



CREATIVE TECHNOLOGY BACHELOR THESIS

# MODELING ASSOCIATIVE MEMORY IN ROBOTS FOR PROMOTING SOCIAL BEHAVIOR

Kuiken, J.A. (Jaro)

Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS)

Supervisor: Kamilaris, Andreas, dr.  
Critical observer: Epa Ranasinghe, C.M.

2020 – 07 – 13

UNIVERSITY OF TWENTE.

## Abstract

The goal of the GP is to create an associative memory model to help social robots behave more empathetically. To achieve this goal, the way associative memory works in humans is explored. Existing associative memory models are explained and discussed. Software frameworks for the few models that have them are experimented with, to see if they could be used to reach the goal of the thesis. Eventually, all existing associative memory models are declared unsuitable for this thesis for differing reason. A decision is made to create a new conceptual associative memory model from scratch, using various design processes. This model is held to certain requirements acquired through different ideation techniques. After the creation of the model and, a conceptual pseudocode implementation is created. Both the model and its implementation are evaluated through different means and declared suitable for reaching the goal of this thesis. The long-term goal that this thesis begins to aim at is to eventually create a social robot that will be put into situations such as supermarkets, airports, museums, public parks, city centers and more. In these places the social robot can then help or socially interact with people in that place in whichever way is appropriate for the location.

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>State of the art</b>	<b>6</b>
2.1	Differences between single- and dual-process models . . . . .	6
2.2	Single-process models . . . . .	6
2.2.1	LIDA's Perceptual Associative Memory (PAM) module . . . . .	7
2.2.2	Search of Associative Memory . . . . .	11
2.2.3	Mnemograms . . . . .	13
2.3	Dual-process models . . . . .	13
2.3.1	Java-based Associative Memory . . . . .	14
2.3.2	Dual-Process Signal Detection (DPSD) . . . . .	15
2.4	Evidence for dual-process models . . . . .	16
<b>3</b>	<b>Method</b>	<b>18</b>
3.1	Orientation . . . . .	18
3.2	Associative memory research . . . . .	18
3.3	Experimenting . . . . .	19
3.4	Learning bottom-up through scenarios . . . . .	20
<b>4</b>	<b>Ideation</b>	<b>23</b>
4.1	Role of the memory model . . . . .	23
4.2	Origins of existing models . . . . .	23
4.3	Single- or dual-process model . . . . .	24
4.4	Software frameworks . . . . .	25
4.5	Scenarios and model requirements . . . . .	35
4.6	Combination of different models . . . . .	37
<b>5</b>	<b>Specification</b>	<b>39</b>
5.1	Requirements . . . . .	39
5.2	Design of model . . . . .	40
5.2.1	Layout . . . . .	40
5.2.2	Item storage in the LTM . . . . .	41
5.2.3	Associations . . . . .	42
5.2.4	Remembering . . . . .	43
<b>6</b>	<b>Realisation</b>	<b>44</b>
6.1	Pseudocode . . . . .	44
6.2	Scenario application . . . . .	48
6.2.1	John and Bob at the park . . . . .	48
<b>7</b>	<b>Evaluation</b>	<b>53</b>
7.1	Application in scenarios . . . . .	53
7.2	Meeting the requirements . . . . .	56
7.3	Combined model . . . . .	57
7.3.1	A model for empathy . . . . .	58

7.3.2	Unified model . . . . .	60
7.4	Ethical Reflection . . . . .	62
7.4.1	Distinction . . . . .	62
7.4.2	Misuse of Robot . . . . .	62
7.4.3	Technophobia . . . . .	63
7.4.4	Playing God . . . . .	64
7.4.5	Privacy Issues . . . . .	64
7.4.6	Job Loss . . . . .	65
7.4.7	Privacy Regulation Agencies . . . . .	65
7.4.8	Users . . . . .	66
7.4.9	Municipalities, Shop Owners and More . . . . .	66
<b>8</b>	<b>Conclusion</b>	<b>67</b>
	<b>Appendices</b>	<b>69</b>
<b>A</b>	<b>Model overview table</b>	<b>69</b>
<b>B</b>	<b>Scenarios</b>	<b>73</b>

# 1 Introduction

Robots are one of many technological advancements that are likely to define the future. Similarly to other quickly evolved technologies, such as smartphones and the internet, adoption of robots is rapidly increasing everywhere. Robot adoption has already begun in leading countries such as Korea, Germany and Japan [24], and it is likely to only increase throughout the world. This adoption of robots comes in many shapes, as there are different types of robots. In a robotics survey conducted by the United Nations [4], robots were grouped into three main types: industrial service robots, professional service robots and service robots.

A big part of research into anthropomorphism in robots has been about improving the physical resemblance of robots to humans. As a consequence of this research, robots have swiftly improved in this area. Take for example Sophia [28], this humanoid has been widely covered by the media. It is one of the most futuristic examples of a robot with physical human qualities, and visually resembles a human quite closely. However even though Hanson Robotics, the makers of Sophia, also incorporated a significant amount of AI features to make its behaviour more human-like, Sophia is still easily distinguishable from a human in this aspect. This is because human-like behavioural qualities in robots are less explored than their physical counterpart. As a social robot needs its behaviour to be of the same caliber as its physical embodiment [11] (though this caliber needs to be evaluated per situation), the behavioural side needs to be explored more.

The client for this paper is the RISE Research Center, located in Cyprus. RISE wants to create social robots that can be used in many real applications, such as supermarket clerks, elderly care, city guides, office assistants, receptionists, airport assistants and more. Specifically they want to create more empathetic social robots, since empathy is a very important aspect for social interaction with and among humans [3]. Thus, for these social robots to be effective in the mentioned situations, they need to become more empathetic. To achieve (part of) the goal of creating more empathetic social robots, two separate GP's were created. This GP will attempt to see how associative memory can be recreated in a social robot to improve their empathy. While the other GP, done by S. Slebos will try to model the reasoning of a robot in such a way that it helps it in executing empathetic actions during its interactions.

Now as was just mentioned, this paper will try to see how associative memory could be recreated in social robots to improve their empathy. This is because an important aspect that improves empathetic behaviour in social robots, is memory [2][15], and specifically associative/recognition memory. Therefore, from the behavioural side of social robots that needs to be explored more, this paper will mainly look into the question 'How can we recreate associative memory in social robots, to improve their social behaviour?'. Apart from this, also questions such as 'How does associative memory work in humans?', 'How can we make robots learn and remember relationships between previously unrelated entities and concepts?', 'How can we make robots store their encounters with

entities and concepts, to improve social behavior towards this entity or concept the next time they encounter it?’ and ‘What efforts exist that try to imitate perceptual associative memory in robots?’. The paper will also give an attempt to incorporate a model or a new version of an existing model, into a possible application.

## 2 State of the art

This paper will incorporate two types of memory, associative and recognition memory. "Associative memory is defined as the ability to learn and remember the relationship between unrelated items such as the name of someone we have just met or the aroma of a particular perfume" [30]. "Recognition memory involves the capacity to remember familiar stimuli when comparing a novel stimulus with one that is already stored in the memory" [20]. In the context of this paper, these two types of memory are almost identical. Therefore, in this paper they will be used interchangeably, according to how they are used in the articles that will be mentioned.

### 2.1 Differences between single- and dual-process models

Many research efforts that have created models for associative memory exist, with the generally accepted way to model associative memory changing throughout the history of advancements the field has made. And in this field, there has been an ongoing debate for years about which type of associative memory model is superior and more strongly supported by empirical evidence. First, it is necessary to discuss the different types of models there are. Broadly speaking, two types of models exist: Single- and dual-process models. To understand the difference between these models, two terms need to be defined, familiarity and recollection. They are defined by Mahoney as such [23]:

Familiarity permits us to identify something as having been seen in the past (e.g., I know I have seen this key before), but affords no contextual detail.

Recollection permits us to recognize the item and recall additional information about the context in which it was originally encountered (e.g., this is the key that opens the shed behind the house).

Knowing these definitions, the two different types of models can be described with some examples.

### 2.2 Single-process models

First off, single-process models. Single-process models account for familiarity, and assume that if an item is recovered from memory, its contextually related information is recovered with it. This means that remembering an item and its contextual information happens in one go. Also if an item is not recovered, its related information is not recovered either. Many single-process models only have a threshold system, which usually means that an item is either fully recalled or not at all. According to prominent researchers in the field [25] however, this is not a correct depiction of the situation. A more correct depiction would be that when something is recalled (which means it has reached the threshold), it is not always fully recollected. Reaching the threshold simply implies that additional information is recovered, how much is recovered may vary from situation to

situation. Since the creation of the field, many research efforts into creating single-process models of associative memory came to be. Some of these models will now be discussed.

### 2.2.1 LIDA’s Perceptual Associative Memory (PAM) module

One of these efforts is the Perceptual Associative Memory (PAM) module of the LIDA cognitive model [9][29], created by the Cognitive Computing Research Group led by Stan Franklin, of Memphis University. The goal of the LIDA model is to provide a control structure for a mind in an autonomous agent. Within it, they try to integrate as much knowledge about the mind from various fields, to make sure it is as accurate as possible. A key concept of this model is what the creators call the LIDA cognitive cycle. This cognitive cycle can be defined as “everything that happens in the agents’ control structure, to produce its next action” and it is the main building block upon which higher-level cognitive processes such as deliberation, reasoning, problem solving, planning, imagining, and more are built. One cycle consists of multiple phases, these are shown in figure 1 below.

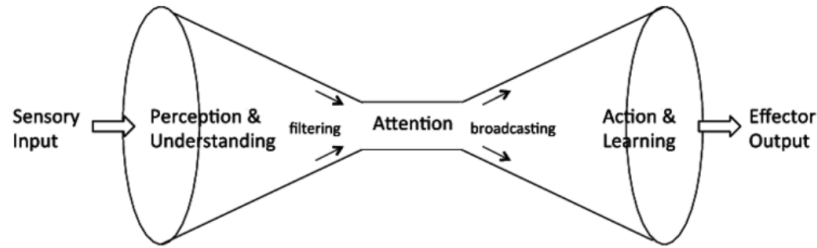


Figure 1: The LIDA cognitive cycle phase diagram

As can be seen in the overview of the LIDA cognitive cycle in figure 2, LIDA has many ‘modules’, each representing a different part of the brain. In this paper, the focus is on their PAM module, which is also present in figure 2.

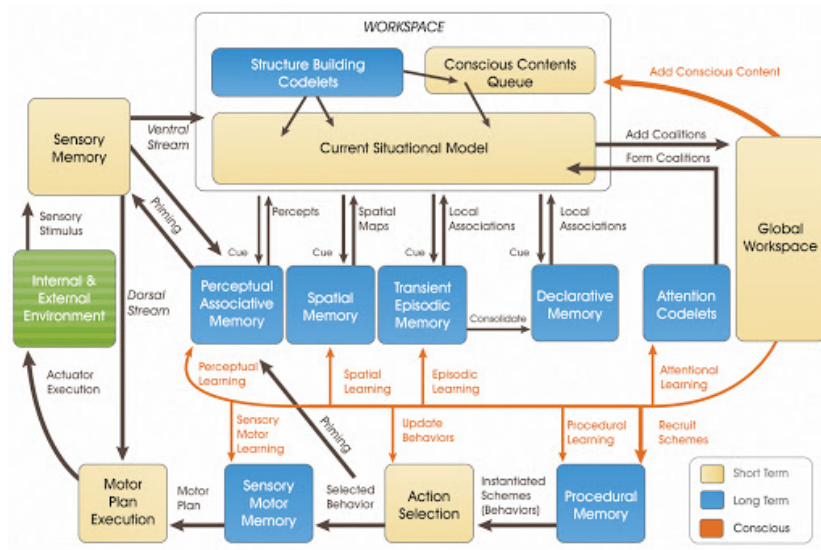


Figure 2: The LIDA cognitive cycle

To understand the role of the PAM module in the LIDA model, first the start of the LIDA cognitive cycle should be explained. This is described by the creators as such:

The LIDA cognitive cycle begins with sensory stimuli, both external and internal, coming to Sensory Memory where it is represented, and engages early feature detectors. The resulting content involves both the Current Situational Model, and Perceptual Associative Memory. The latter serves as recognition memory, producing a percept that is made available to the Current Situational Model. Using both the percept and the incoming content, together with remaining content which has not yet decayed away, the Current Situational Model continually updates itself by cueing Perceptual Associative Memory, Spatial Memory, Transient Episodic Memory and Declarative Memory, and using the returning local associations.

The forwarding of a percept from the PAM to the Current Situational Model, is where the actual associating the way humans do it happens. The PAM receives sensory stimuli from the Sensory Memory, and then by extracting features from these stimuli, puts out a percept. A percept would in this case be a feeling, emotion, action, event, concept, category, etc., and these are recognized by combining certain features of the incoming stimuli, that the PAM then connects to a percept based on previous encounters with similar stimuli. The CCRG team has also created a LIDA framework to express the model in terms of software. They provide exercises along with the framework to help understand the LIDA model, and to explain how to start developing with the framework itself. The

process of going through some of the exercises is described in the next section below.

## Working with the LIDA software framework

The CCRG team incorporated many exercises to help understand the model better, all divided over different sections. While working through the first few exercises, they help you get familiarized with the file layout and the GUI of the framework. Then after the basics, the exercises start to delve into the actual LIDA model, its modules and how they can be and are implemented in the framework. In whole, the tutorial runs multiple simulations with a certain autonomous agent containing a partial or full implementation of a mind according to the LIDA model. These simulations will be putting the agent in specific situations and giving it input, to see what reaction, and thus also output, it will give. The situation that a simple LIDA agent is put in in the tutorial, is one where it has to recognize a blue circle as being blue and a circle, and the same for a red square. Before running simulations, changes, additions and removals will have to be made to certain files in the agents' structure to make it work for a specific assignment the tutorial gives. This is done through Apache's NetBeans IDE, shown in figure 3 below.

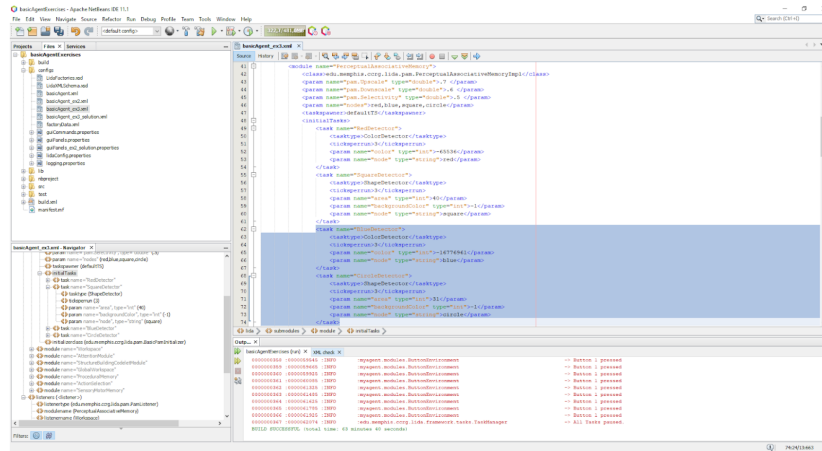


Figure 3: LIDA's software framework opened in Apache's NetBeans IDE

In figure 3 one can see a file layout and navigator on the left side of the IDE, and an opened 'xml' file with console on the right side of the IDE. Figure 3 shows the 3rd exercise of the tutorial, which is adding the missing feature detectors for the colour blue and the shape of a circle to the agents' 'xml' file. After that, when running the simulations, the agent should then be able to respond to the input of a blue circle as well as the input of a red square (which it already could before ex. 3). After changing the necessary files, the rest is

all done by simulation, and the results of these simulations are shown in the frameworks' GUI. What the GUI looks like is shown in figure 4 below.

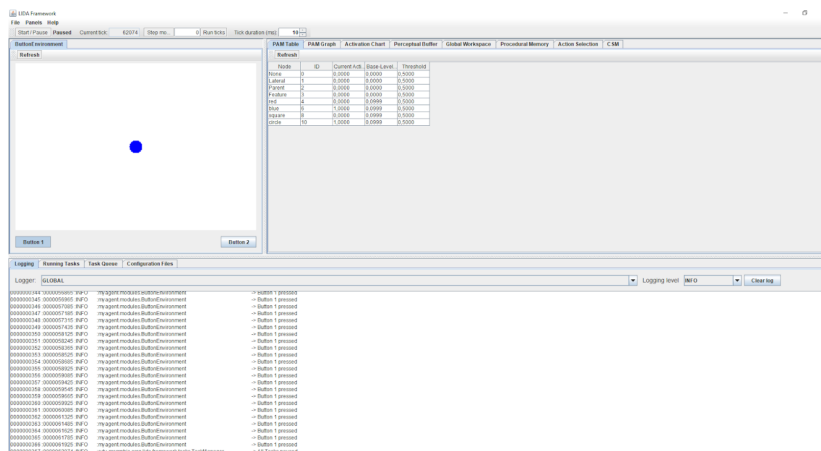


Figure 4: The LIDA software framework’s GUI

This GUI has many different parts, which are used for controlling the simulation. The box on the top left contains the ‘Environment’. This provides all the input to the agent, which in this case is either nothing, a blue circle or a red square. On the bottom is a console which keeps track of what is happening during each moment in the simulation, and the top right box contains all the different modules of the LIDA module that are initialized in the agents’ structure. The agent for the 3rd exercise has the modules Environment, Sensory Memory, PAM, Workspace, Current Situational Model, Attention Module, Global Workspace, Procedural Memory, Action Selection and Sensory Motor Memory. These modules are used by the agent for performing the task of recognizing if the input is nothing, a blue circle or a red square. As one can see in fig. 4, the input coming from the Environment is a blue circle. And when looking at the top right box, where in this case all the different nodes of the PAM are shown in a table, judging from the current activation levels of the ‘blue’ and ‘circle’ nodes, it is clear that the agent recognizes that the input is indeed something blue, and a circle. This recognition comes from the highlighted code of fig. 3 that was added as a part of the 3rd exercise. With this highlighted code, 2 feature detector nodes are added to the PAM module. One is for detecting the shape of a circle, the other for detecting the colour red. Each node has a base-level activation, a threshold and a current activation level. The base-level activation measures how useful a node has been in the past, the current activation level shows the relevance of a node to the current situation. The current activation level of the ‘blue’ node would be around 1.0 if the Environment is giving a blue circle as input to the agent. If the Environment then gives a different input to the agent, the current activation level of the ‘blue’ node would slowly decay, as it loses its relevance to the current situation more and more.

## Perceptual Associative Memory in LIDA

In the LIDA model, PAM is implemented as a slipnet [8], based upon the Copycat paper by Hofstadter and Mitchell [18]. This slipnet has many nodes, and each node may represent feature detectors, a category, a person, a concept, an idea, etc. Links connecting these nodes to each other represent relations between nodes, such as category membership, category inclusion, and spatial, temporal or causal relations. Autonomous agents sense their environment, using sensory modalities such as vision, olfaction and audition. Thus autonomous agents must also have primitive feature detectors, to identify important aspects of the incoming stimuli. In the LIDA model, these primitive feature detectors constitute the nodes of the lowest depth in the slipnet. A single primitive feature detector could for example detect an edge at a specific angle, and multiple of these detectors could combine to form more complex feature detectors, like one that could recognize the shape of a letter. Whenever these detectors detect a feature, they send an activation to nodes and combinations of feature detectors deeper in the slipnet in the form of ‘X is a feature of Y’. This sending of activation goes deeper and deeper until the slipnet stabilizes. At this point the nodes and links with current activation levels above the threshold become part of LIDA’s percept, which gets passed along to the working memory.

