots were using simple pattern matching while the recent chatbots are using more advanced markup languages that are developed specifically for pattern matching like AIML and ChatScript.

Most of the chatbots that we use today are retrieval-based. The systems that are designed with this approach, when they are limited to a specific domain, perform very well. However, for general domain conversational agents, it is a very hard task to create a set of responses or even templates that

can cover the general knowledge. To overcome these issues, the generative approach is used. We will have a quick overview of AIML and ChatScript before discussing the generative approach.

6.2.1.1 AIML

AIML is a markup language based on XML. It was designed for the response generation in ALICE chatbot [86]. Each one of the rules is written inside the category tag (<category>). Inside the category tag, we have pattern (<pattern>) and template (<template>) tags. Listing 6.1 shows the general structure of a rule in AIML language.

Listing 6.1: General structure of a rule in AIML

```
<category>
<pattern>Input</pattern>
<template>Response</template>
</category>
```

Pattern tag is used by the system to check the input with the rule. The pattern should be covered entirely by the input, and the content of pattern tag is case sensitive. It is possible to use wildcards (*) in the pattern in combination with one or more words. Listing 6.2 is an example of using wildcards within the pattern [14].

Listing 6.2: Wildcard usage in the pattern tag

```
<category>
<pattern> I NEED HELP * </pattern>
<template>
    Can you ask for help in the form of a question?
</template>
</category>
```

One of the useful features of the AIML language is the ability to call itself recursively. The recursion is done by including srai (<srai>) tag inside the pattern tag as the output. AIML also has the ability to pass the rest of input, in the case of using wild cards. The rest of an input can be passed to the language with star (<star/>) tag. An example of recursion is shown in Listing 6.3 [14].

Listing 6.3: Recursive calls in AIML

```
IN => Can you please tell me what LINUX is right now?
<category>
<pattern> * RIGHT NOW </pattern>
<template><srai><star/></srai></template>
</category>

IN => CAN YOU PLEASE TELL ME WHAT LINUX IS
<category>
<pattern> CAN YOU PLEASE * </pattern>
<template><srai> Please <star/></srai></template>
</category>

IN => Please TELL ME WHAT LINUX IS
```

AIML has two optional tags that can help with more complicated conversations. The first optional tag is that (<that>); if it is included in the rule (category tag), the previous input should match the contents of that tag. Having that tag inside a rule can help in creating rules that are dependent to previous utterances. The other optional tag is topic (<topic>) which is used outside of the category tags to group a set of category tags. Topic tag allows having the same pattern multiple times as the included rules for the chatbot.

6.2.1.2 ChatScript

ChatScript is the successor of AIML which fixes the issue of not finding a pattern for an input. It has a better syntax and included additional functionalities such as concepts, continuations and functions. A concept can be defined as Listing 6.4 in ChatScript [14].

Listing 6.4: Definition of a concept in ChatScript

```
concept: ~meat ( bacon ham beef meat flesh veal lamb
    chicken pork steak cow pig )
```

The new functionalities in ChatScript are trying to address the need for ontologies in rule-based chatbots. With the new syntax, the knowledge can be included within the scripts. Similar to AIML, ChatScript uses topics to organize the rules. Each topic is saved as a separate file with top file extension. Listing 6.5 depicts an example of a topic written with ChatScript [67].

Listing 6.5: A ChatScript Topic File

```
topic: ~pets (dog cat pet animal bird fish snake)

?: ( << you like snake >>)
 I love pythons except Python (the programming language).

?: ( << you ~like ~animals >>)
 I love all animals.

t: Do you have any pets?
#! yes
 a: ( yes) Great. You like animals.

#! no
 a: (no) You don t like animals?

#! I have two parrots
 a: (parrots) Birds are nice.

#! I have a cat
 a: (cat) I prefer dogs

#! I have a canary
 a: ([parrot bird canary finch swallow]) Birds are nice.

t: I have a dog.
```

6.2.2 Generative Methods

Using generative methods for the conversation in the general domain seems a better approach toward response generation. These methods do not rely on a default set of responses and they create a response for each statement from scratch. Most of the generative methods are based on machine translation but in these systems, instead of translating the input from a language to another, it tries to translate the input to the output.

The research for NLG with generative methods is in its early stages. The current state of the implementations is prone to grammatical mistakes and in some cases, the responses are too general to convey any particular message.

6.2.3 Comparison of Retrieval and Generative Approaches

Using methods based on each one of these approaches has its own advantages and disadvantages. To have a better organization of them we have created a list of them:

- **Grammatical Correctness:** In retrieval based methods, since most of the sentences are predefined, as long as the sentences that are included in the rules or dataset of the system are correct, the response will also have correct grammatical structure. On the other hand, in generative methods, there are usually grammatical mistakes in the structure of responses which is also seen in the machine translation of documents.

- **Remembering Previous Context:** In retrieval based methods, since the conversation is not making any changes to the rules or the classifier, the system can not remember details of a previous conversation. In generative methods, since there are no canned responses, the system usually uses all the new information that is added during a conversation.

- **Ease of Implementation:** Retrieval-based models are easier to implement in comparison to the generative models.

- **Amount of Necessary Data:** The amount of data that is necessary for creating a generative response system is more than the amount of data that we can use for training a classifier that works with a retrieval-based method.