As mentioned before, each node has a base-level activation and a current activation level, but each link between these nodes only has a base-level activation, which mostly functions as a weight on that link. Base-level activation is used for perceptual learning, while the current activation level is directly related to incoming stimuli from the internal or external environment. The current activation level decays quickly, say within 2 seconds. The base-level activation however, is different. Nodes or links with low base-level activations decay quite rapidly, while those with high base-level activations decay quite slowly, possibly persisting for decades. Perceptual learning in the LIDA model occurs in two forms, the strengthening or weakening of the base-level activation of existing nodes and links, as well as the creation of new nodes and links. Any concept or relation that is currently in the active part of the mind has the base-level activation of its corresponding node or link strengthened or weakened as a function of the arousal of the agent. Whenever a new individual item is perceived, this results in a new node being created, together with links into it from the feature detectors of its features. An item being recognized as new happens by means of some attention codelet, this is a piece of code that pays attention to specific things that may be occurring. This attention codelet notices multiple features that are activated in the slipnet, but does not have a common object of which they are features yet.

### 2.2.2 Search of Associative Memory

The Search of Associative Memory model [27] (SAM) was created as an improvement of an older model called the Atkinson-Shiffrin model. The Atkinson-Shiffrin model was, among other points, criticized for including the sensory

registers as part of the memory, therefore the SAM model improved on this and some other criticisms as well. The SAM model has a short-term store (STS) and a long-term store (LTS). Understandably, the STS functions as short-term memory while the LTS functions as long-term memory. When an entity with this memory model sees an item and encapsulates data from this item in its sensory registers, it first stores this data in the STS. However, the STS has a relatively small capacity. Therefore, each time a new item enters the STS and it has reached its maximum capacity, this new item will replace an item already in the STS. The LTS in this model is responsible for storing relationships between different items and of items to their contexts. The amount of contextual information an item has is related to how much time that item has spent in the STS, while the strength of a relationship between two or more items changes depending on how long these items exist at the same time in the STS.

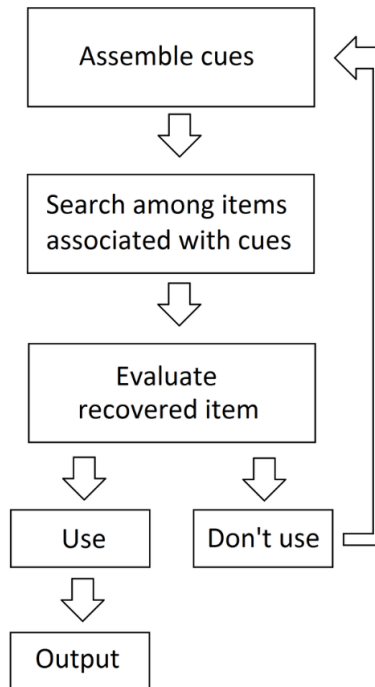


Figure 5: Simplified diagram of retrieval from an item in the LTS under the SAM model

Recalling an item from the LTS uses the concepts introduced for its process, and is depicted in figure 5 above. Each item that is stored in the LTS, has cues related to that item to help the memory remember that item later on. Whenever a situation calls for the remembrance of an item, all the cues related to the item are collected. Using these cues, unconsciously there is determined what area

in the LTS will be searched for the item. Then when an item is recalled, an evaluation will take place to make sure that the recalled item is the one that was meant to be recovered. If not, the recall process will start over again by slightly adjusting the cues related to the item wanted to be recalled.

### 2.2.3 Mnemograms

Another model, that is currently single-process, was created by Bisler [5]. This memory model proposed by Bisler has three different layers. The ultra short-term memory (USTM), the short-term memory (STM) and the long-term memory (LTM). The USTM holds only a few of what the paper calls ‘mnemograms’, the STM has a slightly larger capacity and the LTM has an even larger capacity. These mnemograms are containers that can store many different types of data, and there are many different types. One type of mnemogram that helps the agent associate through its memory, is the class mnemogram. This is a mnemogram that describes a group of mnemograms representing a pattern. Every mnemogram in the memory is linked to others through what they call associations, relating them to each other through the data they store. There are many different types of associations and each association has a weight to indicate its importance. This weight decreases over time, and increases each time it is used for remembering something in a useful way. New mnemograms are first stored in the USTM, while older ones are transferred to the STM and even later to the LTM.

In his paper, Bisler tries to improve the behavioural abilities of an autonomous agent, by giving it a biologically inspired memory. He gives it a simple version of a brain, certain sensors and a memory. He then simulates a basic environment where the agent roams free, and has to distinguish between particles that are food, and particles that are non-edible. At the start of the simulation, the agent has empty memory registers and has no information about the particles. Then after roaming around in the simulation a bit, it encounters different types of particles. Through these encounters, it learns the identifying features of each particle, and stores this in its memory. After a while the agent has enough information about each type of particle to start recognizing the type of a particle when it encounters one. From this moment on, the agent will try to avoid the particles it recognizes as non-edible and only eat the food particles.

## 2.3 Dual-process models

Then, onto dual-process models. Dual-process models have a very different take on how associative memory works. These models propose that recognition memory incorporates both familiarity and recollection as separate processes [23]. These two processes usually either work simultaneously, or on demand. This means that, in contrast to single-process models, it is not only one process working to find an item or its related information. In these models, familiarity is usually modelled based on signal detection theory concepts [12]. Which means that when sensory stimuli are coming in, the process of familiarity will compare

these stimuli to signals currently residing in the memory. And if it finds a (close enough) match, it recalls this item. Recollection on the other hand, is modelled as a threshold process. Which means that in most dual-process models, this process is only active if there is enough (usually sensory) information to conduct a deeper search in the memory than familiarity does. This also means that if the threshold is not reached, the process of recollection fails. Which as you can imagine, is a realistic way of modelling memory. As even if we do remember an item in the memory, we don't always remember all the contextual information of that item. The core assumptions of different dual-process models are often quite similar, however, they still have important differences and still make conflicting predictions about the functioning and neural substrates of the underlying processes of recognition memory.

### **2.3.1 Java-based Associative Memory**

The Java-based Associative Memory model [26] (JAM) is a cognitive model of associative memory designed to understand the way humans hold a conversation, so that agents can keep a more meaningful conversation with people. In the JAM model, there are many many nodes with concepts in them. These nodes are what is currently inside the memory itself, and the concepts in the nodes are the actual memory items. Each node has many associations with other nodes, which is what relates different concepts to each other. Each node also has an activation value, which is the likelihood that the node will currently be activated. The activation of a node can happen in two ways. The first is an external stimulus (e.g., you see an item in the memory), which directly activates the node that contains the item you are seeing. The second is through spreading activation. Every node that is activated spreads part of its activation to neighbouring nodes that it is associated to, through a formula that is described in the paper. This spreading activation process is how this model performs association.

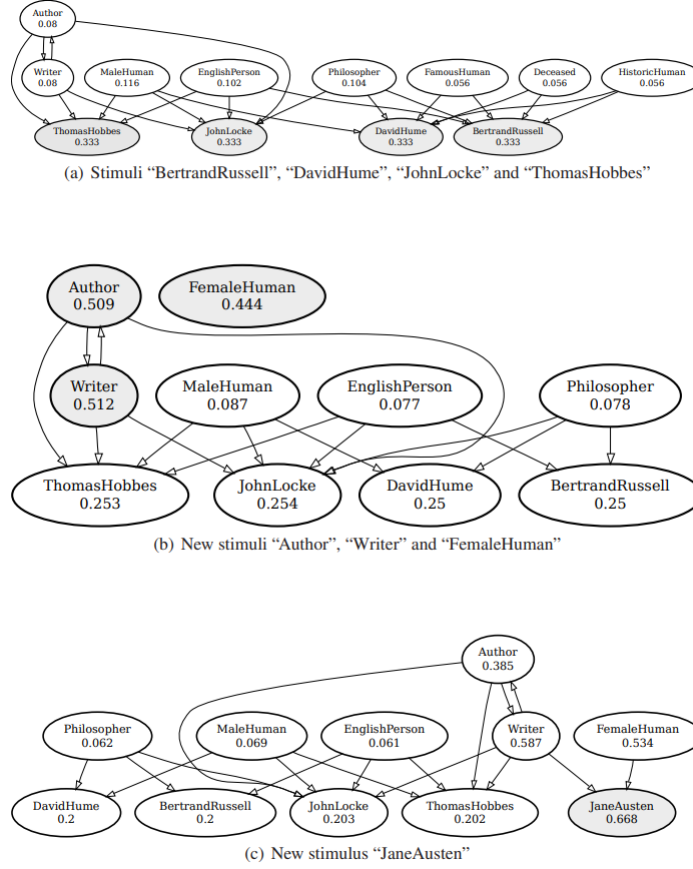


Figure 6: Different types of stimuli as they would appear in the JAM model

### 2.3.2 Dual-Process Signal Detection (DPSD)

The paper [31] in which this model is thought of, first describes the different reasonings behind recognition memory models. It then proceeds to explain why single-process models might not be a good representation of the way it is represented in our brains. After this, a dual-process model is proposed, dubbed the Dual-Process Signal Detection model. According to the DPSD model, items in the memory have a certain ‘memory strength’. The process of familiarity can find items with a higher memory strength more easily than ones with a lower memory strength. This is because items with a higher memory strength will have a stronger presence in the memory than others. Familiarity is modelled as a signal detection process, while recollection is threshold based. With a threshold for the recollection process, this means it can fail. If the recollection process does not reach its threshold, the wanted memory item will not be

recovered. While if the recollection process does reach its threshold, additional contextually related information about the wanted item is recovered. Though additional information is recovered, this does not mean that all the contextually related information about the item is recovered. As mentioned before, this was a common misconception of how associative memory was thought to work in single-process models.

## 2.4 Evidence for dual-process models

Generally, single-process models stem from older research, while dual-process models are more recent. Therefore, it should come as no surprise that the dual-process models are currently more supported by the literature. Yonelinas, a prominent researcher in the field, describes this very well in his paper where he reviews 30 years of related research [32].

First, studies have indicated that familiarity is a faster process than recollection. In a test of speed, participants were usually able to make accurate distinctions between studied and non-studied items, which according to the dual-process theory is the process of familiarity. While under the same conditions, participants struggled with recollecting specific information about said items at a similar speed, which according to the dual-process theory is what recollection does [16][17][13]. The fact that the two ways of remembering have different processing speeds is important, because it shows that they don't use the same process of searching for and returning memories.

Second, throughout different papers it is shown that familiarity and recollection produce different ROC's. ROC's are functions that stem from signal detection theory (SDT). SDT is frequently used to evaluate recognition memory performance, and ROC functions are one way of measuring this performance. Memory functionalities that use the same process, should produce very similarly looking ROC functions. However, the shape of the ROC's for familiarity and recollection changed very much across different conditions. ROC's have been one of the main points of evaluation for every recognition memory model. So the fact that these two functionalities showed very different shapes of ROC is crucial and strongly suggests that there are two different memory processes at play in recognition memory [10][21].

Third, recollection and familiarity show clearly different electrophysiological correlates [6][7]. Event related potentials (ERPs) recorded on the scalp during tests of familiarity and recollection showed a clear distinction between these two functionalities. Although these results do not show which regions of the brain are used for associative memory, it is important to recognize that this does show that there are at least two separate brain processes involved.

Fourth, recollection is more heavily impaired than familiarity by certain brain injuries. An example of this is that patients with amnesia exhibit significantly greater memory impairments in the process of recollection than of that in familiarity [1][19]. This is another crucial result to come out of this field, as it clearly illustrates that these two functionalities use different regions of the brain.

Based on these four pieces of empirical evidence, most researchers in the field currently believe that dual-process models have come closest to representing the way associative memory works in humans. However, this field is still very much developing, as new insights into the human brain come to light every day. Because of this, no definitive conclusions about whether dual-process models are really an accurate way of representing associative memory can be drawn. Nonetheless, for now they can be considered as the type of model that currently best models the way associative memory works in the human brain.

### 3 Method

This GP covers a wide range of topics and areas of research. This is because to correctly create a model of associative memory that could be used in a social robot, many different aspects and views on the same subject need to be combined into one. There are many reasons as to why making a social robot more empathetic is difficult, and the fact that an interdisciplinary approach is needed, is one of them. Using an approach that only takes one or two areas of research into account when making a model for a social robot, is likely to lead to an end product with many faults in the other areas that could have been prevented. Thus for this GP to have a realistic end product, it was necessary to understand the different aspects that could make a robot's memory function the way we wanted it to. Which meant that orientation in subjects that were almost completely outside the scope of the Creative Technology curriculum was needed. To make sure that the orientation into these topics was going in the right direction, a regular check-up with the supervisor was used.

#### 3.1 Orientation

As mentioned previously, creating a model of associative memory for a social robot to improve how empathetic this social robot can be, requires an understanding of many different topics. Research into these topics was mostly done in parallel to each other, but for the sake of clarity they will be described one by one. First, an understanding of what a social robot is, what it can do and how they should be improved with the end goal of this project was needed. For this, general research into robotics, what types of social robots there currently are [28] and what efforts currently exist to improve them [11] was done. Then, empathy [2][3] and different topics about the memory of a social robot were researched. This is mostly where a lot of different fields come together in an interdisciplinary way. Different fields such as Computer Science, Psychology, Ethics, Electrical Engineering, and more all play a role in finding the right answers to questions in these areas of research. So to understand these subjects not only from one perspective, each discipline was researched to some extent. During this research, answers were sought to questions such as what the memory of a robot should look like, how it could be based on the way memory works in humans, how memory can help to create empathy in social robots, which ways of modelling memory in robots have been created throughout the years, and more.

#### 3.2 Associative memory research

After the general orientation phase, papers that are more directly related to associative memory were looked into. Topics such as what is associative memory, how it works in humans and how (associative) memory can help social robots become more empathic [15] were researched. Out of this research, the conclusion came that associative memory is indeed a big factor in improving the empathy

of a social robot. Giving a social robot memory, also helps in letting the people interacting with the robot connect with the robot more easily. It makes sense that if someone were to interact with a social robot on multiple different occasions and the robot itself never remembers them after an interaction, this would start to feel weird and does not make these people think that the robot has any empathy for them. However, if someone interacts with a social robot on multiple different occasions and the robot does remember them and the previous interactions they have had with this person, this can create a sort of social bond that is very normal between humans but less so between humans and robots. This is a big part of why memory is an important aspect that should be incorporated into robots. A social robot could mention previous interactions they have had with this person, or just use certain information from them that is then useful in a new interaction with this person.

This phase in the research is also where all the different associative memory models were found. At the start, as many models as possible were gathered and then all given an attempt to understand how they worked. One by one, a sizable number of models was gathered and generally understood. Each model was usually quite different from the previous ones that were found. To keep a general overview of each model, first a table was made with the most important aspects of each model. This table is shown in Appendix A. This research played the biggest part in shaping Chapter 2, as this talked about all the different models, where they come from and what they are capable of. And it also helped during the Ideation phase, as it helped in shaping many different ideas of how certain models could be used in the end product. Then when this table was finished, the next step was to experiment as much as possible with all the models that had something to experiment with.

### 3.3 Experimenting

After having a general idea of how many models currently exist, what types of models there are and what these models are like, it was time to start time to start thinking about which models would eventually be usable in the final product of this thesis. After the start of this phase and partially in parallel to this phase, a list of requirements for the eventually needed models was made. This list of requirements was then used to see which models were or were not suitable for being used in the end product. For most models this would be done by using the understanding there already was of these models, and working past the requirements one by one checking to see which model meets most or all of them. Then for the models that had software frameworks, these frameworks would be experimented with as much as needed until it was clear whether this framework could be used either fully, partially or not at all in the end product. As expected, there was no software framework specifically aimed at the goal this GP tries to reach. Therefore, the goal was to see if one of these frameworks came close enough to the goal to be able to use it. So these frameworks were explored step by step to reach an answer to the question if they could be used. The first step was getting a basic understanding of how the software worked, and what

goal it was made for. This was done by exploring some of the more generic code, and specifically some of the example applications the creators built into the code. This usually gave a pretty good basic understanding of the framework and its code. Then after this, it was time to start fiddling with the code, to try and apply it in more specific situations that our model would eventually have to be able to function in. If a framework is found that is able to function in one or more of these situations they were placed in, it would be a strong indicator that this framework could eventually be used in the end product.

### 3.4 Learning bottom-up through scenarios

Partially in parallel to experimenting with different frameworks, another method was used to gain a deeper understanding of what we needed to make/create in order to achieve our goal. This GP and the GP of S. Slebos both have the common goal of helping to create more empathetic social robots. So to gain a deeper understanding of how to do this, a bottom-up approach using scenarios was used together with S. Slebos. The idea was to create many different scenarios in which the robot we aim to help create could eventually be placed in. Then, by writing down the different types of interactions it could have and should be able to have, we gain a better understanding of what our models should be able to do and how they should be shaped. By not putting any restrictions or expectations on our models and first take this approach, we start the design of our models with an open mind and the chance of overlooking an important detail in our models is also reduced.

These scenarios were created together with S. Slebos, through a template which was decided on together. The first template that was created is shown in figure 7 below.

Domain: TEMPLATE				
Scenario:			Inputs:	Outputs:
Description:				
Needed contextual information:	Sensory:	Memory:		

Figure 7: First version of scenario template

This first template however, was believed to have too much information for its intended application. Also, the Inputs/Outputs cells were fairly similar to the three information cells. Thus, a second and final version of the template was created in which the Inputs/Outputs cells were taken out. This version is shown in figure 8 below.

<b>Domain: TEMPLATE</b>		
<b>Scenario:</b>		
<b>Description:</b>		
<b>Needed contextual information:</b>	<b>Sensory:</b>	<b>Memory:</b>

Figure 8: Second and final version of scenario template

Through this template, many scenarios were created and thought of that a more empathetic social robot could be placed in. There are however so many applications in which a social robot can be put to use, that these scenarios had to come from different domains to get a broader perspective on what associative memory should be able to help a robot with. These different application domains are shown in figure 9 below.

## Application domains

- **Elderly care**
- **Airport clerk**
- **Museum guide**
- **City centre assistant**
- **Supermarket clerk**
- **Education**
- **Household**

Figure 9: Different applicatoin domains used for scenarios

Each application domain covers a different kind of interaction a social robot could have with a person. And for each application domain, many different scenarios were created. By using the same template for creating these scenarios in different domains, a list of requirements could be obtained of what an associative memory model should be able to do. This could then be applied to any existing or possibly new model to check if it meets this list of requirements. Then if existing models fail to meet the requirements, it can be taken out of the potential models for the end product. And if all existing models would fail to meet the requirements, the list could be used to shape a self-made model into something that does meet them. This way it is possible to make sure the end product ends up adhering to what is expected of it.