- **Closed Domain versus Open Domain:** Retrieval-based methods perform very well in conversations in closed domains, however, having a conversational agent performing in general domain of knowledge is impossible since everything should be included in the predefined rules or dataset. On the other hand, generative methods, theoretically, should be able to work in open domain conversations. Creating such a system

is equivalent to creating an Artificial General Intelligence, which, at the present, is impossible.

Regardless of what method of response generation is used in a system, the answers that it gives should be consistent. This problem becomes more apparent when the questions are about the system itself which is assumed to be an entity. Listing 6.6 shows an example of such question and answer pairs that are inconsistent [15]. In order to overcome this problem, an identity can be designed for the system which can be hard coded and unchangeable, which can make the answers consistent. Adding persona to chatbots is a field of research in NLG.

Listing 6.6: Inconsistent responses by a chatbot

```
message   Where do you live now?
response  I live in Los Angeles.
message   In which city do you live now?
response  I live in Madrid.
message   In which country do you live now?
response  England, you?
```

6.3 Related Work

The retrieval based model for creating conversational agents is an old approach. There are many chatbots that are using rule-based methods for generating responses and this approach is used within recent Loebner winning chatbots. We have listed some of them and their method of implementation in Chapter 2.

The recent research in the area of NLG is more focused on the generative models. There are a number of works that are utilizing different types of neural nets and other machine learning techniques for generating word by word responses.

The first work that tried to use machine translation as a generative method for NLG goes back to 2011. Ritter et al. [66] devised a method to generate responses for micro-blogging posts with machine translation. They found that this task is harder than the actual translation between two languages, because

of the many choices that can be relevant in the case of response generation and also, the words in the output might not be aligned with the input text.

In 2015, Shang et al. used recurrent neural networks for response generation in micro-blogging posts [73]. Sordoni et al. continued their work and extended it from pairs of status-reply to triples of consecutive utterances [76]. Serban et al. extended both of these works with using Hierarchical Recurrent Encoder-Decoder with the recurrent neural networks which shows performance improvement over the previous works, especially in cases of having larger datasets [71]. Sharma et al. improved their work by adding lexicalized data to the training process which makes the sentences sound more natural with better grammar [74]. Li et al. improved previous works by Adversarial Learning which is inspired by Turing Test and uses reinforcement learning [49].

In extending the generative models, Yao and Zweig proposed a model based on recurrent neural networks that models attention and intention within conversations as well [92]. As another improvement, Li et al. tried to address the issue of consistency in the conversational agents by modelling a persona within the neural models [47].

In more recent works, the idea of using a fixed dataset for dialogue agents is challenged [90]. Since this is not the way that humans learn the language, it is believed that a model that considers on-going conversations as part of its training model would achieve better results and will self-improve [90]. Su et al. are following the same goal by combining the supervised method with reinforcement learning [80].

Another issue that is addressed by the work of Li et al. is considering the previous interactions and looking into the future directions of the conversation for creating an appropriate response [48]. They are using deep reinforcement learning to this purpose.

6.4 Our Method

Based on the design criteria, the conversations that we are expecting in our system will be in the general domain, which makes the generative methods

more suitable. However, since we did not have enough time to build an operational generative model, we are using an intuitive combination of a retrieval model, an answering system for questions in general knowledge domain, and the current state of our personalized knowledge base as the method of response generation. In the next subsections, we will explain each one of these components and their operation procedures and then, we will discuss how they work together to create responses.

6.4.1 Rule-Based Response Generation

We have devised a set of rules based on regular expressions which have a very similar functionality to AIML and ChatScript. With these hand-crafted rules, we are covering the answers to the questions that should be generated from the collected information in the PKB, the responses to some of the declarative and imperative statements, and conversation openers.

Based on each class of entity and their attributes, we have a set of rules that can help in identifying answers to the questions that the user might ask. After matching a pattern, a predefined template will be filled with the complementary information from PKB to generate an appropriate response. However, if the question is in the domain of PKB, but the question is missing some information that is necessary to distinguish between the stored entities, the system has rules for generating follow-up questions. The system might need to ask several questions to get the necessary information for answering a specific question. We keep track of these questions and responses in the Question Stack which will help the system, after getting all the necessary information, answer the initial question.

For generating responses to some of the declarative and imperative statements, we also use pattern matching to find a suitable response. The conversation openers are a list of predefined templates that can be chosen and filled according to the current condition and previous conversations. This selection process for conversation openers is also rule-based.

6.4.2 Question Answering

The questions by the users may fall out of the scope of our knowledge bases. These questions are mostly about general information which are known generally by humans. For the questions that are in the general knowledge domain, we are using Wolfram Alpha to find an appropriate answer. Wolfram Alpha has its own natural language understanding system which makes the process of querying straightforward.

In order to find an answer, the question in natural language format is sent to Wolfram Alpha by its API. The response from Wolfram Alpha is in XML format and includes the answer in Natural Language form, in addition to links to images, maps, videos and other contents that may be relevant to the question.

In the current implementation of the system, we are only using the textual answer of the query, omitting all the additional information that is sent by the server. However, the visual contents can be included later to the interface of the client software.

6.4.3 Question Queue

In the process of information extraction, there are attributes for some of the entities that are left blank. We have a set of these missing attributes. Some of these attributes are more important in the context of the conversation. For example, in talking about a family member, it is very important that the system knows the person's name and relation. This information becomes more useful in future when the user mentions that person by name or relation. Based on this observation, the attributes in this set have predefined priorities for selection. Missing attributes with high priority are used for question generation in our RG system.