## 4 Ideation

### 4.1 Role of the memory model

After research into the many necessary topics was concluded, some design decisions needed to be made as the start of the Ideation phase. The biggest of these decisions was what kind of a role the memory model this GP aims to bring forward should play in the mind of a social robot. Questions such as ‘Will it make certain decisions for the robot?’, ‘Should it produce output actions?’ and ‘Does it know how the robot can behave more empathetically?’ come to mind. After some discussions with A. Kamlaris, the supervisor for this project and S. Slebos, the decision was made together that two different models would be made. One from the GP of S. Slebos and one from this GP. S. Slebos will create a model that receives the current state of an interaction the robot is in, runs this information through its various modules and based on the outputs from these modules it decides on an empathetic output action the robot can execute. Among the various modules in this model, is a memory module. In this memory module is the associative memory model that this GP aims to create. This layout was eventually decided on together with S. Slebos, and are discussed more extensively in Chapters 6 and 7. Because this is the layout that was decided on, many questions that arose about the model of this GP could immediately be answered.

This model will not make decisions for the robot, it will not produce output actions nor does it know how the robot can behave more empathetically. All of this is because memory is not supposed to know and do everything. Just as it is in almost any system (alive or not), the memory is a small part of a bigger structure. It is a tool that can and should be used by other parts in the same structure, but it does not exceed the boundaries of what it is meant to do. Therefore, this associative memory model will exist purely to be ‘used’ by another model like that of S. Slebos and it can not function independently. It will also receive input that the robot is currently receiving, but it will only store this information in such a way that it can be most easily accessed by another model when needed. It will store all the necessary information, connect memories to each other in specific ways, remember similar situations that the robot has been in before and how that relates to the one it is currently one, keep track of all the possible memories the robot can have, and retrieve them when they are needed. It can execute many actions within the model itself and the memories it stores, but outside of the model it has no influence. This means it will play a mostly passive role in deciding how the robot should behave empathetically. In short, it will be used as a tool to help reach a decision on a more meaningful empathetic action that the robot can execute, but the model itself will not decide on this.

### 4.2 Origins of existing models

As discussed in Chapter 2, multiple associative memory models have already been created. These models come from varying areas of research such as cogni-

tive psychology, neuroscience, cognitive science, computer science, biology and combinations of these. Because of this, there is immense variation between most models. Some models are based on the same principles and have certain similarities, this is however the minority. Though these models have many differences as they come from different fields, there is one thing they all have in common. They were created as an attempt at modeling how associative memory works in humans. Now this is a goal that is quite close to the goal of this GP, and these models are very useful to learn from and experiment with. However, most of these existing models usually have some problem when checked if they are suitable for this GP. They are either outdated and no longer supported by current literature, never implemented into something that can be experimented with or are implemented but not with the goal in mind that this GP aims to achieve. This might pose a problem.

Each of the aforementioned problems with existing models can result in them not being suitable for the end goal of this GP. For most types of models, a problem often exists that makes it less suitable. For example, models that are outdated and thus no longer supported by current literature, such as the SAM [27] model, incorporate functionalities that are no longer believed to be how associative memory works in humans. Models that are not implemented into something to experiment with, such as the Dual-Process Signal Detection [19] model, are either mostly or fully theoretical. While they might have incorporated many state of the art functionalities, they are often too new to have been applied into something usable. And lastly, models that are implemented but not with the goal in mind that this GP aims to achieve, such as the LIDA [9] and the JAM [26] models, have existing software frameworks that can be experimented with. However, these frameworks are often created for goals that are quite different from the goal of this GP. This makes it likely that they are either not suitable from the start or require too big of a change to make them suitable.

### 4.3 Single- or dual-process model

As discussed previously in Chapter 2, two main types of models were found. These are the single- and dual-process models. At the end of the state of the art, it was shown why dual-process models are believed to be more supported by current literature in the field. This means that a dual-process model would be the most ideal candidate for use in the end product of this GP. What was quickly discovered however, is that as dual-process models are generally newer and thus more supported by current literature, they often do not have an existing implementation that can be used. This made it virtually impossible for most of the dual-process models to be experimented with, which made them not suitable from the start. While it is unfortunate that most dual-process models are not suitable, it might not even be needed to have such an accurate depiction of how associative memory works in humans for the goal of this GP. The goal of this GP is to create an associative memory model that can be used as a tool to give a robot more options for empathetic actions it can execute. This is

a relatively specific goal, and might not need the most accurate depiction we currently have of human memory. While it is preferable, as it gives the model more scalability to at some point exceed the goal of this GP, it might not be necessary. There are many good purely theoretical models out there, but for this GP models that have been applied in some way are needed. Thus, if there are single-process models or slightly older dual-process models that do have an existing application, these are highly favorable for this GP.

## 4.4 Software frameworks

This brings us to the two main models that have been applied in such a way that they could be experimented with and possibly used for the end goal of this GP. The LIDA PAM module and the JAM model. The PAM module will be discussed first, as it has already been extensively talked about in Chapter 2. Throughout the State of the art research phase, the LIDA model is the only model that was experimented with. Specifically with the software framework that implemented the PAM module’s functionality. The PAM module from the LIDA model is the closest that this model came to something that could be used for this GP, as the name Perceptual Associative Memory also indicates. Therefore, as soon as the software framework for the PAM module was found, it was experimented with.

Its creators set up a website [14] almost entirely dedicated to the LIDA model. On this website are introductions, lectures, papers, tutorials, events, press coverage and more all related to the LIDA model and its separate modules. Through this website they allow you access to the framework, and help you learn how it works through a long tutorial that starts with a very basic form of the model, and adds more modules one by one to give a thorough understanding of how it works. This was tremendously helpful, as it gave specific details on what the PAM module was made to do in the LIDA model and could thus give an idea if it could be used for this GP. Unfortunately, after working through the tutorial and having a discussion about it with one of the framework’s co-creators, it was decided that the PAM module and its framework would not be suitable for the end goal of this GP. The PAM module ended up falling short, as it was created only for very simple environments. After the co-creator pointed out that the PAM module might be too basic for the goal, this was indeed confirmed by working through the tutorial and noticing the simple environments it was made for. This meant that it would not be a suitable framework/model.

Then, back to the Ideation phase. As it was now clear that models with existing implementations were needed, the JAM model was the next best bet. As was mentioned in the JAM paper [26], a Java software framework existed for the JAM model. Therefore, the creators of the JAM model were contacted, to see if they would be interested in sharing their framework and possibly collaborating with this GP. Thankfully, they were kind enough to share the framework and help with getting to know the framework enough to work with it independently. The creators gave access to their GitLab repository, which can be seen in figure 10 below.

**CMM** [Request Access](#)

214 Commits 5 Branches 0 Tags 300.8 MB Files

Cognitive Memory Model

master CMM / +

History Find file Web IDE Clone

No license. All rights reserved

Name	Last commit	Last update
.settings	initial commit	4 years ago
META-INF	tweaks	3 years ago
bin/META-INF	first version of mu mutator	3 years ago
contrib	first version of mu mutator	3 years ago
etc	initial commit	4 years ago
examples	MemConfigs and simulator for data collection 2020	3 months ago
lib	latest version of workload optimizer	3 years ago
logs	filter works on recsys	3 years ago
out/production/CMM/META-INF	no class files	3 years ago
src	Memo 9 Levels in Tuner (not 12, correction of last commit)	10 months ago
test/sessionSOTests	Last additions to code documentation	3 years ago
.classpath	Fix classpath, add jar for vector generation	3 years ago
.gitignore	final tweaks to filter workload experiment	3 years ago
.project	initial commit	4 years ago
CMM.iml	Improvements: many versions of WinStrategy, the best is WinS...	3 years ago
build.xml	initial commit	4 years ago

Figure 10: GitLab repository of CMM/JAM model

The name was slightly altered from JAM to CMM, as it also contained more general parts than just the JAM code. This was quite a big repository, each folder contained many different parts and at first it was unsure which belonged to the JAM model. However after some searching, the right folder was found and the JAM framework could be installed. The JAM framework in the Eclipse IDE is shown in figure 11 below.

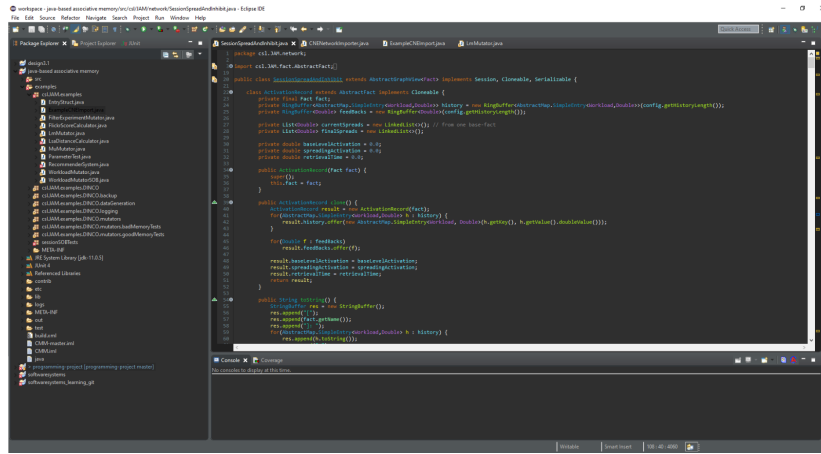


Figure 11: JAM framework in Eclipse IDE

When it was installed, first the general structure of the framework with its many folders needed to be discovered. After some digging, it was found that three main folders were used:

- The 'src' folder. Which as in almost any other project contains all the necessary code for the project to function as necessary
- The 'examples' folder. Which contains many sub folders, each with its own set of example applications that specific part of the code could be used for
- The 'contrib' folder. Which similar to other big projects is used for storing files or software that are needed for the project to function, but might not actually be maintained by the creators of the project.

The 'src' folder was worked through first, through this the basic structure of the framework was discovered. This made it clear which parts of the framework relied on what code, and how this influenced what could be used for this GP. Next was the 'examples' folder. This folder was mainly used as a way of learning how all the separate parts of code in the framework could be used to apply the JAM model to certain situations. First however, some basics of how the model works needed to be understood. This was mainly done through the 'ExampleCNEImport' class, which can be seen in figure 12 below.

```

1 package csl.jam.examples;
2
3 import csl.jam.fact.Fact;
4
5 public class ExampleCNEImport {
6
7     public static void main(String[] args) throws Exception {
8         Network network = new POJONetwork();
9         CNENetworkImporter importer = new CNENetworkImporter(); // All of this is:
10         importer.loadRelationNodes = false; // Create a network with a knowledge base of a specific language
11         network = importer.importKnowledge(network, "/contrib/export5.de.cne"); // Create a network with a knowledge base of a specific language
12         //network = importer.importKnowledge(network, "/contrib/test.cne");
13
14         System.out.println("Vertices: " + Integer.valueOf(network.getVertices().size()).toString());
15         System.out.println("Edges: " + Integer.valueOf(network.getEdges().size()).toString());
16
17         SessionSpreadAndInhibit s = new SessionSpreadAndInhibit(network);
18         FactFactory factory = network.getFactory();
19         POSTaggerLanguage postagger = new POSTaggerSLanguage();
20
21         //List<String> lines = Files.readAllLines(Paths.get("/contrib/illse_except_demo.txt"), StandardCharsets.UTF_8); // Specify the text file that you want to give as input
22         List<String> lines = Files.readAllLines(Paths.get("/contrib/input.txt"), StandardCharsets.UTF_8); //
23
24         int wordCounter = 0;
25         int activeWordCounter = 0;
26         int notFoundCounter = 0;
27         double sumOfActivationProbs = 0.0;
28
29         for(String line : lines) {
30             System.out.println("Line: " + line); // do line by line through the text file
31             String[] words = postagger.tokenize(line, "de", "contrib/lemmatizer");
32             //String[] words = line.split(" "); // do not use tokenizer if line contains non-words (e.g. tags)
33
34             // filter words
35             String blacklist = "contrib/stopwords_german_extended.txt"; // extended: based on most frequent 1000 words, removed nouns, verbs, adjectives
36             List<String> bl = Files.readAllLines(Paths.get(blacklist), StandardCharsets.UTF_8);
37             List<String> bl = new HashSet<String>(bl);
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

Figure 12: JAM framework's ExampleCNEImport class

What this class shows an example of, and what is the basis of the JAM model, is how they use the ConceptNet [22] as a knowledge base to help the model understand interrelations between different words and concepts. ConceptNet is an open source semantic network that is designed to help computers understand the meanings of words that people use. It originated from an MIT Media Lab project started in 1999, and has now become one of the most prominent knowledge bases that can be used to help computers understand language in a more human way. ConceptNet has many parts to it, but quickly summarized it is a massive knowledge graph that contains almost any word that is ever used and more importantly it contains all the relations between different words and how often they are used in connection to each other and in what ways. Through this, computers can input certain words into ConceptNet, and receive words back that are related to those words in specific ways. It utilizes ConceptNet to receive a list of possible words that might be applicable to certain words. The JAM model then uses the words it receives from ConceptNet and extracts the most useful words depending on the context the input words were in. This is the basic functionality of the JAM model and also the functionality that might be useful for this GP.

Back to the ExampleCNEImport class, let's see how it works. The first part of the class is shown again in figure 13 below.

```

public static void main(String[] args) throws Exception {
    Network network = new POJONetwork(); //
    CNENetworkImporter importer = new CNENetworkImporter(); // All of this is:
    importer.loadRelationNodes = false; // Create a network with a knowledge base of a specific language
    network = importer.importKnowledge(network, "/contrib/export5.de.cne"); // Create a network with a knowledge base of a specific language
    //network = importer.importKnowledge(network, "/contrib/test.cne");

    System.out.println("Vertices: " + Integer.valueOf(network.getVertices().size()).toString());
    System.out.println("Edges: " + Integer.valueOf(network.getEdges().size()).toString());

    SessionSpreadAndInhibit s = new SessionSpreadAndInhibit(network);
    FactFactory factory = network.getFactory();
    POSTaggerLanguage postagger = new POSTaggerSLanguage();
}

```

Figure 13: Beginning of the ExampleCNEImport class

What can be seen here is that it first creates an instance of a knowledge base, in the code it's called a 'Network'. This Network is initialized in a specific language, which is done by the 'importKnowledge()' method. This method

imports a ‘de.CNE’ file, which contains all the necessary German (indicated by the ‘de’, but it could be any supported language) words and their relations to other words, basically the actual knowledge base. Then when the knowledge base is imported, the necessary methods are called to get the Network ready for use.

```
// Specify the text file that you want to give as input
List<String> lines = Files.readAllLines(Paths.get("./contrib/input.txt"), StandardCharsets.UTF_8);

int wordCounter = 0;
int activatedCounter = 0;
int notFoundCounter = 0;
double sumOfActivationProbs = 0.0;
```

Figure 14: Input specification

After this as seen in figure 14 above, a certain text file is specified which contains the textual input that will be given to the model. In the case of this GP, this could be a sentence that the person who is interacting has said to the robot. Also, certain variables are initialized for use from this point on. Then as can be seen in figure 15 below, the input is lemmatized and filtered for use.

```
for(String line : lines) { // Go line by line through the text file
    System.out.println("next line: " + line);
    String[] words = postagger.tokenize(line, "de", "contrib/lemmatizer");
    //String[] words = line.split(" "); // do not use tokenizer if line contains non-words (e.g. tags)

    // filter words
    String blacklistFile = "contrib/stopwords-german-extended.txt"; // extended: based on most frequent 1000 words, removed nouns, verbs, adjectives
    List<String> bl = Files.readAllLines(Paths.get(blacklistFile), StandardCharsets.UTF_8);
    final HashSet<String> blSet = new HashSet<String>(bl);

    List<String> filteredWords = new ArrayList<String>();
    for(String w : words) {
        if(!blSet.contains(w) && !w.startsWith("(")) {
            filteredWords.add(w);
        }
    }
    words = filteredWords.toArray(new String[0]);
}
```

Figure 15: Lemmatization and filtering of input

Lemmatization means grouping together the multiple forms of the same words that exist in the input. If the input would for example have the words ‘become’, ‘becoming’ and ‘became’, these would all be lemmatized to ‘become’, the most basic form of the words. This is because ConceptNet is only able to analyze the most basic forms of words. Then after this, all the stopwords that exist in a certain language, in this case German, are taken out of the input. This is because ConceptNet is not able to give any extra meaning to the output using these words, so they might as well be taken out. Then the model performs many actions on this input, using ConceptNet where needed and some math is also involved. Lastly, for each word in the input that ConceptNet recognizes, the JAM model prints out the 20 words that are most likely to be usable in the same conversation as the word that is being analyzed. An example output is shown in figure 16 below.

```

##### Next word (counter: 0) #####
flughafen (VVINF/VERB) -> flughafen
flughafen: 1.6996690255890117 / 3.3993380511780233
flug: 0.0 / 0.8160557731067463
hafen: 0.0 / 0.8160557731067463
verkehrsweg: 0.0 / 0.8160557731067463
zivilflughafen: 0.0 / 0.8160557731067463
hauptstadtflughafen: 0.0 / 0.8160557731067463
airport: 0.0 / 0.8160557731067463
veraltet: 0.0 / 0.8160557731067463
flugzeug: 0.0 / 0.8160557731067463
strecke: 0.0 / 0.3085932024880062
linie: 0.0 / 0.3085932024880062
verkehr: 0.0 / 0.3085932024880062
weg: 0.0 / 0.27206078858299154
refugium: 0.0 / 0.2582808730431094
raumflughafen: 0.0 / 0.2582808730431094
anlegestelle: 0.0 / 0.2582808730431094
schmelztiegel: 0.0 / 0.2582808730431094
heimathafen: 0.0 / 0.2582808730431094
seehafen: 0.0 / 0.2582808730431094
hafenanlage: 0.0 / 0.2582808730431094
anlegen: 0.0 / 0.2582808730431094
next line: flug
##### Next word (counter: 1) #####
flug (VVFIN/VERB) -> flug
retrieval probability before adding: 0.7374972203018961
flug: 1.6996690255890115 / 3.7207092441800116
flughafen: 0.8289859803936981 / 1.8283152321138825
flugzeug: 0.0 / 0.8099593875718449
weg: 0.0 / 0.5955110957247254
gesetz: 0.0 / 0.5811096508918452
abflug: 0.0 / 0.5811096508918452
fliegen: 0.0 / 0.5811096508918452
reise: 0.0 / 0.5811096508918452
hafen: 0.0 / 0.5658283886943223
zivilflughafen: 0.0 / 0.5658283886943223
hauptstadtflughafen: 0.0 / 0.5658283886943223
airport: 0.0 / 0.5658283886943223
charterflug: 0.0 / 0.48826199775504514

```

Figure 16: Output of ExampleCNEImport class

As can be seen in the figure, each word in the input is analyzed one by one. First they are transformed into their most basic form, nothing changes in this example however as they are already in their basic form. Then the word itself and the top 20 most related words are shown with two values. The right value, which is the base activation level, indicates how ‘activated’ a word is without input and the left value, which is the current activation level, is how activated it is now. The closer these are to zero the more activated a word is. These values are also relative to how often a word is generally used in language, so words that are more often used always have both activation levels closer to zero even if they are not being affected by the input. This is also taken into account when returning the most related words.

After this ExampleCNEImport class was explored, the most important functions of the model had already been discovered. Importing a knowledge base, choosing a language, inserting an input file, lemmatizing and filtering of the input file, the actual analysis of the input file and the eventual output the model creates. A good example of how this part of the model can be used in a certain

situation will be given now. Within the framework there is the ‘LmMutator’ class, which aims at analyzing an entire sentence instead of separate words. The hope is that through this class it is possible to let the model see correlations between words in a sentence. If this is possible, this model could be suitable for the goal of this GP. As in that case, it could see correlations between different things the robot could be sensing, and associate different things to this input based on the context of the situation and what the robot has stored in its memory.

Now back to the LmMutator class and the example usage of the model it can give. It analyzes a sentence, and the hope is that through this it can see interrelations between different words in that sentence the same way humans can. A picture of the main() method, which is where most of the functionalities are seen, is shown in figure 17 below.