6.4.4 Combined Response Generation Model

For combining these three methods, the result of Speech Act Recognition is very helpful. The general question answering mechanism is only used when

the input is an interrogative statement and the answer can not be resolved based on the defined rules. Also, in the case of using pronouns for referring to previous entities within the question, since we are using CKB to store the last known entities and their origin (PKB or general domain), it is easier to choose one of the methods of question answering.

Question queues are usually used when some of the vital information is missing in declarative statements. For creating responses from the set of questions, we are using rules as well.

6.5 Conclusion and Future Work

The current response generation model is far from perfect. As it was mentioned in the beginning of Section 6.4, our goal is conversations in general domain with incorporating personal information that is stored in PKB. For achieving our goal, a generative response generation method is necessary and should be considered as a future direction for this part of our project.

Chapter 7

System Design

In this chapter, we will look into the implementation of the conversational agent that we have built with the intention of helping the elderly. It is very important to consider that the current implementation is the proof of concept and there are more steps necessary in the implementation to have the system ready for a public release.

This chapter is organized in five sections. In Section 7.1, we will look into the platforms that we have chosen for the implementation of the agent. In Section 7.2 and Section 7.3, we will discuss the components that are included in the client-side and server-side platforms of the agent respectively. In Section 7.4, we will follow the procedure of how a message is processed in the system and in Section 7.5, we will conclude this chapter with some future directions for better implementation of the system.

7.1 System Platform

We need to implement our conversational agent on a platform that gives easy access to the system while keeping complicated procedures to a minimum. The most common devices meeting these criteria are smart devices, which compared to the traditional desktop computers, are easier to use. Because of their limited computational power, we need a server to compliment the app on the devices. In the following sub-sections, we are going to look into our client and server platforms, the user identification method and the way that client and server communicate with each other.

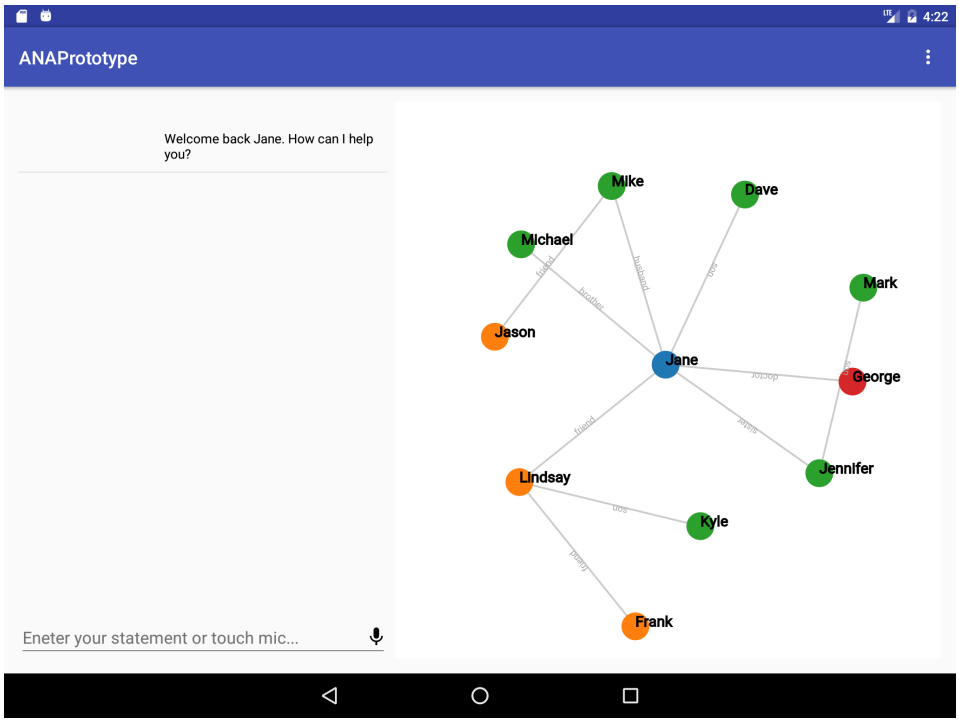


Figure 7.1: Current user interface of the prototype

7.1.1 Client

There is a wide range of mobile smart devices available on the market. We need to consider the market share of each platform and the average price of devices based on it to make an informed decision about our target platform. By considering these factors, we have chosen Android devices as our target in the implementation of the client software for end users.

We are targeting Android tablet devices in particular for our interface because of their larger screens which make them preferable for the elderly who may have vision impairment. Figure 7.1 depicts the current interface of our system.

7.1.2 Server

Implementing all the necessary functionalities for the system operation on the end devices is very desirable since there would not be a need for a network connection for communicating with the agent; however, the necessary computational power for some of the components makes it impossible to implement

them on mobile devices. To move the heavy computational operations from end users' devices, we are using a server to complement the app.

Moving computationally heavy components from end devices to a server has many advantages. This move makes our application adaptable by a larger range of Android devices and reduces the amount of necessary storage on the devices. Also, by having some of the information stored on the server, migrating from an end device to another can be easier.

The server and libraries that we are using are implemented in Java with no dependencies to the operating system, which makes it platform independent. For now, we are using Debian as our server platform.

7.1.3 User Identification

Since the system is designed to deal with multiple users, we need to identify unique users. We are using Google Sign-In as our method of identification and authentication. In Android devices, to access all the capabilities of the device, it is necessary to have a Google account associated with the device. Google Sign-In gives developers access to the data from Google account which can be used within the apps. We are using only basic information from Google Account which includes user's name and Google ID.

7.1.4 Client-Server Communication

The current design of the system needs a network connection to reach our server for the operation. Our communication has a RESTful like behaviour and the messages moving between client and server are in the form of Java Script Object Notation (JSON). We are using sockets over TCP for the message passing between client and server. Figure 7.2 shows the components of client and server and how these components communicate with each other.

7.2 Client-Side Components

We have tried to move most of the components to the server side to make the application as light as possible. Having a lighter application makes the

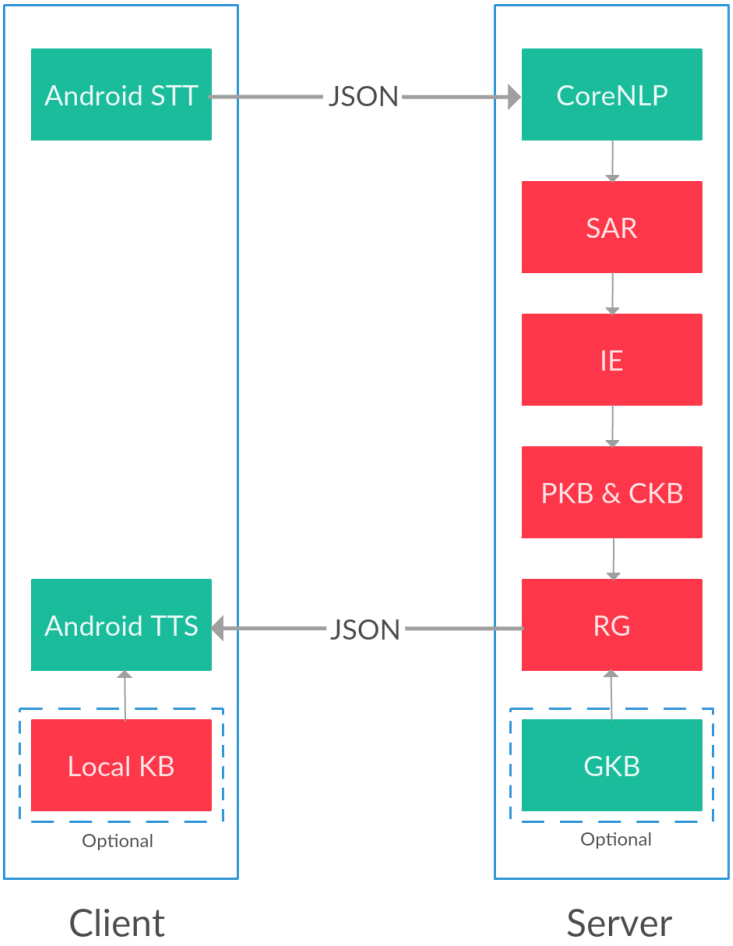


Figure 7.2: Client and server components

system compatible with cheaper Android devices, which in return, can make the adaptation of our system in the future easier. The following components are part of our client application:

- **Speech to Text Engine:** In the Android operating system, we have SpeechRecognizer in android.speech package which can do the conversion of speech to text. This library has a high accuracy for the conversion and an extensive number of supported languages.

- **Text to Speech Engine:** We are using android.speech.tts package for reading responses generated by the system to the user. There are other engines with more natural voices but the inclusion of this package within the operating system makes it a better option for our system. However,

if the default text to speech engine of the operating system is changed by the user, the system will also use the option that the user has set for the operating system.

- **Local Knowledge Base:** We store some of the information locally in the device. This information is related to the tasks that are supposed to be done on the device with the data that is already in it. For example, for calling to someone known by the system, we are keeping the matching contact details from the system and contact data on the device in the local knowledge base.

7.3 Server-Side Components

The components that we have discussed in Chapters 3 through 6 are usually dealing with intensive computations which makes them unsuitable to be implemented on handheld devices with low computational power and memory. The implementation of these systems on the server makes the app more responsive. Most of these components rely on the result of NLP analysis of input. In our case, most of the analysis are done by Stanford CoreNLP. CoreNLP requires more than 2 GB of RAM for running our required analysis on the input. We are using the following parts of CoreNLP in our system: Tokenizer, Lemmatization, Part of Speech Tagger, Dependency Parser, Named Entity Recognizer, and Open Information Extraction.

The result of CoreNLP analysis feeds our different components. Our main components on the server are:

- **Speech Act Recognizer:** It uses the result of CoreNLP analysis and SVMLight library to classify the sentences into three categories of declarative, interrogative and imperative.

- **Information Extractor:** Based on the result of CoreNLP analysis, it extracts relevant information to populate our knowledge bases.

- **Response Generator:** Based on the speech act of the sentence, extracted information and available information in the knowledge bases,

it generates the most appropriate response to each sentence. It also can also generate a sentence to start a conversation without the user's initiation.

- **Knowledge Bases:** We have PKB and CKB which we store on the server locally. We are using MySQL server for implementation of local knowledge bases. We also use general knowledge bases like Wolfram Alpha for answering questions in the general domain.

7.4 Working Procedure of the System

In this section, we will look into how the agent processes received messages from users. Generating a conversation opener by the server is done by the response generation mechanism, which is going to use question queues and conversation history to find an appropriate initiation.

When a message is received by the server, it contains the textual content of user's utterance with the unique identifier. The textual content of the message is analyzed by CoreNLP tools before feeding the message to our components.