```
public static void main(String[] args) {
    String database = "/contrib/exports/de.long.cse";
    String blacklist = "/contrib/stopwords/german.txt";
    String lemmatizerCache = "/contrib/german-lemmatizer";
    String ngramFile = "/contrib/in.noun.nm";
    String textfileLong = "/contrib/wikipedia8.txt";
    NGrams nGrams = new NGrams(ngramFile);
    LmMutator mutator = new LmMutator(database, blacklist, lemmatizerCache, textfileLong, textfileLong, nGrams.getVocab(), nGrams.getInverseVocab());

    MemoryModel.MemoryConfiguration s = mutator.getPredefinedConfig();

    HashSet<String> targets = new HashSet<>();
    targets.clear();
    targets.add("regenschirm");
    targets.add("schirm");

    mutator.analyzeSentence(s, "nachmittag person park regen laufen regnet regenschirm", targets);
}
```

Figure 17: Main() method of LmMutator class

As can be seen at the top of the figure, first many files with different file types are retrieved from their necessary folders and initialized into the ‘LmMutator’. This LmMutator combines the Network that was talked about earlier with different necessary components to be able to analyze a sentence. Then, the ‘targets’ are set. These targets are certain words that will constantly have how active they are shown each time a new word is analyzed. This can be done to see if a certain sentence activates a word that would be expected to activate. In this example sentence the words are in German, but the sentence translates to ‘afternoon person park rain walking raining storm’. Now from this sentence, a word that is reasonably expected to activate is ‘umbrella’. Therefore the word umbrella, which in German is either ‘Regenschirm’ or ‘Schirm’, is put as a target word. This way the LmMutator class will always show both the activation levels of these two words. When the sentence is then analyzed, part of the output that is gotten is shown in figures 18, 19 and 20 below.

```

stimulate: person
activation count: 42458
regenschirm: -6.42253260736078 / -6.781237693873051
schirm: -5.935869679213336 / -6.480207534876469
unsympathische: -4.003972041803865
unsympathischer: -4.003972041803865
unterhalt: -4.003972041803865
unterhaltung: -4.003972041803865
unternahmen: -4.003972041803865
unternehmen: -4.003972041803865
unternehmen: -4.003972041803865
unternehmer: -4.003972041803865
unternimmt: -4.003972041803865
unternommen: -4.003972041803865
untreu: -4.003972041803865
untreue: -4.003972041803865
verabredung: -4.003972041803865
verb: -4.003972041803865
vereinsmeier: -4.003972041803865
vergangenheit: -4.003972041803865
vergewaltigung: -4.003972041803865
verkehrsmittel: -4.003972041803865
vermieter: -4.003972041803865
verräter: -4.003972041803865
versammlung: -4.003972041803865
verteilen: -4.003972041803865
verteilt: -4.003972041803865
vertrauensperson: -4.003972041803865
vertreter: -4.003972041803865
verwandtschaft: -4.003972041803865
vorgänger: -4.003972041803865
vorreiter: -4.003972041803865
weltenbummler: -4.003972041803865
wettkampf: -4.003972041803865
widerstandskämpfer: -4.003972041803865

```

Figure 18: First example output of LmMutator class

Now as can be seen in figure 18 above, here the word ‘person’ is ‘stimulated’. This means that it is recognized by the model as being input, and is now being analyzed. The ‘activation count’ below that is how many other words have a changed activation because of the stimulated word, for the word ‘person’ this number is quite high as it is a word that can be used with many other words. Then, the two target words are shown, with their base activation level their current activation level. As is visible, the current activation level does not change much from their base activation level, which means that the target words are not very strongly related to the stimulated word, which in this case makes sense.

```
stimulate: park
activation count: 16238
regenschirm: -6.6443763859759395 / -6.781237693873051
schirm: -6.3433462269793575 / -6.480207534876469
fahrzeug: -3.7121622177263154
garage: -3.7121622177263154
gearbeitet: -3.7121622177263154
gegenstand: -3.7121622177263154
gelegt: -3.7121622177263154
gestartet: -3.7121622177263154
gestellt: -3.7121622177263154
handlung: -3.7121622177263154
hingesetzt: -3.7121622177263154
hingestellt: -3.7121622177263154
hinsetze: -3.7121622177263154
hinsetzen: -3.7121622177263154
hinsetzt: -3.7121622177263154
hinsetzten: -3.7121622177263154
hinstelle: -3.7121622177263154
hinstellen: -3.7121622177263154
hinstellt: -3.7121622177263154
hinzusetzen: -3.7121622177263154
hinzustellen: -3.7121622177263154
leg: -3.7121622177263154
lege: -3.7121622177263154
legen: -3.7121622177263154
legst: -3.7121622177263154
legt: -3.7121622177263154
legte: -3.7121622177263154
legten: -3.7121622177263154
starte: -3.7121622177263154
starten: -3.7121622177263154
stell: -3.7121622177263154
stelle: -3.7121622177263154
stellen: -3.7121622177263154
```

Figure 19: Second example output of LmMutator class

The same goes for other words such as ‘park’ when they are stimulated. As can be seen in figure 19 above, the two different words for umbrella both activate even less with this word. This is mostly because park is a less often used word. However when a word such as ‘rain’ is stimulated, we get the output shown in figure 20 below.

```

stimulate: regen
activation count: 21301
regenschirm: -5.426218660965169 / -6.781237693873051
schirm: -5.125188501968587 / -6.480207534876469
gemüt: -3.8089620374764768
iris: -3.7846829353231852
glatteis: -3.756185725727732
hässlich: -3.756185725727732
hässliche: -3.756185725727732
hässlichen: -3.756185725727732
hässliches: -3.756185725727732
hässlichste: -3.756185725727732
hässlichsten: -3.756185725727732
aufgeregt: -3.6537385690718787
aufrege: -3.6537385690718787
aufregen: -3.6537385690718787
aufregend: -3.6537385690718787
aufregt: -3.6537385690718787
aufzuregen: -3.6537385690718787
beweg: -3.6537385690718787
bewege: -3.6537385690718787
bewegen: -3.6537385690718787
bewegend: -3.6537385690718787
bewegt: -3.6537385690718787
bewogen: -3.6537385690718787
eisregen: -3.6537385690718787
erregen: -3.6537385690718787
erregt: -3.6537385690718787
flüssigkeit: -3.6537385690718787
gerappelt: -3.6537385690718787
geregnet: -3.6537385690718787
hagel: -3.6537385690718787
rappelte: -3.6537385690718787
rege: -3.6537385690718787

```

Figure 20: Third example output of LmMutator class

Now while the two words may not be at an activation of zero, they changed quite a lot compared to the last two times when words were stimulated. Indicating that these two words are more strongly related to the stimulated word, which corresponds to what was expected. This gave confirmation that the model was indeed working as expected, and is able to associate words in a sentence to other words in its database. The unfortunate discovery that came after this however, was that the model was not able to understand interrelations between different words in the same sentence. By using different example sentences and target words, it was found that many target words that were expected to be activated by a certain combination of input words, actually did not. And this ended up being, because the model is just not capable of understanding interrelations between words in the same sentence. This was a discovery that impacted the suitability of the JAM model in such a massively negative way that it ended up not being usable for this GP. This also meant that there were no longer any models with existing software frameworks out there that could be used for this project. Which in turn meant that a new model had to be created. This is where the creation of scenarios came in handy.

## 4.5 Scenarios and model requirements

As discussed before in Chapter 3, scenarios were created to analyze what an associative memory model should consist of, in the context of it helping a robot to execute more empathetic output action in its interactions with people. The template for creating these scenarios and the various application domains that were chosen were shown before. There are two reasons that these application domains were chosen. The first being that these are application domains in which a social robot is very likely to eventually be placed in, when they are advanced enough. Then the second reason they were chosen is that these domains vary immensely in location, context and types of interactions a social robot can have. This makes sure that the requirements that come out of these scenarios satisfy any need that the social robot could have of its associative memory, in all the possible different domains that it can be placed in. This list of requirements will be started off with the two main functionalities of a memory model. The first being that the model should store memories in such a way, that they can be most easily retrieved and the second is that the model should be able to deliver memories to other processes in the memory when they are needed. These two requirements can be assumed as being necessary, as a memory model can not function without them. Then from these two, new requirements will be added one by one depending on if a scenario shows that they are necessary for the model to work in that situation. Multiple scenarios will be discussed to see if they create new requirements that should be added to the list.

First is a scenario where the robot greets people at the entrance of a supermarket, this is from the ‘Supermarket clerk’ application domain and the scenario is shown in figure 21 below.

<b>Domain:</b> Supermarket clerk		
<b>Scenario:</b> Entrance greeter		
<b>Description:</b> The robot would be standing at the entrance/exit of the supermarket. It will give normal greetings to customers the robot has not seen before. But whenever a customer the robot has seen before comes in or goes out, it will try to give a more personal greeting message or a shopping tip if there is a discount that might be interesting to that specific customer.		
<b>Needed information:</b> Weather info, current discounts	<b>Sensory:</b> Computer vision, speech	<b>Memory:</b> Identifying features of customers, past encounters

**Example scenario:**

Action: Person enters the shop with facemask and gloves

Context: Corona pandemic going on

Sensory: Tess

Memory: Familiar, person entered without protection last time

Output: speech/conversation

Exact output action: "Hi Tess! I see you're taking the pandemic more seriously already, that's amazing!"

Figure 21: Entrance greeter scenario

As can be seen in this scenario, the robot needs to be able to recognize cus-

tomers as they walk in and remember past interactions they have had with the recognized customers. These are already two requirements that the model should be able to do, even for the most basic interactions. There are multiple ways of meeting the first requirement. Facial recognition or voice recognition are examples, but these both have flaws. People do not always show their face or speak to the robot. Therefore, the method that was chosen is to store the most clearly identifying features of a customer. These identifying features would then be used to create some sort of ID for the customer, which the current input can be compared to. If the robot would see that customer again after having created an ID for them in its memory, they would repeat this same process of creating an ID for the customer and this ID would be compared to the ID that is stored in the robot's memory. Then if these ID's match, the customer is recognized. As mentioned before, when a customer is recognized past interactions the robot has had with this customer should be remembered. As there are already multiple concepts that need to be stored related to the same person, a memory structure is also needed to keep all these concepts stored somewhere.

This memory structure will store more than these two concepts, as will be discovered with the next scenario.

<b>Domain:</b> Airport clerk		
<b>Scenario:</b> Store guide		
<b>Description:</b> This robot would be situated in the airport area where all the shops are. People can come up to the robot, tell it what they are looking for, and the robot can guide them to a store that has that product. If that product is not sold in the airport, the robot will give a suggestion of a product based on its past interactions and insight of the person who's asking.		
<b>Needed information:</b> Airport layout, store locations, product line of all stores	<b>Sensory:</b> Listen to the person, speech, computer vision	<b>Memory:</b> Past interactions, past solutions

**Example scenario:**

Action: Person comes back from long business trip, asks where the whiskey store is

Context: Evening of work day

Sensory: John

Memory: Familiar, wife likes whiskey and wine

Output: speech/conversation

Exact output action: "Hi John! Unfortunately this airport closed its whiskey store. But may I suggest the wine store that replaced it, your wife likes that too right? The wine store is to your left here."

Figure 22: In-museum clerk scenario

In the scenario in figure 22 above, the robot should be able to guide travelers to the store they are looking for or a related store that they would enjoy too. In the example scenario that is given John asks where the whiskey store is located. The robot has seen John before as he is a frequent flyer through this airport, so through remembering John and using the associative function of its mem-

ory model the robot remembers that John has a wife and that he usually buys something for her when he gets back. The robot knows there was previously a whiskey store here, but that is now replaced with a wine store and through association it also remembers that John’s wife enjoys wine as well. Therefore after all this, the robot suggests that John go to the wine store that replaced the whiskey store he initially wanted to find. By exploring this scenario, it can be seen that memories should be related to each other and somehow connected, so they can be found from the memories that relate to it. This is the third concept that should be stored in the memory structure mentioned earlier. Then, there should be some process that actually associates from a certain memory to find other memories that are strongly related to that memory. This is how the robot found the memory of John’s wife and that she likes wine as well.

These two scenarios were handpicked from a bigger list of scenarios. The reason these two scenarios were picked is because they allowed for a quick introduction and explanation into the various requirements for the model that were discovered through the use of all the scenarios. Of course not all of these requirements were discovered specifically because of these two scenarios, but they provide a simple way of explaining them. The rest of the scenarios that were created together with S. Slebos can be seen in Appendix B. These will however not be discussed here, as the requirements of the memory model of this GP that were discovered through scenarios have already been introduced and explained. There are still more functionalities incorporated into the final memory model that have not been mentioned yet, most of these came from other already existing models and will be discussed in section 4.6 below.

## 4.6 Combination of different models

Certain requirements for the memory model were found through placing a hypothetical social robot in different scenarios, to learn what the model should be able to do. However, the model is not yet complete. Many associative memory models already exist and from these models is where the last requirements for the model came from. For example, almost all models that have a clear layout had two or three different layers of memory. According to most research, three different layers of memory is most supported by the current literature as to how it is in humans. Therefore this will also be a requirement for the model that will be created. Which layer of memory stores what kind of information is mentioned in the paper about ‘mnemograms’ [5], therefore this memory model will have a similar layout. Three layers of increasing size, each with its own functions. There needs to be a central layer of memory that receives data from the current input and data from the memory of the robot, which is also connected to the other model which uses this model as a tool. This is also a requirement for the model. Also as mentioned already, there needs to be some connection between different memories. This is a concept that returns in almost any existing associative memory model as well and they are almost always called ‘associations’. These associations are what connect memories to each other in different ways, to show the many different relationships memories can have between each

other. This is why there should be different types of associations and associations should have different strengths.

All these different requirements that come out of the different existing models, were put into a sketch of the model layout and some of its functions during the Ideation phase. This sketch can be seen in figure 23 below.

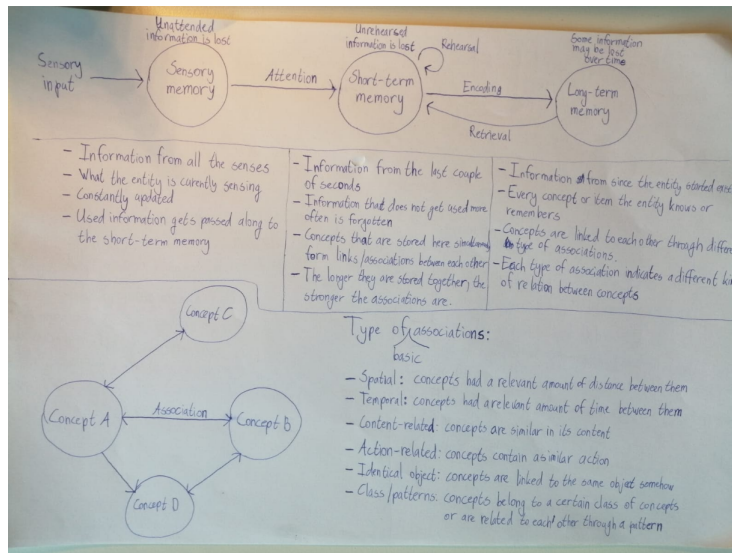


Figure 23: Sketch of existing requirements of memory model

In this sketch, the three layers of memory are given certain names based on how they are most often used in literature. Each layer is already given relatively thought out functionalities and amount of information it stores. When information gets passed on or retrieved from one layer to another is also explained somewhat, but will be discussed more thoroughly in Chapter 5. In this sketch, the memory structures that were mentioned before are called 'concepts'. This name will be changed later on in Chapter 5. For now however, how the associations worked when the model was in this phase will be explained with the word concept indicating the memory structures. Each concept has its memory content, which is the actual memory (e.g., a person) that is stored. Between concepts that are related to each other in some way are associations that point both ways. This association makes sure that the process that does the associating in the memory model can find strongly related concepts to a concept that the process started out from. Each association has a type and a strength. The type indicates the different kinds of relationships there can be between concepts and the strength shows how strong this relationship is.

After this the Ideation phase concluded. In the Specification phase, each requirement will be put in a list and be revisited and thought about more thoroughly. In this next phase the final version of the new model that was created is also presented and explained in more detail.

## 5 Specification

As discussed in the Ideation phase, the associative memory model that will be presented in this Specification phase is primarily conceptual. Even its implementation which is shown in the Realisation phase. This is because there were unfortunately no existing models with software frameworks that were suitable for the goal of this GP. Therefore, this model will be a tool that is supposed to be used by another process or model. This model and its conceptual implementation are also meant as helping hands in a more applied implementation in a future project

### 5.1 Requirements

For the model that will be created in this phase, certain requirements were discovered through different means in the Ideation phase. The requirements of what the model should be able to do to function optimally will be listed below. The model should:

- Store memories in such a way that they can be efficiently retrieved
- Deliver memories to another process/model when they are needed
- Recognize things it has seen before
- Remember past interactions it has had with things it recognizes
- Store all information related to one thing in a memory structure
- Make connections between memories that are related to each other
- Find memories that are most strongly related to a certain memory
- Have three layers of memory, each increasing in size
- Have a central layer of memory that executes most of the actions within the model and is connected to the model that uses this model as a tool
- Have connections between memories with different types and strengths

These requirements vary greatly in importance and functionality, however meeting all the requirements should be the goal. Only when they are all met, will the model most likely work as expected in all the different scenarios that were created. These requirements will now be worked out into its final form and explained how the different functionalities work in the actual model itself.

## 5.2 Design of model

### 5.2.1 Layout

There would be three different layers of memory, Sensory Memory (SM), Short-Term Memory (STM) and Long-Term Memory (LTM). As can be seen in figure 24 below, in this order they have an increasing amount of things they can store.

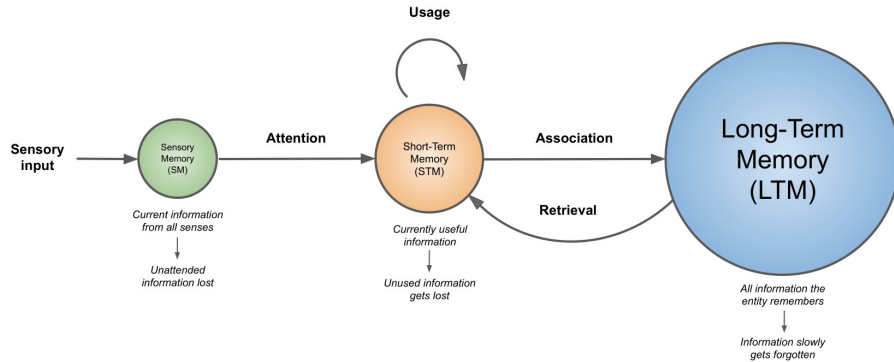


Figure 24: Layout of the different layers of memory

With the LTM storing by far the most, as this is where an entire lifetime of memory is stored. There is no real start or end to these three layers of memory, however they do have a limit as to how much data they can store. The SM stores what all the senses have been sensing the last couple of seconds. The SM sends data to the STM if this data is sensed for a longer amount of time, or if the robot pays attention to it. In the STM, all the latest data from the last couple of hours is stored, and also all the data that currently needs to be remembered from the LTM. The STM sends all the data it contains to the LTM to be remembered, and along with it it tells the LTM how important of a memory it is, how it relates to other memories, etc. In the LTM, all the memory from the entire lifetime of the robot is stored.

Many memories that were once stored in the LTM quickly get forgotten, as they did reach the STM and thus the LTM, but they were not important enough or not sensed for long enough to remember them. Many other memories however are still stored in the LTM, and these memories are needed now and then by the main process in the brain of the robot. In general, whenever the social robot with this memory model needs to think of or remember something, it will be retrieved from the LTM to the STM. And when a memory item is retrieved to the STM, we consider that item to be remembered. Then when an item is in the STM, it can be accessed and used by the main brain process. This is why the STM can be seen as the central part of the model, almost like the CPU of the model. It performs most of the actions, and is the biggest factor in how the model works. The SM can be seen as a buffer for giving data to the STM,

while the LTM is simply a database that the STM uses to make associations and perform other memory tasks.

### 5.2.2 Item storage in the LTM

Memories are called ‘memory items’, as these memory items are not just what memory it contains. Memory items in the LTM are stored in a network. More specifically, a network that has the characteristics of a weighted directed graph. In this network, each memory item has different things related to it. First and foremost it is meant to store its memory content. However, this is not all. Apart from its memory content, each memory item contains these three key things:

1. An ID, made up of the most basic identifying features of the memory content, which is used for recognizing something quickly when it is sensed. The most distinguishing features of an item are combined so that whenever something is sensed, these ID’s can be compared to whatever is being sensed to check if something has been sensed before or not.
2. An individual activation level of the item, which indicates how strong the memory of it is and how long it will still be stored. Activation levels of all items diminish at the same speed, so items with lower activation levels will be forgotten more easily. There are three ways the activation level of an item can increase.
  - When an item is sensed. If this happens its activation level increases immensely temporarily, and is cut down to a lower level when it is no longer sensed. This lower level however, is still higher than what the level was before the item was sensed.
  - When an item is remembered. In this case a memory item is retrieved from the LTM to the STM, to be used in a certain thought process or action involving the retrieved memory item. Any time this happens, the activation level of the item is increased corresponding to how long it stayed in the STM.
  - When an item is associated. This means two things. It means that if a memory process is traversing the memory in search for a certain memory item and that item is eventually found and retrieved to the STM, its activation level is increased the same way when an item is directly remembered. The second thing it could mean, is all the memory items that a memory process traverses past in search for a specific memory item. All the items that a memory process traverses past in its search, are also given a certain increase in activation level. However, because these items are only traversed past and not remembered, this increase in activation is lower.
3. Past ‘encounters’ data entries. Each time that the robot recognizes that it is sensing a human that it has sensed before, it makes an ‘encounter’ data entry in the human’s corresponding memory item to store that it has

encountered that human again. In this ‘encounter’ data entry, it stores everything about the encounter. It stores what the person looked like, at what time they were at the location, for how long, with who, what they were doing, etc. This way, each time that a memory item comes back to the same place where the robot is, the robot will remember their past visits as well. Each part of this ‘encounter’ has its own individual activation level, and this also decays. If something of an ‘encounter’ is not used, it will most likely be forgotten relatively quickly. While if something of a certain ‘encounter’ is used, its activation level will be higher again so that it is stored longer.

4. ‘Associations’ that connect a memory item to other items in the memory. An association is a link between two items that could be described as a directed edge, with a weight, (i.e., association) that points from one vertex (i.e., memory item) to another vertex. These associations indicate that the two items are connected to each other somehow, and the direction of the association makes sure that any process that traverses through these memory items follows the directions that the association is in. These associations could mean any obvious connection such as ‘These two people are father and son’, or something as vague as ‘The last time I tasted this flavour, I was in country X’. Each association also has its own type and weight. The type indicates in what way the two items are connected to each other, the weight indicates how strong that connection is and how long it should remain stored.

### 5.2.3 Associations

There are an unthinkable amount of different types of connections two memory items can have with each other. With some connections being stronger and more important than others. Connections between people because they are family for example, are important and strong connections that should be permanently remembered. But a robot will not realise this unless we make sure it does. Therefore, some types of connections need to be given its importance beforehand. For the robot that we aim to build, this can be done manually. As we aim to place the robot in only one or at least a small amount of different scenarios. Because of this, manually indicating which types of associations should be permanent is still doable. This could also be done through its knowledge base if the robot needs to be applicable in more than one scenario. The robot has a knowledge base which in that case could tell the process that makes associations, which connections are generally more important in life than others. For example, two people being father and son is generally a more important connection than a smell that you associate to a place that you have been before. These things that people learn throughout life, can for the robot be taken out of its knowledge base.

And because there are many different types of connections, we need to somehow tell the robot which types of connections it can make, and when it should

make these. If we wanted the robot to be able to perform in every possible scenario, we would make these types very generalised. However the robot we aim to create, will most likely be situated in one place most of the time. Therefore, the connections that it is able to make can be narrowed down to only the ones that will be used in that specific place. An example of this is given in section 6.2.1, where one scenario is worked out in more detail.

#### 5.2.4 Remembering

Memory items are stored in this network configuration, because items that are needed together once, will most likely be needed together again. Therefore, by connecting associated items together, it becomes easier to find the needed items in the memory through the items they are connected with. The process of remembering an item starts in the LTM at one or more items that are currently being sensed. Then from these items they traverse to a specific item in the memory that is needed in that moment. This traversing is done by following whatever associations these items have, starting at the associations that have the highest weight. Because the weight of the associations is also based on how often certain items are remembered together, it is highly likely that two items with a higher weight association need to be remembered together again. Then from each item that the process started at, it starts moving to other items one by one. It keeps going in the same direction as long as the weight of the association that it is connected to the next item is higher than any one of weights of the associations from the starting item. At some point the wanted item is found, and this memory item is then retrieved to the STM (i.e., remembered).

An example of this could be when someone asks you what the name of your dog is. In that instance you sense the question, and then from the starting items 'name', 'dog' and possibly others, your brain traverses through your memory to find the memory item that contains the answer you are looking for. Then, whenever the needed item is found and is retrieved to the STM a new association with a certain type and weight is created from the item that was sensed, to the item that was remembered. Because apparently there is a connection between these two items in that direction.

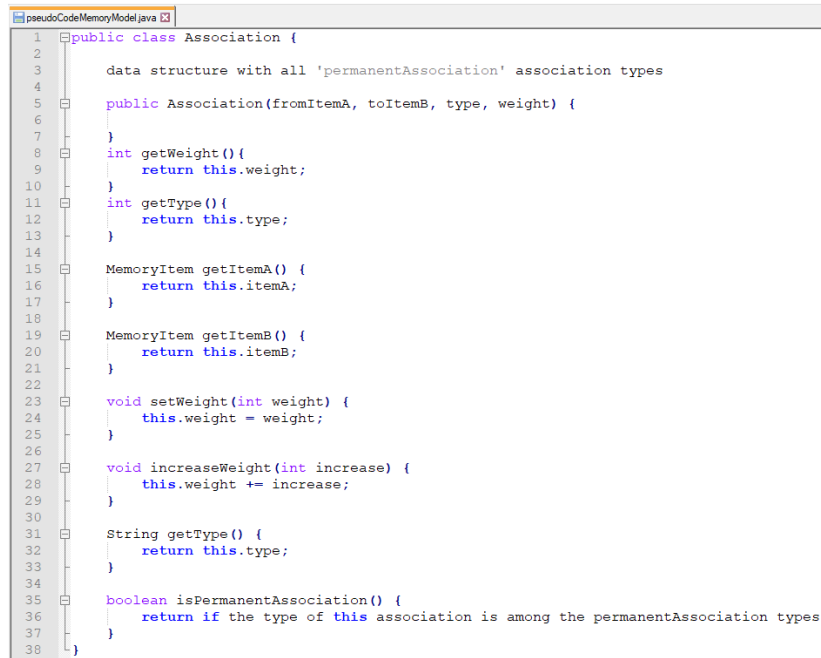
Two memory items can also be associated with each other in both directions. Then there would need to be two separate associations for each direction, each most likely with its own type and weight. This is because the opposite version of an association does not necessarily have the same type or strength. A simple example of this would be the memory items of your bike and the key that opens the lock of your bike. When you see the key, you immediately think of the bike that the key unlocks. However when you see your bike, you could think of going for a bike ride, your feelings toward the bike, any special moments you have had with the bike, etc. Among these associations could be the key that opens the bike, but it would most likely have a very different weight and type than the association from the key to the bike.

## 6 Realisation

To achieve the goal of this GP, some form of implementation needed to be made for the newly created memory model. However, a fully implemented memory model application is very complex and challenging as can be seen by not many existing models even having one. Because of this, much of the time from the Realisation phase was first spent on trying to find and understand an existing software framework that could be used for this GP. Unfortunately this ended up giving no results and an implementation was still needed. As a full implementation into code was likely to be too ambitious of a goal, it was decided to create a conceptual pseudocode implementation of the newly created model. A goal was set to create a conceptual pseudocode which implemented most, but hopefully all of the requirements given in the Specification phase.

### 6.1 Pseudocode

The created pseudocode will now be shown in parts, explaining each class one by one to give a more thorough understanding. First, the Association class will be explained.