First, the message and the result of the analysis is passed to the speech act recognizer. It classifies the statement into three categories of declarative, interrogative and imperative. Then, regardless of the class of the message, we pass the message and analysis result to the information extraction component. If the sentence is declarative, the extracted information will be added to the personal knowledge base. Also, based on the extracted information, the context knowledge base is updated as well.

After having the message analyzed by SAR and IE, we need to create an appropriate response to the message. According to the class of message, response generation will try to answer the question, if the class of the utterance is interrogative, create an acknowledgement for a comment, if the message is imperative, or create a declarative or interrogative statement as a response to the declarative statement by the user. Figure 7.3 shows a flow chart that captures a simplified procedure of message processing.

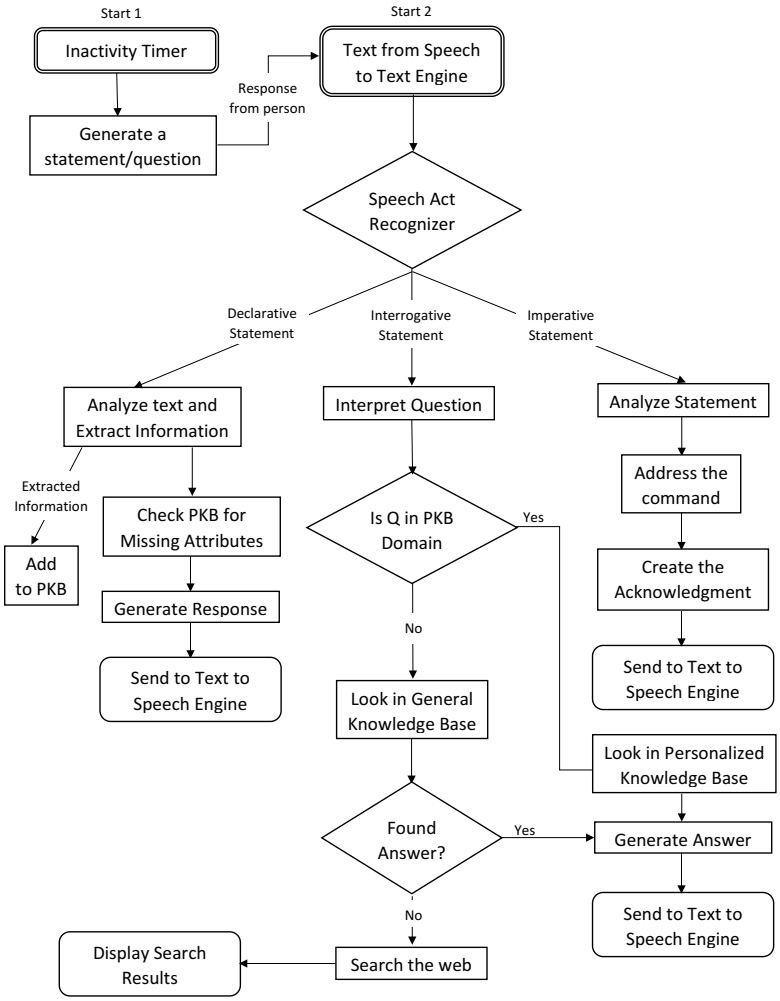


Figure 7.3: Simplified flowchart of system

7.5 Conclusion

The current implementation of the agent is a proof-of-concept implementation to show that the current architecture of the system is able to address some of the stated goals of the agent. However, there is more work necessary for making the agent more suitable for users.

On the client side, we need a better interface design which can help users' interactions become more personalized. For example, adding an avatar for the agent can make the app more relatable. Also, using a better speech engine for responses can improve the overall user experience on the system.

On the server side, each component can be improved to make better responses to users' statements. We have discussed these changes in each chapter related to each component separately. However, the most important component that can make a significant improvement in the conversations is the response generator since it is the way of communicating different messages to the user.

Chapter 8

Conclusion

Our work is motivated by many issues that are dictated by lifestyle changes in the last century. We have tried to devise an assistive smart agent that would help the elderly with such issues like companionship and nursing. The communication medium of our choice for this agent is speech. We have identified the necessary building blocks of the agent and tried to find or devise a method for the handling of the operations inside of each block. We also created a database of extracted information from conversations with users which is employed for question answering and response generation.

We had an internal evaluation of the system with the help of our colleagues. Since we do not have the ethics approval for this evaluation, we cannot include the results in the manuscript. In this evaluation, we asked the participants to rank three conversations by different anonymous chatbots including ANA in two different scenarios and two conversations with and without a personalized knowledge base. The results showed that almost all of the participant preferred ANA to the other chatbots and all the participants preferred the output with PKB to the one without it. For conducting a less biased evaluation, instead of asking all the participant to rank the same conversation, it is better to have a pool of conversations with different scenarios for choosing two or more scenarios for the ranking process by each participant.

The current implementation of building blocks and system is proof of concept and the current functional state of the system may not meet all the requirements for proper user experience. There is more work to be done for

each identified block and the system itself to achieve our goal. However, with all the limitations, the current functionality of the system is promising and it shows that having a knowledge base with personal information of the users can improve the fluency of conversations.t

In addition to the current blocks and operational units, we can add emotion analysis to the system as well. The generated responses can be based on the emotional state of the user in order to mimic sympathy and care. Also, with learning-based methods, we can have a system that adapts with the preference of the user in the method of response generation. This can include characteristics like being formal or casual in conversation, the mood of the agent, as well as adapting the emotional state of the responses to the learned patterns of the user's behaviour. These new additions can be done in a separate module which depends heavily on RG or can be part of the RG component. These improvements are considered as future expansions to the current system.