```
1 public class Association {
2
3     data structure with all 'permanentAssociation' association types
4
5     public Association(fromItemA, toItemB, type, weight) {
6
7     }
8
9     int getWeight() {
10         return this.weight;
11     }
12
13     int getType() {
14         return this.type;
15     }
16
17     MemoryItem getItemA() {
18         return this.itemA;
19     }
20
21     MemoryItem getItemB() {
22         return this.itemB;
23     }
24
25     void setWeight(int weight) {
26         this.weight = weight;
27     }
28
29     void increaseWeight(int increase) {
30         this.weight += increase;
31     }
32
33     String getType() {
34         return this.type;
35     }
36
37     boolean isPermanentAssociation() {
38         return if the type of this association is among the permanentAssociation types;
39     }
40 }
```

Figure 25: Association class

The Association class in figure 25 above represents the associations that connect related memory items to each other. As mentioned before, each association has a type, a weight, and a direction, each in its own way indicating what sort

of relationship the memory items have between each other. There are getters (methods that retrieve, ‘get’, a specific value of an association) for each variable the association has and setters for the variables that need them. The direction an association is in depends on which memory item is initialized in what spot when an association is created. As mentioned in section 5.2.3, the types of associations will be chosen manually per situation. Among these different types certain types should never be forgotten, an example of this would be a family relationship. In the Association class these types are put into a separate data structure which gives them the ‘permanentAssociation’ status. Then the next class is the MemoryItem class.

```

39 public class MemoryItem {
40
41     data structure that contains all associations to and from this MemoryItem
42     data structure that contains all past encounters with this MemoryItem
43     boolean permanentItem = false;
44
45     public MemoryItem(ID, activationLevel) {
46
47     }
48
49     void createAssociation(itemB, type, weight){
50         associationX = new Association(this, itemB, type, weight);
51         add associationX to data structure
52     }
53
54     void addEncounter(time, place, location, relevantInformation) {
55         add encounter to data structure
56     }
57
58
59     void increaseActivation(amount) {
60         this.activationLevel += amount;
61     }
62
63     void decayActivation() {
64         this.activationLevel -= a set amount of 'forgetting decay';
65     }
66
67     void setPermanentItemTrue() {
68         permanentItem = true;
69     }
70
71     int getActivation() {
72         return this.activationLevel;
73     }
74
75     boolean hasAssociation(itemB) {
76         return if this item has an association with itemB
77     }
78
79     boolean isPermanentItem() {
80         return this.permanentItem;
81     }
82
83     Association getAllAssociations() {
84         return the data structure that contains all associations of this item
85     }
86
87     Association getAssociation(itemB) {
88         return this item's association with itemB
89     }
90
91     Association getStrongestAssociation() {
92         return the strongest association this item has with another item
93     }
94 }

```

Figure 26: MemoryItem class

The four concepts that a MemoryItem should store are incorporated into this class in different ways. Each MemoryItem first gets initialized with an ID that is created by the SM and an activationLevel of a certain level. Then the associations and past interactions that a MemoryItem should store are both stored in

separate data structures. These data structures are automatically updated when a new interaction happens or when a new association is created. This is done through the method which creates a new association and the method that adds a new encounter to the corresponding data structure. The class has a method for increasing the activationLevel of the MemoryItem, which would be called if an item is retrieved to the STM. There is a method for returning the strongest association an item has, multiple methods related to the ‘permanentItem’ status an item can have (which works the same way as the permanentAssociations) and then there are the usual getters and setters.

```

95 public class SensoryMemory {
96
97     data structure with current sensory input
98     int maxItems = X;
99
100    if (input != null) {
101        | extract inputID from data structure
102        | send inputID and related data to ShortTermMemory
103    }
104
105    if (datastructure.size() > maxItems) {
106        | send least important item to ShortTermMemory
107    }
108 }
109

```

Figure 27: SensoryMemory class

Then comes the SensoryMemory class as shown in figure 27 above, the smallest class. It has two main functions. The first is extracting an ID from the data structure which contains the current sensory input that is being perceived and sending this to the STM. The second is deleting any perceived input from the Sensory Memory that was never used if the max amount of items is exceeded.

```

110 public class ShortTermMemory {
111
112     data structure with currently needed items and input received from SensoryMemory
113     int maxItems = X;
114
115     if (receiving data from SensoryMemory) {
116         compare inputID to all IDs of items in LongTermMemory
117         if (inputID != any ID in LongTermMemory) {
118             create new MemoryItem in LongTermMemory
119             make inputID the ID of the new MemoryItem
120         } else {
121             get MemoryItem corresponding to the inputID from LongTermMemory
122         }
123         store a new encounter in the MemoryItem with all received data, addEncounter()
124     }
125
126     for (MemoryItem itemA : all MemoryItems in ShortTermMemory) { // once in a while
127         itemA.increaseActivation(medium amount);
128         for (MemoryItem itemB : all MemoryItems in ShortTermMemory) {
129             if (itemA.hasAssociation(itemB)) {
130                 itemA.getAssociation(itemB).increaseWeight(medium amount);
131             } else if (!itemA.hasAssociation(itemB) && robot recognizes a new association from the possible types) {
132                 itemA.createAssociation(itemB, type, weight);
133             }
134         }
135     }
136
137     if (datastructure.size() > maxItem) {
138         send least necessary item to LongTermMemory
139     }
140 }
141

```

Figure 28: ShortTermMemory class

Then the second and core layer of memory, the ShortTermMemory class. The data structure in this class stores the currently needed items from the LTM and the data received from the SM. Then there are three main functionalities the class performs. It compares the inputID's it receives from the SM to the stored ID's in the LTM. If the inputID is the same as an ID in the LTM, the MemoryItem corresponding to this ID is retrieved to the STM. If it is not the same, a new MemoryItem is created in the LTM with the inputID as its ID. In both cases a new encounter is made with this MemoryItem. The next functionality of the class is to loop through all the MemoryItems currently in the STM and update their activationLevel. Then, the second for loop goes through all the MemoryItems as well. If there is an existing association between the item from the first and the item from the second for loop, its weight is increased. If there is no association between the two items and the model recognizes there could be one, a new one is created. Lastly if the max amount of items is exceeded, the least necessary items are sent to the LTM. The last class in the pseudocode is the LongTermMemory class, which is the layer of memory that stores almost all of the memories.

```

142 public class LongTermMemory {
143
144     data structure with all MemoryItems the robot remembers
145
146     for (MemoryItem item : all MemoryItems in LongTermMemory) {
147         if (!item.isPermanentItem()) {
148             item.decayActivation();
149         }
150         if (item.getActivation() < 0) {
151             remove all associations related to this item and remove the item from data structure
152             continue;
153         }
154         associations_temp_structure = item.getAllAssociations();
155         for (Association assoc : all Associations in associations_temp_structure) {
156             if (!assoc.isPermanentAssociation()) {
157                 assoc.decayWeight();
158             }
159             if (assoc.getWeight() < 0) {
160                 delete association from the item's association data structure
161             }
162         }
163     }
164
165     void associateFromItem(MemoryItem itemA) {
166
167         temporary_data_structure containing the top X most strongly associated items to itemA
168
169         Association currentAssociation = itemA.getStrongestAssociation();
170         Association strongestAssociation = currentAssociation;
171
172         while (temporary_data_structure.size() < X) {
173             if (currentAssociation.getWeight() >= strongestAssociation.getWeight()) {
174                 itemA = currentAssociation.getItemB();
175                 strongestAssociation = currentAssociation;
176             } else {
177                 itemA = strongestAssociation.getItemB();
178             }
179             store itemA in temporary_data_structure
180         }
181         send all items in the temporary_data_structure to ShortTermMemory to be used for creating an output
182         for (MemoryItem item : all MemoryItems in temporary data structure) {
183             item.increaseActivation(small amount);
184         }
185     }
186 }

```

Figure 29: LongTermMemory class

As can be seen in figure 29 above, the first part of this class is two for loops, each going through all the different MemoryItems in the data structure that holds them. For each item that does not have the permanentItem status, first its activationLevel gets decayed. When the activationLevel of an item becomes zero, they are removed from the LTM (i.e., forgotten). Then, the weight of each association (without the permanentAssociation status) that item has with another item is decayed. When the weight of an association hits zero, it is removed from the LTM as well. Then there is the associateFromItem method. This method takes a ‘starting’ MemoryItem as input and returns the top X most strongly related items to that starting item. As these most strongly associated items are very likely to be useful in an interaction, this method can be used to give the robot more options for creating an empathetic output action in its interaction with someone.

## 6.2 Scenario application

Now that the general functionalities of the pseudocode has been explained, an example of how it works will be given in a more detailed scenario.

### 6.2.1 John and Bob at the park

This scenario was picked from the many different scenarios to be worked out in more detail. In this scenario, the social robot who is given the name Billy is

situated at a park in a city. A father John and his son Bob is who the robot interacts with in this scenario. It first sees John and his son together in the park and then John alone the next day. For this specific location the robot is situated in, a specific list of different association types has been made. These are the types of associations the robot will pay attention to when. This may however not be an exhaustive list of associations it would actually need in real life, as it is merely to give an example of how the pseudocode would work.

Example association types in this scenario:

- Family relationships (eg., John isFatherOf Bob)
- Carrying (eg., John isCarrying a dog)
- Wearing clothes (eg., Bob isWearing a raincoat)
- Type of vehicle (eg., John cameWith a car)
- Type of activity (eg., John isReading a book)

#### First meeting - John and Bob together



Figure 30: First part of park scenario.

This scenario will now be discussed step by step:

- Billy sees John and Bob for the first time  
The information of John and Bob is in the SensoryMemory class. As the robot is seeing people that aren't normally there, the input is not null. The robot extracts two inputID's, one for John and one for Bob and sends these to the ShortTermMemory class. From the ShortTermMemory, the

two inputID's are compared to all the IDs the robot currently has in its LongTermMemory. Because the robot has never seen John or Bob before, two new MemoryItems are created in the LongTermMemory class, each with John or Bob's inputID as the ID of their corresponding MemoryItem. After these two items are created, a new encounter is stored in both, which by storing the time and location they were at, indicates that John and Bob were in the same place at the same time.

- Bob is playing on the swing  
When the robot sees that Bob is playing on the swing, Bob's ID and the ID of the swing in the park are extracted in the Sensory Memory and sent to the ShortTermMemory. As the robot has seen them before, their MemoryItems are retrieved from the LongTermMemory. Then, because the robot sees these two items interacting with each other, there is an association created between them.
- John calls Bob 'son'  
When the robot hears this, a new 'isFatherOf' association between them is immediately created indicating they are father and son. When making this association, it is indicated that this is a family relationship, and is thus an association that should not be forgotten. This is indicated as the association being a 'permanentAssociation'. This status will make sure this association's weight will remain constant, while other associations' weights will decay and eventually be forgotten.
- John and Bob came with a car  
Because of this, between the memory items for John, Bob and car there will now be new 'cameWith' associations created. Also, that they came with a car will be stored in the 'past encounters' data structure.

## Second meeting - Seeing John alone

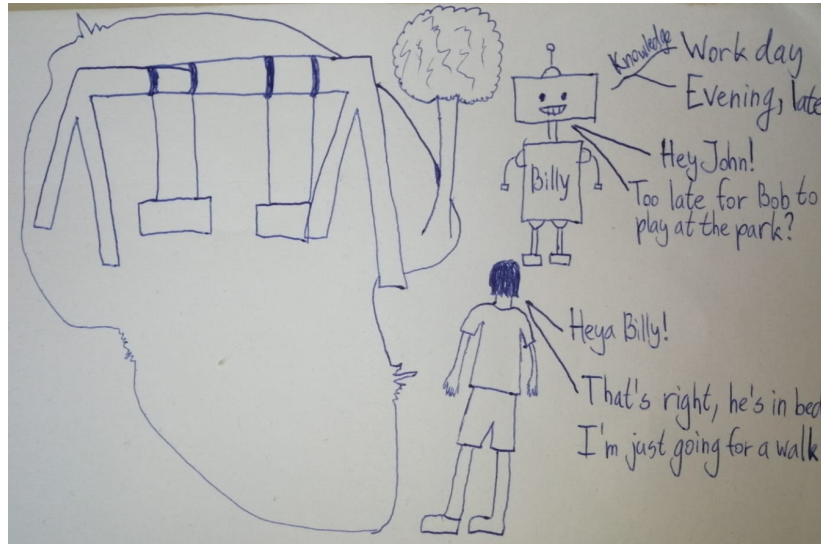


Figure 31: Second part of park scenario.

- Billy sees John

In the SensoryMemory John's ID is extracted and sent to the ShortTermMemory. Then from there, this ID is compared to the ID's of items in the LongTermMemory. When a match is found, John's MemoryItem is retrieved to the ShortTermMemory.

From here on out, most of the actions the robot has to perform are outside of the scope of actions the associative memory model this GP has created can make. A model that is however capable of executing the necessary actions, is S. Slebos' model. These are actions such as producing output actions and thinking about which output action to make and they are shown in the list below.

- Produce greeting sentence: "Hey John!"
- John works hard and could be blowing off steam after a work day → He might appreciate some talking
- *Start thinking about a personalized sentence*
- Using the related information, Billy realizes John is now here without Bob
- Billy knows that its late on a workday and Bob is a young kid → Bob most likely in bed
- *Personalized sentence found*

- Produce output sentence: “Too late for Bob to play at the park?”

However for certain actions in this list that S. Slebos’ model needs to execute, it requires the memories from the model of this GP. Exactly for this reason, that for almost any scenario both of the models are needed in unison, it was decided to combine the models into one. This way, the combined model can execute actions that both separate models could while taking necessary information from each other to execute these actions as well. The combined model that was eventually created and S. Slebos’ separate model will be discussed more extensively in Chapter 7.

## 7 Evaluation

As the model and its implementation are created, they should now be evaluated through different means. First, the model will be applied in four scenarios from different application domains to see if it would work in these varying domains. Then, the requirements shown in Chapter 5 will be held next to the functionalities of the pseudocode to see if it meets all the requirements in the list. Lastly, S. Slebos' model will be quickly introduced and then combined with this model into a more unified one.

### 7.1 Application in scenarios

The first scenario the model will be applied in comes from the 'Supermarket clerk' domain.

<b>Domain:</b> Supermarket clerk		
<b>Scenario:</b> Recipe assistant		
<b>Description:</b> The robot would be situated at the beginning of the supermarket. This way when customers enter the supermarket, they can immediately go to the robot for assistance with picking out a recipe. They could enter what they would like to eat and what ingredients they already have or want to use. The robot would then give a suggestion that is as personalized as possible.		
<b>Needed information:</b> Recipes, store layout, current discounts	<b>Sensory:</b> Hearing, speech, computer vision	<b>Memory:</b> Identifying features, past encounters

**Example scenario:**

Action: Person asks what they can make with their leftover rice

Context: Before dinnertime

Sensory: Tess

Memory: Familiar, likes spicy curries

Output: speech/conversation

Exact output action: "You like spicy curries right Tess? Well chicken breasts are on discount right now, so you could make that! Chicken is located next to the cheese section, and you can find a spicy curry mix in aisle 2."

Figure 32: Recipe assistant scenario

The scenario shown in figure 32 above, is one where the robot helps customers in the supermarket to pick out a recipe that is suited for them and the ingredients they already have or want to use that day. Customers can approach the robot and enter certain ingredients they want to use or already have into the robot's terminal. Then, the robot can suggest a recipe based on input the customer has entered and previous experience the robot has with that customer. In the example scenario, a customer comes in before dinnertime and asks the robot what they could make with their leftover rice. The robot first recognizes the customer and retrieves their corresponding MemoryItem to the STM. It then

processes the customer's request, now with the contextual information in their MemoryItem. The robot associates from the MemoryItems corresponding to rice and dinner, and quickly comes upon the MemoryItem that corresponds to curry and recognizes that this customer especially likes spicy curries for dinner. It checks the necessary ingredients in most spicy curry recipes to see if there is a discount on any of them and notices that chicken breasts are on discount. This information is then sent to the model that produces the output actions, which means this model has done its task.

The next scenario is one from the 'Museum guide' application domain and is shown in figure 33 below.

<b>Domain:</b> Museum guide		
<b>Scenario:</b> In-museum clerk		
<b>Description:</b> This robot could serve the function of keeping an eye on the museum indoors; while also being able to provide useful and meaningful information on the current expo's.		
<b>Needed information:</b> Environmental data, information on the current expo and its individual properties	<b>Sensory:</b> Computer vision, Speech, maybe an interface would also do	<b>Memory:</b> On relatable facts to the questions asked, past interactions, relating to this person's behavior throughout the museum.

**Example scenario:**

Action: Someone called john is not understanding the art object in front of him, so asks the clerk

Context: afternoon, quiet,

Sensory: John, interested

Memory: quiet => time to explain, art is by artistX, artistX is parody artist, painting is parodyOf otherpainting

Output: speech/conversation

Exact output action: "John, this is an artwork by artistX, he often tries to mock with other artists in making parodies; maybe look at the painting opposite of it; thats what its based on"

Figure 33: In-museum clerk scenario

In this scenario, the robot is situated inside a museum to keep an eye on the museum while being able to provide interesting information on the current expo's to the visitors. The museum's visitor can come up to the robot to ask a question about an artwork when they are interested in knowing more or possibly do not understand something. This is exactly what happens in the example scenario. A visitor mentions that he does not understand the artwork in front of him and asks the robot to explain some more. The robot does not recognize the customer, but it does recognize the artwork that the visitor is talking about and retrieves its corresponding MemoryItem. This information is enough for the model that produces output action to decide on the next actions for the

robot, so the information is passed on and this model's task is complete. This is a scenario where not many of this model's functionalities are needed, but it can still assist the model that produces the output actions.

Another scenario is one from the 'Airport clerk' domain, specifically one where the robot is situated behind an information kiosk in the airport.

<b>Domain:</b> Airport clerk		
<b>Scenario:</b> Information desk		
<b>Description:</b> This robot would be situated behind an information kiosk in the airport. People can come up to the robot as they could to any information kiosk clerk and ask it questions.		
<b>Needed information:</b> Airport layout, general airport information, current flight information	<b>Sensory:</b> Listen to the person, speech, computer vision	<b>Memory:</b> Past interactions, past solutions

**Example scenario:**

Action: Person asks where they can check-in for their flight, one suitcase with them

Context: Holiday

Sensory: Tess

Memory: Familiar, always has two suitcases

Output: speech/conversation

Exact output action: "Hi Tess! Only one suitcase this time? For your flight you can check-in at gate G10, which you will find if you follow the yellow line!"

Figure 34: Information desk scenario

In the scenario, which can be seen in figure 34 above, travelers can come to the information kiosk to ask the robot airport related questions. Which is exactly what a traveler does in the example scenario. The traveler asks the robot where they can check-in for their flight and shows the robot their ticket. The robot first recognizes the traveler and retrieves their MemoryItem. From the past encounters it is noticed that this traveler always has two suitcases with them, but now only one, this information is stored to be sent to the other model later on. The robot then identifies which flight they are taking from their ticket and checks the current flight information that it remembers. It finds the corresponding flight among the current flight and the gate that it is located. The model then sends all this information to the model that produces the output actions.

Then the last scenario that the model will be applied in. This is a scenario in the 'City centre' domain.

<b>Domain:</b> City centre		
<b>Scenario:</b> Hotspot information		
<b>Description:</b> The robot would be situated at a specific tourist hotspot. It would give tourists who come by more personal answers to questions that might not be on general signs.		
<b>Needed information:</b> General city information, city history information, information about the touristic hotspot	<b>Sensory:</b> Hearing, speech, computer vision	<b>Memory:</b> Past encounters, past questions and answers to those questions

**Example scenario:**

Action: Person asks the most interesting fact that most people don't know about the hotspot

Context: Holiday

Sensory: Tess

Memory: Familiar, is interested in history

Output: speech/conversation

Exact output action: "Hi Tess! Something most people don't know about X, is that Y happened to it in YEAR."

Figure 35: Hotspot information scenario

In this scenario which can be seen in figure 35 above, the robot is situated at a tourist hotspot. People can come up to the robot and ask for answers to more specific questions than the ones that are on general signs. Or it could possibly even completely replace signs like this. In the example scenario, someone asks what the most interesting fact is that most people don't know about the hotspot. The robot first recognizes the person and retrieves their MemoryItem to the STM. The model then sees that this person is often interested in the history of things, which might be good to know for giving an interesting response. This and other possibly useful information is then sent to the model that produces an output action.

## 7.2 Meeting the requirements

Throughout the design process of the model and its implementation, a list of requirements was created to keep track of what the model should be able to do. This list will be presented again below, to walk through each point one by one to show if and how the requirement is met.

- *Store memories in such a way that they can be efficiently retrieved*  
This requirement is mostly met by the LTM. This memory layer stores all the memories the robot still remembers throughout its lifetime. Each memory has its own MemoryItem with the four different concepts it stores and each item is connected to other items by means of associations, which allow memory retrieval processes to more easily find the item they need.
- *Deliver memories to another process/model when they are needed*  
The model retrieves memories based on current sensory input it receives.

Therefore, the memories that are being retrieved are very likely to be useful in interactions. These memories will be delivered to another process/model when this model has finished its processes.

- *Recognize things it has seen before*  
The SM extracts inputID's based on the current sensory input and then sends this to the STM. The STM compares the inputID to the ID's that are in the LTM and when there is a match something is recognized.
- *Remember past interactions it has had with things it recognizes*  
When something is recognized, the corresponding MemoryItem is retrieved from the LTM to the STM. In this MemoryItem each encounter the robot has had with it is stored in the 'past encounters' data structure.
- *Store all information related to one thing in a memory structure*  
This requirement is met by the MemoryItem class.
- *Make connections between memories that are related to each other*  
This requirement is met through the Associations that are connected between MemoryItems.
- *Find memories that are most strongly related to a certain memory*  
The 'associateFromItem' method in the LTM class meets this requirement. With this method, the top X most strongly related items to a specific item are returned.
- *Have three layers of memory, each increasing in size*  
This is met by the SM, STM and LTM.
- *Have a central layer of memory that executes most of the actions within the model and is connected to the model that uses this model as a tool*  
As discussed before, the STM is the central layer of memory in this model and is also the layer of memory that would be connected to other models. This is shown in an actual model in section 7.3.2.
- *Have connections between memories with different types and strengths*  
The connections are made by the Associations. These each have different types that are chosen based on what the robot's task is, weights that show how strong an association is and even directions that indicate that an association only works one way.

### 7.3 Combined model

As mentioned before, this GP is in collaboration with another GP by S. Slebos. In his thesis, he created a model that produces empathetic output actions in an interaction based on input it receives and multiple different contextual factors. One of these factors is memory and its many functionalities which are incorporated into the model of this thesis. To evaluate S. Slebos' model and the model of this thesis, it was decided to combine the models into a more unified

one. This, as the combined model could perform all the actions needed to help a social robot become more empathetic in its interactions.

### 7.3.1 A model for empathy

The model of S. Slebos will now be briefly explained with its two diagrams. In figure 36 a diagram is shown that has different modules which eventually lead to an action being executed which is shown in the next diagram. To reach a decision on which action to execute, first a list of processes need to be completed. This list is shown below.

1. Input clause
2. Define problem
3. Identify goal
4. Sense environment
5. Understand situation
6. Update states
7. Interpret subject
8. Let 'empathetic module' update goal
9. Define solution space (sets of actions to reach goal)
10. Resolve possible solutions
  - (a) Regain information
11. Check for memory
  - (a) Is memory applicable
  - (b) Feed with memory
12. Evaluate possible solutions
  - (a) Best solution
  - (b) Most empathetic solution
  - (c) Quick solution
13. Settle on solution
14. Check with empathetic module if this is empathetic
  - (a) Re-evaluate solutions
  - (b) Try with parallel actions

15. Start actions
16. Solution is in action
17. Keep the loop running and re-evaluate

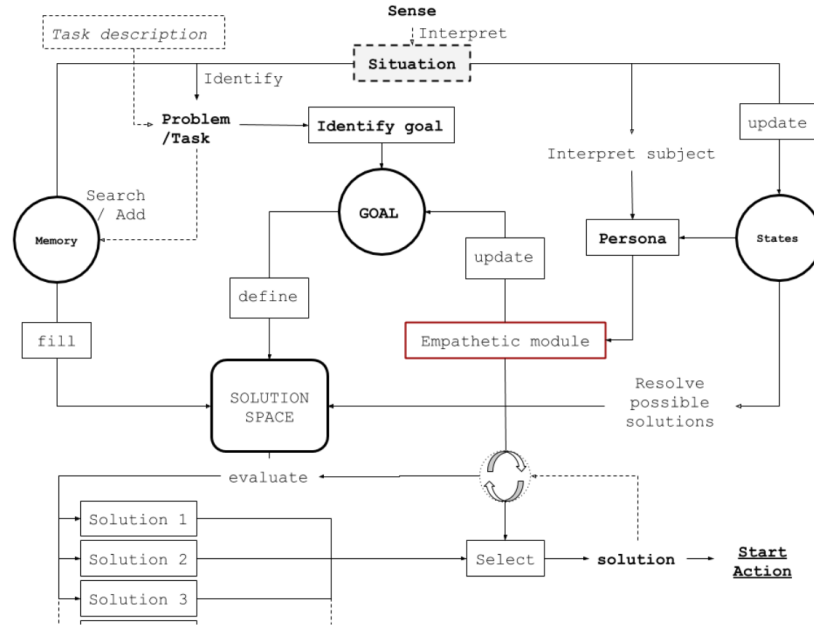


Figure 36: Diagram to reach a decision on executing an empathetic action

The list is taken from S. Slebos' thesis and this is also where a more detailed explanation can be found. Of course a model does not work in this linear way, but this list provides a simple overview of how the diagram can be used. After this diagram has reached a decision on what action to execute, the next diagram which is shown below comes into play.

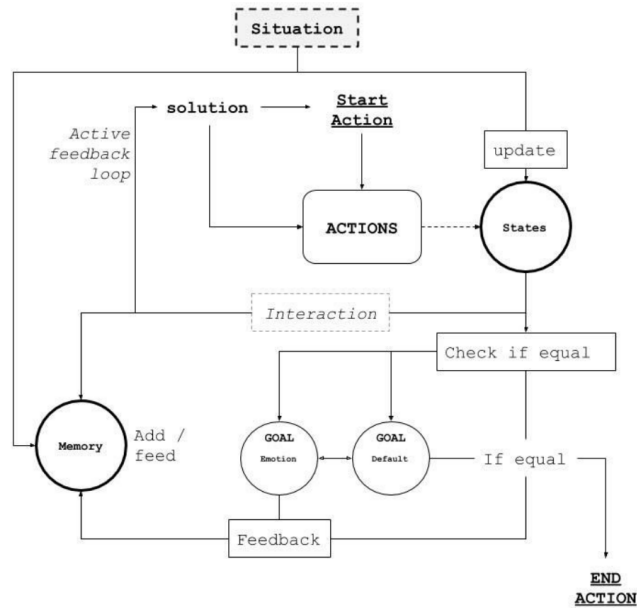


Figure 37: Processes that occur during the execution of an empathetic action

This diagram shows the processes and feedback loop involved in executing an action. These are, in similar fashion to the previous diagram, shown in a list.

1. Continuous active feedback
2. Keep track of emotional state
3. Check if goals are reached
  - (a) Is the default goal reached
  - (b) Is the emotional goal reached (if not, store in memory and learn from this)
4. Store events in memory
5. Feedback to solution space

Once again taken from S. Slebos' thesis where a more detailed explanation can be found. As in the other diagram, the list is linear which is not how the diagram works. But it provides a simple overview of what the diagram does.

### 7.3.2 Unified model

Then as mentioned before, both of S. Slebos' diagrams will be combined with the model of this thesis. They work in the same way as briefly explained in the