Bibliography

[1] Hadeel Al-Zubaide and Ayman A Issa. Ontbot: Ontology based chatbot. In *Innovation in Information & Communication Technology (ISIICT), 2011 Fourth International Symposium on*, pages 7–12. IEEE, 2011.

[2] Peggy M Andersen, Philip J Hayes, Alison K Huettner, Linda M Schmandt, Irene B Nirenburg, and Steven P Weinstein. Automatic extraction of facts from press releases to generate news stories. In *Proceedings of the third conference on Applied natural language processing*, pages 170–177. Association for Computational Linguistics, 1992.

[3] Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. *arXiv preprint arXiv:1603.06042*, 2016.

[4] Gabor Angeli, Melvin Johnson Premkumar, and Christopher D Manning. Leveraging linguistic structure for open domain information extraction. *Linguistics*, 1(24), 2015.

[5] Chinatsu Aone, Lauren Halverson, Tom Hampton, and Mila Ramos-Santacruz. Sra: Description of the ie2 system used for muc-7. In *Proceedings of the seventh message understanding conference (MUC-7)*, 1998.

[6] Chinatsu Aone and Mila Ramos-Santacruz. Rees: a large-scale relation and event extraction system. In *Proceedings of the sixth conference on Applied natural language processing*, pages 76–83. Association for Computational Linguistics, 2000.

[7] Amaya Arcelus, Megan Howell Jones, Rafik Goubran, and Frank Knoefel. Integration of smart home technologies in a health monitoring system for the elderly. In *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*, volume 2, pages 820–825. IEEE, 2007.

[8] Agnese Augello, Giovanni Pilato, Giorgio Vassallo, and Salvatore Gaglio. A semantic layer on semi-structured data sources for intuitive chatbots. In *Complex, Intelligent and Software Intensive Systems, 2009. CISIS'09. International Conference on*, pages 760–765. IEEE, 2009.

[9] Agnese Augello, Giovanni Pilato, Giorgio Vassallo, and Salvatore Gaglio. Chatbots as interface to ontologies. In *Advances onto the Internet of Things*, pages 285–299. Springer, 2014.

[10] John Langshaw Austin. *How to do things with words*. Oxford university press, 1975.

[11] Roy F Baumeister and Mark R Leary. The need to belong: desire for interpersonal attachments as a fundamental human motivation. *Psychological bulletin*, 117(3):497, 1995.

[12] Roberto Bernabei, Francesco Landi, Giovanni Gambassi, Antonio Sgadari, Giuseppe Zuccala, Vincent Mor, Laurence Z Rubenstein, and PierUgo Carbonin. Randomised trial of impact of model of integrated care and case management for older people living in the community. *Bmj*, 316(7141):1348, 1998.

[13] Daniel M Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. Nymble: a high-performance learning name-finder. In *Proceedings of the fifth conference on Applied natural language processing*, pages 194–201. Association for Computational Linguistics, 1997.

[14] Luka Bradeško and Dunja Mladenić. A survey of chatbot systems through a loebner prize competition. In *Proceedings of Slovenian Language Technologies Society Eighth Conference of Language Technologies*, pages 34–37, 2012.

[15] Denny Britz. Deep learning for chatbots, part 1 introduction. http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction/, 2016. Accessed: 2017-02-23.

[16] P. D. Census Bureau. Projections of the population by selected age groups and sex for the united states. http://www.census.gov/population/projections/data/national/2012/summarytables.html, 2012. Accessed: 2016-07-24.

[17] June A Carletti. Volunteers provide companionship therapy under social service supervision. *Psychiatric Services*, 15(12):691–693, 1964.

[18] Asli Celikyilmaz, Dilek Hakkani-Tur, Hua He, Greg Kondrak, and Denilson Barbosa. The actortopic model for extracting social networks in literary narrative. In *NIPS Workshop: Machine Learning for Social Computing*, 2010.

[19] William W Cohen, Vitor R Carvalho, and Tom M Mitchell. Learning to classify email into"speech acts". In *EMNLP*, pages 309–316, 2004.

[20] Kenneth Mark Colby. Modeling a paranoid mind. *Behavioral and Brain Sciences*, 4(04):515–534, 1981.

[21] Gao Cong, Long Wang, Chin-Yew Lin, Young-In Song, and Yueheng Sun. Finding question-answer pairs from online forums. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 467–474. ACM, 2008.

[22] Alex Cranz. No Really, Google Assistant Is Dumb. https://gizmodo.com/no-really-google-assistant-is-dumb-1787951274, 2016. Accessed: 2016-11-24.

[23] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. A framework and graphical development environment for robust nlp tools and applications. In *ACL*, pages 168–175, 2002.

[24] Christopher Davis. Uf "smart home" demonstrates concept of automated elderly help and care. http://publish-www-ufl.wcm.osg.ufl.edu/news—archive/archive/2003/11/uf-smart-home-demonstrates-concept-of-automated-elderly-help-and-care.html, 2003. Accessed: 2016-11-24.

[25] Randall Davis, Howard Shrobe, and Peter Szolovits. What is a knowledge representation? *AI magazine*, 14(1):17, 1993.

[26] George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. The automatic content extraction (ace) program-tasks, data, and evaluation. In *LREC*, volume 2, page 1, 2004.

[27] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM, 2014.

[28] Witold Drozdzynski, Hans-Ulrich Krieger, Jakub Piskorski, Ulrich Schäfer, and Feiyu Xu. Shallow processing with unification and typed feature structures - foundations and applications. *KI*, 18(1):17, 2004.

[29] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

[30] Thomas F Garrity, Lorann F Stallones, Martin B Marx, and Timothy P Johnson. Pet ownership and attachment as supportive factors in the health of the elderly. *Anthrozoös*, 3(1):35–44, 1989.

[31] Anne Hélène Gauthier et al. The state and the family: A comparative analysis of family policies in industrialized countries. *OUP Catalogue*, 1998.

[32] Google. SyntaxNet: Neural Models of Syntax. https://github.com/tensorflow/models/tree/master/syntaxnet, 2016. Accessed: 2016-11-24.

[33] Ralph Grishman and Beth Sundheim. Message understanding conference-6: A brief history. In *COLING*, volume 96, pages 466–471, 1996.

[34] Jerry R Hobbs, Douglas Appelt, John Bear, David Israel, Megumi Kameyama, Mark Stickel, and Mabry Tyson. Fastus: A cascaded finite-state transducer for extracting information from natural-language text. *arXiv preprint cmp-lg/9705013*, 1997.

[35] Kevin Humphreys, Robert Gaizauskas, Saliha Azzam, Chris Huyck, Brian Mitchell, Hamish Cunningham, and Yorick Wilks. University of sheffield: Description of the lasie-ii system as used for muc-7. In *Proceedings of the Seventh Message Understanding Conferences (MUC-7)*. Citeseer, 1998.

[36] Immigration, Refugees and Citizenship Canada. Notice supplementary information 2016 immigration levels plan. http://www.cic.gc.ca/english/department/media/notices/2016-03-08.asp, 2016. Accessed: 2016-11-21.

[37] Paul S. Jacobs and Lisa F Rau. Scisor: Extracting information from on-line news. *Communications of the ACM*, 33(11):88–97, 1990.

[38] Jiyou Jia. Csiec: A computer assisted english learning chatbot based on textual knowledge and reasoning. *Knowledge-Based Systems*, 22(4):249–255, 2009.

[39] Thorsten Joachims. Making large scale svm learning practical. Technical report, Universität Dortmund, 1999.

[40] Tomoko Kamimura, Rina Ishiwata, and Takenobu Inoue. Medication reminder device for the elderly patients with mild cognitive impairment. *American journal of Alzheimer's disease and other dementias*, 27(4):238–242, 2012.

[41] Sidney Katz, Laurence G Branch, Michael H Branson, Joseph A Papsidero, John C Beck, and David S Greer. Active life expectancy. *New England journal of medicine*, 309(20):1218–1224, 1983.

[42] Peter Kemper. The use of formal and informal home care by the disabled elderly. *Health services research*, 27(4):421, 1992.

[43] Andrew Lampert, Robert Dale, and Cecile Paris. Detecting emails containing requests for action. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 984–992. Association for Computational Linguistics, 2010.

[44] Wendy Lehnert, Claire Cardie, David Fisher, John McCarthy, Ellen Riloff, and Stephen Soderland. University of massachusetts: Muc-4 test results and analysis. In *Proceedings of the 4th conference on Message understanding*, pages 151–158. Association for Computational Linguistics, 1992.

[45] Wendy Lehnert, Claire Cardie, David Fisher, Ellen Riloff, and Robert Williams. University of massachusetts: Description of the circus system as used for muc-3. In *Proceedings of the 3rd conference on Message understanding*, pages 223–233. Association for Computational Linguistics, 1991.

[46] Baichuan Li, Xiance Si, Michael R Lyu, Irwin King, and Edward Y Chang. Question identification on twitter. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 2477–2480. ACM, 2011.

[47] Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155*, 2016.

[48] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.

[49] Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*, 2017.

[50] Wolfgang Lutz, Warren Sanderson, and Sergei Scherbov. The coming acceleration of global population ageing. *Nature*, 451(7179):716–719, 2008.

[51] Steven L Lytinen and Anatole Gershman. Atrans automatic processing of money transfer messages. In *AAAI*, volume 86, pages 1089–1093, 1986.

[52] Karl F MacDorman, Sandosh K Vasudevan, and Chin-Chang Ho. Does japan really have robot mania? comparing attitudes by implicit and explicit measures. *AI & society*, 23(4):485–510, 2009.

[53] John Markoff and Paul Mozur. For Sympathetic Ear, More Chinese Turn to Smartphone Program. http://www.nytimes.com/2015/08/04/science/for-sympathetic-ear-more-chinese-turn-to-smartphone-program.html, 2015. Accessed: 2016-11-24.

[54] Colin D Mathers, Gretchen A Stevens, Ties Boerma, Richard A White, and Martin I Tobias. Causes of international increases in older age life expectancy. *The Lancet*, 385(9967):540–548, 2015.

[55] Manish Mehta and Andrea Corradini. Developing a conversational agent using ontologies. In *International Conference on Human-Computer Interaction*, pages 154–164. Springer, 2007.

[56] Renee P Meyer and Dean Schuyler. Old age and loneliness. *The primary care companion for CNS disorders*, 13(2):e1, 2011.