previous section, however now the memory module is given in more detail with the model of this GP. This should make both memories complete and overcome the shortcomings they might have had, such as the model of this GP needing another model to decide on empathetic actions to execute. Both combined diagrams are shown in figures 38 and 39 below.

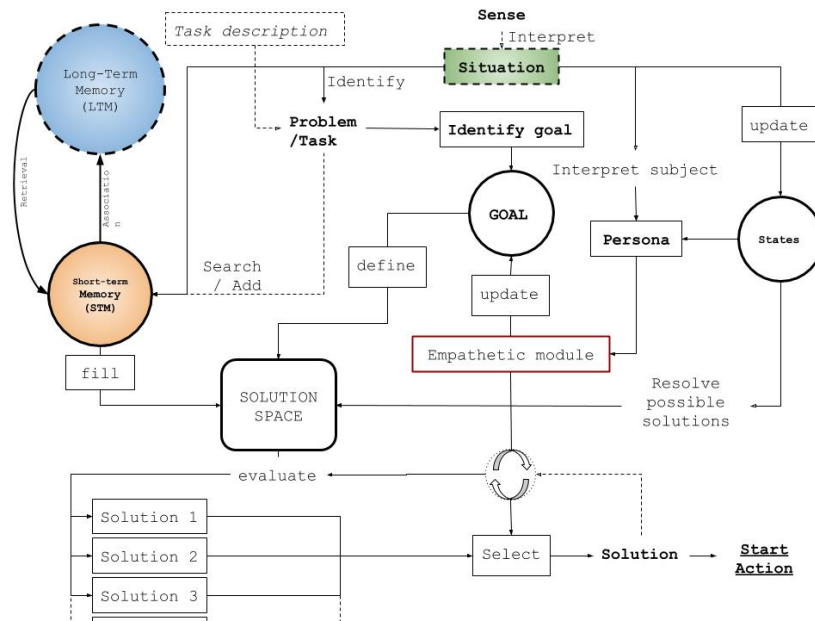


Figure 38: Combined diagram for deciding on an empathetic output action

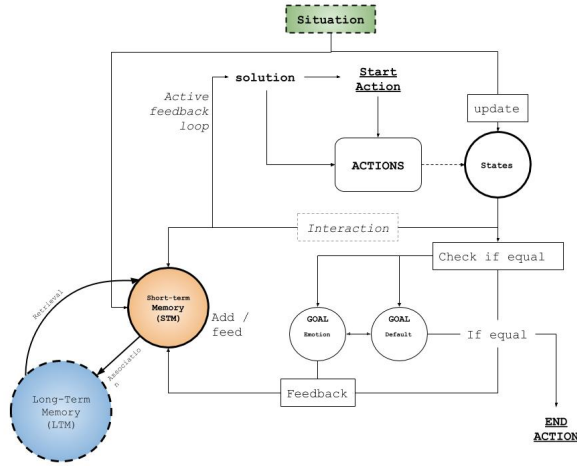


Figure 39: Combined diagram for executing an empathetic output action

## 7.4 Ethical Reflection

This thesis is supposed to be a building block towards a more concrete implementation of an associative memory model into a social robot. Which means that this topic should be revisited in the future, to build and improve upon it. For this reason, certain ethical issues that might become a problem when coming closer to a concrete application will be listed here. These ethical issues are taken from the final ethics report that was written for the Reflection II course.

### 7.4.1 Distinction

For some people, keeping a clear distinction between robots and humans is an important aspect of robots that helps them separate between what beings are 'alive' and 'not alive'. This is something that has been debated in the robotics field for a long time now. The other side of this however, is that for social robots to have meaningful interaction with people, it is exactly the opposite. In that scenario it should look more like a human, otherwise we don't see it as a valid social partner. A solution could be to decide how human to make the social robot depending on how heavily it will interact with people.

### 7.4.2 Misuse of Robot

The misuse of any property that is placed in a public place will always pose a possible problem. Almost anything that's worth something and is in a public place, is at risk of destruction, stealing or other misuse. Robots unfortunately are no different.

## **Hacking**

The social robot this GP aims to put in a public space in the future, could be hacked. There are enough people with malicious intent out there that could aim at hacking the social robot. This could be for many reasons. Control of the robot, seeing what it can do, gaining information about the maker, gaining private information of the people the robot has talked to that might be in its memory, and more. When the robot is being built, strict considerations need to be made security wise, to be sure that hacking of the robot is very hard and thus very unlikely.

## **Stealing**

Stealing of the robot or parts of the robot is also something that should be considered. It is very likely, that a robot of this caliber would need some powerful processors, quality design parts or other materials that are worth something. This means that it is possible for someone to want to steal the robot or some of its parts. Therefore, the robot should be stationed in such a way that makes it hard to steal as a whole, possibly even attached to something if it's task does not require it to move. To prevent the stealing from separate parts, the most valuable parts should be protected by some measure like being surrounded by impenetrable materials or another solution.

## **Destroying**

Destruction of property on public grounds is almost inevitable. One of the worst things that could happen to a robot that is destroyed on some level, is that it's still functional but its safety mechanisms aren't. This would mean that it could still move and perform actions, but the mechanisms that were implemented to prevent the robot from performing any harmful or problematic actions are rendered inactive. This should however be solvable, by safety mechanisms separate from each other, which can also prevent the robot from certain actions if other safety mechanisms fail because of destruction. The robot will of course also have its own cameras to see who is performing the destruction, it could possibly even be guarded by other cameras around the robot.

### **7.4.3 Technophobia**

Technophobia is a slight problem for any project aiming to advance technology in what some could consider a futuristic way. Not everyone enjoys technology moving forward in a high pace, and this is something to think about when placing a high tech project in a public place. If the social robot would for example be placed in a supermarket, possibly not all customer would enjoy the robot being there and could even be scared of it. Especially when these robots start to actually behave like humans as well, it could be possible that many people will be scared of them and the technology that created them. As many people are not yet ready to experience a robot that is very much like us humans.

People being scared of technology often leads to the misuse or destruction of this technology, as they don't know what they are dealing with. A solution to this, is actually easier than it looks. Technophobia often stems from not knowing enough about the particular technology the person is afraid of. Therefore, by educating all the customers in the supermarket example of what the robot is for, why it is there, how it was made and more, it should become less scary and more mundane to people.

#### **7.4.4 Playing God**

A slightly religious implication to think about, is that you could view this research as playing God. In a way this research is trying to dissect nature, and recreate it digitally. We are recreating ourselves in 1's and 0's and you could see that as slowly making nature obsolete. Now of course this is taking it to the extreme, but in the end this could be what it boils down to. At some point we might not need to reproduce anymore, because we have ourselves to thank for being able to recreate entities just like us without it. It is unsure if this will happen, and if this project is even related to that problem, but it is something to think about.

#### **7.4.5 Privacy Issues**

The fact that the robot will interact with people and store almost every interaction into its memory, means that some privacy issues will arise.

##### **Data Storage**

All the data of these interactions with people will probably be stored in its physical memory. This means that technically, this information is all in a public space 'available' to be stolen or damaged. For unknown reasons, people could end up wanting to steal or destroy whatever possibly valuable information is stored in its memory. Because of this, some security measures preventing this should be installed. The data could be stored in the cloud, or whenever the robot is being tampered with it could lock its memory somehow. Or its memory could be encased in something hard to penetrate.

##### **Data Usage**

Another privacy issue that could arise, is rightful use of personal data. Through its interaction with people, it could have some sensitive personal data stored in its memory. This sensitive personal information should only be used what the data is meant for, not for unrelated commercial gains, possibly like companies who like to know more about people to use it for marketing. According to experts, a likely scenario is that the researchers, creators of the project and other privacy organisations will follow current privacy guidelines like the GDPR in the building of social robots.

#### **7.4.6 Job Loss**

The fear that technologies that automate certain jobs ends up in job loss, has been around for centuries. Currently many people are worried about how robots are likely to take over human jobs, as it has already happened in many job types. The reason people are worried, is because robots are a cheaper and more stable workforce. Through the improvement of robots, they will get more and more human-like. The more human-like and better quality these robots are, the faster they will be put into the field to replace workers that are currently fulfilling certain job positions. This is almost inevitable and is one of the great challenges that comes with AI and robotics. By researching the goal of implementing associative memory into robots to make them more empathic, we come scarily close to replacing quite some jobs that right now still require humans. This could be a massive social disruption in the coming years. The moment robots become more human-like, there are many fields in which these robots could be put to work. However according to research done by the OECD, the amount of jobs that will be automated soon by robots, are lower than people fear. There are jobs that are likely to be automated soon, but this

#### **7.4.7 Privacy Regulation Agencies**

For this GP to be successful, during the development and deployment of the eventual social robot that is the goal, privacy regulation agencies should be quite heavily involved. The collaboration with these agencies is of utmost importance, as this robot will most likely gather some amount of personal data from people. For this to work it is very important that this data is taken care of in an appropriate, rightful and lawful way. If this project approaches privacy regulation agencies at some point, it is very likely that they are willing to cooperate or at least assist in any way they can to help make sure that this project will properly adhere to the privacy laws. It is very important that this will be done, even if it is not in the early stages of the project. These agencies can do a great deal of harm to the project if privacy laws are not properly followed. They could shut the project down entirely, if it turns out that the way data is dealt with is open to risk. Because implications of not properly adhering to the privacy laws can be catastrophic to the project, especially nowadays with the GDPR, extra measures need to be taken to make sure these problems are seriously looked at. Privacy should not only be a serious issue to work on within the company. Because if this is done, there is the possibility that everyone will at some point agree that they have taken proper measures to keep the privacy issue under control. However, it could very well be that an agency that is meant to keep track of this does not agree with the general consensus within the company. If outside opinions are requested too late, the possibility arises of having wasted a lot of time and money into a product that needs to be drastically changed.

#### **7.4.8 Users**

There are many factors at play that influence how willing and appreciative the users are towards a robotic system like the one that this project eventually aims to create. One of these factors is that the users should be able to trust the robots and what the robot does with their data. If the users of the robot do not trust it and what it is doing with their data, then they will be very reluctant in interacting with it. They will be especially reluctant in interacting in a way that gives the robot the personal data it so desperately needs to become more personally sociable. This is because in general, people want to know what happens to the data that certain systems collect of them. Therefore, the less transparent that a system is about what it is doing with their data, the harder people find it to trust that system. The easiest and most obvious way to solve this, is to be very transparent with the user about what data it uses. It should show the user what data the robot is sensing, what data from everything that it senses it is storing, and which of that data is personal. The robot would most likely have some sort of way that the user can accept that its data is being used to make the robot more personally sociable. This would be asked to each new user before they use the service the robot provides them. And if users are not satisfied with the terms after using the service the robot provides, they can always indicate that they no longer wish for their data to be stored.

#### **7.4.9 Municipalities, Shop Owners and More**

The robot could be placed in various places that are not under the projects control, which means that there will be an owner of the place the robot is situated in. To make sure that there is as little resistance as possible, thorough talks will have to be had with the person or institution responsible for the area the robot is placed in. This could be a municipality if it is placed in a park, or a shop owner if its placed in something like a supermarket. By having these talks with the ones responsible for the area, it can hopefully be avoided that they will be unhappy with the situation and possibly even work against the project at some point. The ones responsible for the area should be informed about what the robot is, what it can and will do, and how it can influence the people in that area which they are responsible for. By hopefully making sure they know whats going on, any problems regarding these people or institutions can be avoided.

## 8 Conclusion

The goal of this thesis was to use an existing or create a new associative memory model to help a social robot behave more empathetically in its interactions with people. This social robot would be placed in many different associations, which is why the model should be able to function in more than one specific situation. In Chapter 2 it was shown that all existing associative memory models are based on how associative memory works in humans. Also, there are two main types of associative memory models, single- and dual-process. Among which the dual-process models are most supported by current literature in the many fields that this thesis entails.

For the implementation of a model, it was decided to experiment with every relevant model that had a software framework. This ended up being two different models. Both frameworks were promising in their own ways, but in the end fell short of being able to reach the goal of this thesis. For this reason, a new model needed to be created from scratch. To create this new model, first multiple scenarios from different application domains were created in which the robot could eventually be situated. These scenarios were used in a bottom-up approach to learn what the model should be like. Requirements for the model were created based on these scenarios, literature on associative memory and existing models. In this phase, it was decided that the model and its implementation would both be conceptual and the model would be a tool that should be used by another model. The model would function purely as a memory model, which meant that another model should decide on how to behave empathetically using the contextual information the model from this thesis can provide.

After this, the model was slowly built up and created according to the requirements that were made. A layout was created with three layers, each layer with its own size and many functionalities. For the conceptual implementation, pseudocode was created from the models layout, functionality and requirements. The model and its pseudocode was then evaluated through different means. First, they were applied in four scenarios from differing application domains to see if they would function in these situations. Then, each of the requirements that were made earlier on was walked through to make sure the model and pseudocode both met all the requirements. The model was then combined with the model S. Slebos created to make a more unified model. Lastly, the model and the purpose of this thesis was ethically reflected upon.

To finish the conclusion, the main and sub research questions that were created at the start of the thesis will be discussed and reflected upon. The main research question "How can we recreate associative memory in social robots, to improve their social behaviour?" was answered through various means. A conceptual model and implementation was created, which could be improved and built upon to eventually become the actual associative memory model of a social robot. Another answer to this question could be in one of the currently existing models. If they are eventually improved enough to be able to reach the goal, they can also be used as a way of modeling associative memory for robots. Lastly, the requirements that were made could also be used to create another

model. The sub research question (SRQ) "How does associative memory work in humans?" can not be given a definitive answer. However, as aforementioned the dual-process are currently the closest research has gotten to understanding how associative memory works in humans. The SRQ "How can we make robots learn and remember relationships between previously unrelated entities and concepts?" is answered by the means of the 'associations' that were created in the model and pseudocode. Each time a a new relationship is discovered, this relationship is stored by creating an association between two memories. The SRQ "How can we make robots store their encounters with entities and concepts, to improve social behavior towards this entity or concept the next time they encounter it?" is answered by means of the 'past encounters' that are stored by the model. And lastly the SRQ "What efforts exist that try to imitate perceptual associative memory in robots?" is answered by the research that was done in Chapter 2 and the different models and papers that were created on the topic.

# Appendices

## A Model overview table

Model	Model Type	Model description	Advantages	Disadvantages	Implementation
<b>SAM</b>	Single-process	Short-term and long-term store (STS, LTS). Sensory input forwards items to the STS. If the STS is full, this new item replaces an old one, the old one goes to the LTS. All items in LTS have context and links to other items in it, that grow stronger the longer an item is in STS.	Although concepts may be old, they are well defined and explained. Concepts are more easily translatable to code than other models. Model is most likely good enough for use in empathic robots that perform slightly more simple tasks.	Has been heavily criticised by the psychological/neurological field. Only familiarity is modelled as a process, not according to current literature. Only a theoretical psychology/neurology model, has no implementation yet.	Theoretical experiments

<b>Mnemograms</b>	Single-process (so far)	<p>Mnemograms are data units that can hold different types of data and there are many types. Ultra short-term memory (USTM) holds only a few mnemograms of current sensory input, short-term memory (STM) has a larger capacity and long-term memory (LTM) even larger. An ‘association’ links two mnemograms together and there are many different types of associations (spatial, temporal, content related, etc).</p>	<p>Many kinds of links between items allow for different relationships instead of just strength of the relationship. Clear description of concepts and functionalities. Concepts are more easily translatable to code than other models.</p>	<p>Doesn’t seem to rely as heavily on related research in the field. Initially created for a specific implementation, hasn’t been explored for further use</p>	<p>Simulation in 3D environment with agent</p>
-------------------	-------------------------	--	--	--	--

<b>Java-based Associative Memory</b>	Dual-process	<p>Cognitive model of associative memory designed to understand the way humans hold a conversation, so that agents can keep a more meaningful conversation with people. Nodes, with concepts and associations. Each node has an activation value, which is the likelihood of the node getting activated. This can happen in two ways: One is external stimulus, which senses the concept of the node in some way. Another is through spreading activation, every node that is activated spreads part of this activation to neighbouring nodes.</p>	<p>Made to be implemented in Java, there already is a library. Specifically made for conversation with people. Clear description of concepts and functionalities</p>	<p>More a model for human-like conversation and speech patterns than a memory system.</p>	JAM Java library
--------------------------------------	--------------	--	--	---	------------------

<b>SDM</b>	Single-process	SDM is a content-addressable memory technique, that allows large patterns of data to be stored. And later retrieving them based on partial matches with patterns from current sensory inputs.	Has been extensively researched, iterated and implemented in software. Some form of the model already exists in certain robots.	Requires large storage capacity due to its 'sparse' nature. Needs relatively large search cues to recover memories through associations, unlike us humans.	LIDA PAM module, LIDA framework, CMattie
<b>DPSD</b>	Dual-process	Recollection as threshold process and familiarity as signal detection process are separate processes that simultaneously function as a memory. If an item cant be recovered with the familiarity process, a deeper search of recollection that usually takes longer will try to recover the item.	Recollection and familiarity are separate and simultaneously working processes. Created according to the latest empirical research.	Vague to no explanation on how the recollection and familiarity processes work. Only a theoretical psychology/neurology model, has no implementation yet.	LIDA PAM module, LIDA framework, CMattie

## B Scenarios

<b>Domain:</b> Elderly care		
<b>Scenario:</b> Social companion		
<b>Description:</b> The social robot would have the task of providing social companionship to an elderly; since the loneliness of the elderly is more and more often considered a problem. The robot would need to provide meaningful conversation relating to everyday problems. In this sense, it is important that the robot makes useful connections and can recall previous interaction.		
<b>Needed information:</b> Context of the environment, personal information, social skills	<b>Sensory:</b> Speech, text, computer vision (to some <u>extend</u> )	<b>Memory:</b> Past interactions, social states, events, up to date events

**Example scenario:**

Action: Breakfast, Elderly enters the room not looking happy

Context: Home situation, morning

Sensory: John, unhappy

Memory: unhappy when slept bad;

Output: speech/conversation

Exact output action: "Morning John! How are you doing, slept bad?"

<b>Domain:</b> Elderly care		
<b>Scenario:</b> Memory support		
<b>Description:</b> The robot would be applicable in scenarios where the elderly are struggling with remembering important tasks or just general important notes. The robot could be the social companion acting as a reminder.		
<b>Needed information:</b> Context of the environment, personal information, social skills, the importance of matters	<b>Sensory:</b> Speech, text,	<b>Memory:</b> Past interactions, social states, events, up to date events, personal information, information to remember.

**Example scenario:**

Action: Elderly is just sitting at home, forgetting what the next step was

Context: afternoon, call scheduled

Sensory: John, dreaming

Memory: There's a call scheduled, when john is dreaming he often is forgetting his duties and schedule

Output: speech/conversation