[57] Paul Mozur. Chatting With Xiaoice. http://www.nytimes.com/interactive/2015/07/27/science/chatting-with-xiaoice.html, 2015. Accessed: 2016-11-24.

[58] Günter Neumann and Jakub Piskorski. A shallow text processing core engine. *Computational Intelligence*, 18(3):451–476, 2002.

[59] Hyacinth S Nwana. Software agents: An overview. *The knowledge engineering review*, 11(03):205–244, 1996.

[60] Giovanni Pilato, Agnese Augello, and Salvatore Gaglio. A modular system oriented to the design of versatile knowledge bases for chatbots. *ISRN Artificial Intelligence*, 2012, 2012.

[61] Giovanni Pilato, Agnese Augello, Giorgio Vassallo, and Salvatore Gaglio. Sub-symbolic semantic layer in cyc for intuitive chat-bots. In *International Conference on Semantic Computing (ICSC 2007)*, pages 121–128. IEEE, 2007.

[62] Jakub Piskorski. Express–extraction pattern recognition engine and specification suite. In *Finite-state Methods and Natural Language Processing: 6th International Workshop, FSMNLP 2007. Revised Papers*, page 166. Universitätsverlag Potsdam, 2008.

[63] Ashequl Qadir and Ellen Riloff. Classifying sentences as speech acts in message board posts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 748–758. Association for Computational Linguistics, 2011.

[64] Kevin Quinn and Osmar Zaiane. Identifying questions & requests in conversation. In *Proceedings of the 2014 International C* Conference on Computer Science & Software Engineering*, page 10. ACM, 2014.

[65] Ehud Reiter and Robert Dale. Building applied natural language generation systems. *Natural Language Engineering*, 3(01):57–87, 1997.

[66] Alan Ritter, Colin Cherry, and William B Dolan. Data-driven response generation in social media. In *Proceedings of the conference on empirical methods in natural language processing*, pages 583–593. Association for Computational Linguistics, 2011.

[67] Giorgio Robino. How to build your first chatbot using chatscript. https://medium.freecodecamp.com/chatscript-for-beginners-chatbots-developers-c58bb591da8, 2016. Accessed: 2017-02-23.

[68] John R Searle. *Indirect speech acts*. na, 1975.

[69] John R Searle. A classification of illocutionary acts. *Language in society*, 5(01):1–23, 1976.

[70] John R Searle. Minds, brains, and programs. *Behavioral and brain sciences*, 3(03):417–424, 1980.

[71] Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. *arXiv preprint arXiv:1507.04808*, 2015.

[72] Burr Settles. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, pages 104–107. Association for Computational Linguistics, 2004.

[73] Lifeng Shang, Zhengdong Lu, and Hang Li. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*, 2015.

[74] Shikhar Sharma, Jing He, Kaheer Suleman, Hannes Schulz, and Philip Bachman. Natural language generation in dialogue using lexicalized and delexicalized data. *arXiv preprint arXiv:1606.03632*, 2016.

[75] Lokesh Shrestha and Kathleen McKeown. Detection of question-answer pairs in email conversations. In *Proceedings of the 20th international conference on Computational Linguistics*, page 889. Association for Computational Linguistics, 2004.

[76] Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714*, 2015.

[77] Nova Spivack. How Siri Works: Interview with Tom Gruber, CTO of SIRI. http://www.novaspivack.com/technology/how-hisiri-works-interview-with-tom-gruber-cto-of-siri, 2010. Accessed: 2016-11-24.

[78] Canada Statistics. Living arrangements of seniors. https://www12.statcan.gc.ca/census-recensement/2011/as-sa/98-312-x/98-312-x2011003_4-eng.cfm, 2011. Accessed: 2016-07-24.

[79] Andreas Stolcke, Noah Coccaro, Rebecca Bates, Paul Taylor, Carol Van Ess-Dykema, Klaus Ries, Elizabeth Shriberg, Daniel Jurafsky, Rachel Martin, and Marie Meteer. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373, 2000.

[80] Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. Continuously learning neural dialogue management. *arXiv preprint arXiv:1606.02689*, 2016.

[81] Martin Theobald. Java native interface (jni) for svmlight. http://adrem.ua.ac.be/ tmartin/, 1999. Accessed: 2016-11-21.

[82] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[83] Alan M Turing. Intelligent machinery, a heretical theory. *The Turing Test: Verbal Behavior as the Hallmark of Intelligence*, 105, 1948.

[84] Alan M Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[85] Jennifer B Unger, C Anderson Johnson, and Gary Marks. Functional decline in the elderly: evidence for direct and stress-buffering protective effects of social interactions and physical activity. *Annals of Behavioral Medicine*, 19(2):152–160, 1997.

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

[87] Richard S Wallace. The anatomy of alice. In *Parsing the Turing Test*, pages 181–210. Springer, 2009.

[88] Yongdong Wang. Your Next New Best Friend Might Be a Robot. http://nautil.us/issue/33/attraction/your-next-new-best-friend-might-be-a-robot, 2016. Accessed: 2017-02-27.

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

[90] Jason E Weston. Dialog-based language learning. In *Advances in Neural Information Processing Systems*, pages 829–837, 2016.

[91] Michael Wooldridge and Nicholas R Jennings. Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(02):115–152, 1995.

[92] Kaisheng Yao, Geoffrey Zweig, and Baolin Peng. Attention with intention for a neural network conversation model. *arXiv preprint arXiv:1510.08565*, 2015.