Exact output action: "Hey John, you have a call to make, remember! Your son is expecting a call in a minute."

<b>Domain:</b> Elderly care		
<b>Scenario:</b> Rehabilitation assistant		
<b>Description:</b> In this scenario, the robot has the task as a supportive system in the rehabilitation process. A robot could give feedback on exercises or even be the leading agent in the process: so the robot would monitor and construct the rehabilitation. The robot would tell the subject what exercises to do, maybe by showing a screen or some other output mechanism. It would capture the movements of the subject and incorporate that in the feedback and the general processes of the system		
<b>Needed information:</b> Medical background on rehabilitation, social context, social skills applicable to this domain	<b>Sensory:</b> Computer vision, speech, text	<b>Memory:</b> The current state of the process, personal feedback

**Example scenario:**

Action: Elderly is ready for the rehabilitation, said that he is ready

Context: morning, after breakfast,

Sensory: Jeff, speech recalling he is ready

Memory: Last exercise was yesterday, progress is 45%, health is going up

Output: speech/conversation + screen illustration

Exact output action: "Morning Jeff, how are you feeling, you seem to be progressing well!"

//if feeling well; start with the exercise, else update the exercise to feeling & update progress

...

Sensory2: Jeff recalling that he is doing ok

Output2: "That's good to hear!, let's start with this exercise to warm up ... can you see me clearly? ... "

<b>Domain:</b> Supermarket clerk		
<b>Scenario:</b> Entrance greeter		
<b>Description:</b> The robot would be standing at the entrance/exit of the supermarket. It will give normal greetings to customers the robot has not seen before. But whenever a customer the robot has seen before comes in or goes out, it will try to give a more personal greeting message or a shopping tip if there is a discount that might be interesting to that specific customer.		
<b>Needed information:</b> Weather info, current discounts	<b>Sensory:</b> Computer vision, speech	<b>Memory:</b> Identifying features of customers, past encounters

**Example scenario:**

Action: Person enters the shop with facemask and gloves

Context: Corona pandemic going on

Sensory: Tess

Memory: Familiar, person entered without protection last time

Output: speech/conversation

Exact output action: "Hi Tess! I see you're taking the pandemic more seriously already, that's amazing!"

<b>Domain:</b> Supermarket clerk		
<b>Scenario:</b> Item locator		
<b>Description:</b> The robot would be standing in a strategically placed spot in the supermarket. Customers can ask the robot where an item or type of item is located. The robot will give a standard answer to customers it has not encountered before. If the robot recognizes the customer, it will try to give a more personalized answer or even suggest a certain item they have bought before (possibly if that item is on discount).		
<b>Needed information:</b> Store layout, item locations, current discounts	<b>Sensory:</b> Hearing, speech, computer vision	<b>Memory:</b> Identifying features, past encounters

**Example scenario:**

Action: Person shopping for dinner, asks where the wraps are located

Context: Dinnertime, shop mostly empty

Sensory: Tess

Memory: Familiar

Output 1: speech/conversation

Exact output action: "Hi Tess! Shopping for dinner a bit late today?"

Tess responds with: "Yes, work took a bit longer than expected unfortunately."

Output 2: speech/conversation

Exact output action: "That's too bad, it happens. The wraps are in aisle 1, at the end on the bottom right!"

<b>Domain:</b> Supermarket clerk		
<b>Scenario:</b> Recipe assistant		
<b>Description:</b> The robot would be situated at the beginning of the supermarket. This way when customers enter the supermarket, they can immediately go to the robot for assistance with picking out a recipe. They could enter what they would like to eat and what ingredients they already have or want to use. The robot would then give a suggestion that is as personalized as possible.		
<b>Needed information:</b> Recipes, store layout, current discounts	<b>Sensory:</b> Hearing, speech, computer vision	<b>Memory:</b> Identifying features, past encounters

**Example scenario:**

Action: Person asks what they can make with their leftover rice

Context: Before dinnertime

Sensory: Tess

Memory: Familiar, likes spicy curries

Output: speech/conversation

Exact output action: "You like spicy curries right Tess? Well chicken breasts are on discount right now, so you could make that! Chicken is located next to the cheese section, and you can find a spicy curry mix in aisle 2."

<b>Domain:</b> Supermarket clerk		
<b>Scenario:</b> Shopping assistant		
<b>Description:</b> This would be a more mobile robot, or one that could fit in a shopping cart. (Possibly only less abled) Customers can take the robot with them on their shopping, and assist in any way needed. It could help picking out a recipe, locate items, suggest items and grab items from the shelves if the customer has trouble with this. Meanwhile it would also socially interact with the customer as personally as possible.		
<b>Needed information:</b> Recipes, store layout, item locations, current discounts	<b>Sensory:</b> Hearing, speech, computer vision	<b>Memory:</b> Identifying features, past encounters

**Example scenario:**

Action: Person tries to grab eggs that are on the top shelf

Context: Afternoon

Sensory: Janet

Memory: Familiar, has a hard time reaching items on high shelves

Output 1: speech/conversation

Exact output action: "Having trouble reaching for the eggs Janet? Let me grab it for you!"

Output 2: assist shopping

Exact output action: Grab the right brand of eggs from the top shelf and put them in shopping cart

<b>Domain:</b> Supermarket clerk		
<b>Scenario:</b> Stock clerk		
<b>Description:</b> The robot could be put to use in any aisle, stocking products in shelves efficiently. Customers can go up to the robot, and ask it to help them find a certain item. The robot can then relinquish its stocking duties for a while, have a short social interaction with the customer and show them where an item is located.		
<b>Needed information:</b> Store layout, item locations	<b>Sensory:</b> Hearing, speech, computer vision	<b>Memory:</b> Identifying features, past encounters

**Example scenario:**

Action: Person asks where the sugar is located

Context: Workday before dinnertime

Sensory: John

Memory: Familiar, usually shops with son Billy

Output 1: speech/conversation

Exact output action: "Hi John! Billy not with you today?"

John responds with "Not today, he's playing with friends until dinner."

Output 2: speech/conversation

Exact output action: "I see, good for him! The sugar is located in aisle 4, halfway through the aisle on the left bottom shelf."

<b>Domain:</b> Museum guide		
<b>Scenario:</b> Front desk/ help		
<b>Description:</b> This social robot would function on the front/entrance of a museum, where it would be able to help by-passing tourists/ visitors with anything that bothers them: finding the restrooms, having lost a jacket, getting information on the current expositions. This robot would be able to generally help people with their problems, or route them to a place where the solution is likely to be.		
<b>Needed information:</b> Environmental data, up to dates on the museums whereabouts.	<b>Sensory:</b> Visual, computer vision, speech,	<b>Memory:</b> On identical situations, previous solutions, social interactions and recent problems

**Example scenario:**

Action: Someone called jeff lost his kid in the museum

Context: afternoon, busy

Sensory: Jeff, slight panic

Memory: Busy days, kids often go to the kids corner, intercom could be used, people with panic should be calmed.

Output: speech/conversation + intercom + screen

Exact output action: "Hey Jeff, don't worry we'll find him, what's his name?" + Request camera image of kids corner

/\* name = x, first show camera image; If jeff calms down, continue, else request for human assistance

...

Sensory2: Jeff recalling the name of the kid (Bob)

Output2: "Thanks, could you maybe look on my screen and see if bob is in the kids corner?"

/\* if kid not there, use intercom. Else tell directions

...

Sensory3: Jeff confirms seeing bob in the kids corner

Output3: "That's good! You can get there by taking these stairs down and walking straight from there! Just follow the signs!"

<b>Domain:</b> Museum guide		
<b>Scenario:</b> In-museum clerk		
<b>Description:</b> This robot could serve the function of keeping an eye on the museum indoors; while also being able to provide useful and meaningful information on the current expo's.		
<b>Needed information:</b> Environmental data, information on the current expo and its individual properties	<b>Sensory:</b> Computer vision, Speech, maybe an interface would also do	<b>Memory:</b> On relatable facts to the questions asked, past interactions, relating to this person's behavior throughout the museum.

**Example scenario:**

Action: Someone called john is not understanding the art object in front of him, so asks the clerk

Context: afternoon, quiet,

Sensory: John, interested

Memory: quiet => time to explain, art is by artistX, artistX is parody artist, painting is parodyOf otherpainting

Output: speech/conversation

Exact output action: "John, this is an artwork by artistX, he often tries to mock with other artists in making parodies; maybe look at the painting opposite of it; thats what its based on"

<b>Domain:</b> Airport clerk		
<b>Scenario:</b> Information desk		
<b>Description:</b> This robot would be situated behind an information kiosk in the airport. People can come up to the robot as they could to any information kiosk clerk and ask it questions.		
<b>Needed information:</b> Airport layout, general airport information, current flight information	<b>Sensory:</b> Listen to the person, speech, computer vision	<b>Memory:</b> Past interactions, past solutions

**Example scenario:**

Action: Person asks where they can check-in for their flight, one suitcase with them

Context: Holiday

Sensory: Tess

Memory: Familiar, always has two suitcases

Output: speech/conversation

Exact output action: "Hi Tess! Only one suitcase this time? For your flight you can check-in at gate G10, which you will find if you follow the yellow line!"

<b>Domain:</b> Airport clerk		
<b>Scenario:</b> Walkthrough guide		
<b>Description:</b> This robot would be situated at the beginning of the airport. People that have just entered the airport and want some help getting them to where they need to be, can use it as an assistant that will guide them there.		
<b>Needed information:</b> Airport layout, general airport information, current flight information	<b>Sensory:</b> Listen to the person, speech, computer vision	<b>Memory:</b> Past interactions, past solutions

**Example scenario:**

Action: Person looks stressed, asks where she should check-in for her flight

Context: Busy weekend at the start of holiday

Sensory: Tess

Memory: Familiar, usually calms down when someone helps her

Output: speech/conversation

Exact output action: "Hi Tess! You look a bit stressed, let me help you. It looks like you have more than enough time to get to your check-in gate, I'll show you where you have to go!"

<b>Domain:</b> Airport clerk		
<b>Scenario:</b> Store guide		
<b>Description:</b> This robot would be situated in the airport area where all the shops are. People can come up to the robot, tell it what they are looking for, and the robot can guide them to a store that has that product. If that product is not sold in the airport, the robot will give a suggestion of a product based on its past interactions and insight of the person who's asking.		
<b>Needed information:</b> Airport layout, store locations, product line of all stores	<b>Sensory:</b> Listen to the person, speech, computer vision	<b>Memory:</b> Past interactions, past solutions

**Example scenario:**

Action: Person comes back from long business trip, asks where the whiskey store is

Context: Evening of work day

Sensory: John

Memory: Familiar, wife likes whiskey and wine

Output: speech/conversation

Exact output action: "Hi John! Unfortunately this airport closed its whiskey store. But may I suggest the wine store that replaced it, your wife likes that too right? The wine store is to your left here."

<b>Domain:</b> Education		
<b>Scenario:</b> Peer (child)		
<b>Description:</b> The robot would act as a peer of a learning child. In this sense it would be empathetic and follow the track of the child peer-wise. It would learn together with the child and would recall on previous interactions		
<b>Needed information:</b> Educational matters of subject, learning progress	<b>Sensory:</b> Textbased, speech probably; text	<b>Memory:</b> Previous interactions, progress, struggles

**Example scenario:**

Action: Kid would be coming home from school and is starting with the homework.

Context: afternoon,

Sensory: Bob, homework on Maths

Memory: Bob has difficulties with Maths, Last progress was chapter 4.6 70%, Bob always forgets to grab some tea or lemonade before he starts.

Output: speech/conversation

Exact output action: "Hey Bob, ready for Maths today; at what question are we? Don't you forget your lemonade!"

<b>Domain:</b> Education		
<b>Scenario:</b> Teacher / teaching assistant		
<b>Description:</b> This would be a more high level task, where the robot actually provides the student some matters. Logically this could be applied in the first learning steps of youngsters. But also improve social behavior.		
<b>Needed information:</b> Subject information, psychological knowledge	<b>Sensory:</b> Speech, computer vision	<b>Memory:</b> Current state of learning, emotional state of subject

**Example scenario:**

Action: Class of Kids, learning the alphabet

Context: midday, much energy

Sensory: class full of engaged youngsters, not grasping the letter 'x'

Memory: 'x' is a hard letter to understand, middays it is important to not be too complex, break is coming.

Output: speech/conversation screen

Exact output action: "Okay Kids, lets just write 'x' like this a hundred times! And try ro pronounce it like eggs everytime we write it! After that there is a break!"

Domain: Household		
Scenario: home assistant - reminder		
<b>Description:</b> This home assistant would remind you of your daily tasks and inform you over things relevant to you; some 'google home' like devices. It is present as a social go-to and just your personal assistant.		
<b>Needed information:</b> Environmental data, social states,	<b>Sensory:</b> Speech, maybe vision	<b>Memory:</b> Social states (previous), the tasks (though probably linked to some calendar), things you like and current event links

**Example scenario:**

Action: Sally comes home from the fitness and was supposed to do some gardening in the afternoon

Context: afternoon, rainy

Sensory: Sally, tired

Memory: Fitness is tiring, tired people don't like tasks, gardening is outside, being outside is not nice when it rains.

Output: speech/conversation

Exact output action: "Afternoon Sally, you were supposed to do some gardening this afternoon, but maybe postpone that to a more sunny day? Then you can also take some rest?"

/\* if postponing, find a new spot in the agenda and propose that, else wish good luck.

...

Sensory2: Sally agrees on postponing, but asks for a simpler task

Output2: "That's good, shall I postpone it to tomorrow afternoon?"

Output3: "Maybe you can then do the laundry now instead of tomorrow?"

Domain: Household		
Scenario: Personal trainer		
<b>Description:</b> The Robot would act as a personal trainer, where it would help you keep in shape and keep track of your training		
<b>Needed information:</b> What is fit/shape, background on fitness, Health knowledge, maybe music?	<b>Sensory:</b> Visual, maybe sound	<b>Memory:</b> Previous workouts, errors, <u>hiccups</u> and persona's

**Example scenario:**

Action: Jim tunes in for his afternoon session

Context: afternoon, rainy

Sensory: Jim, not so energetic

Memory: Fitness is tiring, doing sports makes you energetic, sports require energy, Jim is currently doing a 'core' buildup, core requires good nutrition.

Output: speech/conversation, screen

Exact output action: "Have you had a good meal Jim? You will need some energy today"

/\* if yes, lets start, else, maybe tune the workout a bit and tip him on better eating next time.

...

Sensory2: Jim confirms having had a good meal, but just a long day

Output2: "Okay, then we can start, are you ready?"

<b>Domain:</b> Household		
<b>Scenario:</b> Housekeeping		
<b>Description:</b> This robot would have as task to keep the house clean and tidy. This robot's tasks would entail actions as cleaning, doing the dishes, laundry, vacuum cleaning. This robot should be able to understand what the 'owner' asks from 'him/her'. The robot can be asked to clean the upstairs floor and can be asked to clean up a room.		
<b>Needed information:</b> Topographic information on households, classes existing in households	<b>Sensory:</b> Visual, speech, topographical information; connection to internet for 'owners' whereabouts	<b>Memory:</b> To-go state, locations of objects, relatable tasks

**Example scenario:**

Action: Someone spilled coffee  
Context: house is quite clean, there's a small social coffee drinking thing going on  
Sensory: Jeff, Sally, + 3 friends  
Memory: Cleaning makes noise, noise is not nice in conversation, coffee is not supposed to be on the floor, it should be cleaned  
Output: speech/conversation  
Exact output action: "Hey Jeff, shall I clean that now, it will make some noise?"  
/\* if yes; clean.exe, else store that it should be cleaned  
...  
Sensory2: jeff recalls he will do it himself quickly  
Memory: cleaning requires a towel  
Output2: "Here, I'll bring you a towel"  
/\* remove that it should be cleaned later

<b>Domain:</b> City centre		
<b>Scenario:</b> Hotspot information		
<b>Description:</b> The robot would be situated at a specific tourist hotspot. It would give tourists who come by more personal answers to questions that might not be on general signs.		
<b>Needed information:</b> General city information, city history information, information about the touristic hotspot	<b>Sensory:</b> Hearing, speech, computer vision	<b>Memory:</b> Past encounters, past questions and answers to those questions

**Example scenario:**

Action: Person asks the most interesting fact that most people don't know about the hotspot  
Context: Holiday  
Sensory: Tess  
Memory: Familiar, is interested in history  
Output: speech/conversation  
Exact output action: "Hi Tess! Something most people don't know about X, is that Y happened to it in YEAR."

<b>Domain:</b> City centre		
<b>Scenario:</b> Tourism tips		
<b>Description:</b> The robot would be situated at the place where tourists usually enter the city the first time, like the rain station. This way tourists who want help to pick out where in the city they should go, could go up to the robot.		
<b>Needed information:</b> General city information, knowledge of touristic hotspots, knowledge of non-hotspot locations	<b>Sensory:</b> Hearing, speech, computer vision	<b>Memory:</b> Past encounters, past itinerary plans given and reviews of those plans

**Example scenario:**

Action: Persons ask what are some fun things to do in the city

Context: Amsterdam group holiday, good weather for a week

Sensory: Group of guys in their teens

Memory: Not familiar, but teens are usually interested in Red Light district and more

Output: speech/conversation

Exact output action: "Hi guys! I think you will probably enjoy seeing the Red Light district, the Vondelpark and the clubs in the city center at night. Here is a schedule you could follow!"

## References

- [1] J. P. Aggleton, D. McMackin, K. Carpenter, J. Hornak, N. Kapur, S. Halpin, C. M. Wiles, H. Kamel, P. Brennan, S. Carton, and D. Gaffan. Differential cognitive effects of colloid cysts in the third ventricle that spare or compromise the fornix. *Brain*, 123(4):800–815, 04 2000.
- [2] M.I. Ahmad, O. Mubin, S. Shahid, and J. Orlando. Robot’s adaptive emotional feedback sustains children’s social engagement and promotes their vocabulary learning: a long-term child–robot interaction study. *Adaptive Behavior*, 27(4):243–266, 2019.
- [3] C Anderson and D. Keltner. The role of empathy in the formation and maintenance of social bonds. *Behavioral and Brain Sciences*, 25:21 – 22, 02 2002.
- [4] Nomura T. Kanda T. Suzuki T. Bartneck, C. and K. Kennsuke. A cross-cultural study on attitudes towards robots. 2005.
- [5] A. Bisler. An associative memory for autonomous agents. 10 2004.
- [6] T. Curran. Brain potentials of recollection and familiarity. *Memory & Cognition*, 28(6):923–938, Nov 2000.
- [7] E. Düzel, A.P. Yonelinas, G.R. Mangun, H.J. Heinze, and E. Tulving. Event-related brain potential correlates of two states of conscious awareness in memory. *Proceedings of the National Academy of Sciences of the United States of America*, 94(11):5973–5978, May 1997. 9159185[pmid].
- [8] S. Franklin. Cognitive robots: perceptual associative memory and learning. In *ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication*, pages 427–433, Nashville, TN, USA, 2005.
- [9] S. Franklin, T. Madl, S. Strain, U. Faghihi, D. Dong, S. Kugele, J. Snaider, P. Agrawal, and S. Chen. A lida cognitive model tutorial, April 2016.
- [10] M. Glanzer, K. Kim, A. Hilford, and J.K. Adams. Slope of the receiver-operating characteristic in recognition memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 25(2):500–513, 1999.
- [11] J. Goetz, S. Kiesler, and A. Powers. Matching robot appearance and behavior to tasks to improve human-robot cooperation. In *The 12th IEEE International Workshop on Robot and Human Interactive Communication, 2003. Proceedings. ROMAN 2003.*, pages 55–60, 2003.
- [12] D.M. Green and J.A. Swets. *Signal detection theory and psychophysics*. Signal detection theory and psychophysics. John Wiley, Oxford, England, 1966.

- [13] S.D. Gronlund, M.B. Edwards, and D.D. Ohrt. Comparison of the retrieval of item versus spatial position information. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 23(5):1261–1274, 1997.
- [14] Cognitive Computing Research Group. Robot density rises globally. <http://ccrg.cs.memphis.edu/index.html>. Accessed: 2020-03-14.
- [15] H. Hastie, M.Y. Lim, S.C. Janarthanam, A. Deshmukh, R. Aylett, M.E. Foster, and L. Hall. I remember you! interaction with memory for an empathic virtual robotic tutor. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016)*, pages 931–939, United States, May 2016. Association for Computing Machinery. 15th International Conference on Autonomous Agents and Multiagent Systems 2016, AAMAS 2016; Conference date: 09-05-2016 Through 13-05-2016.
- [16] D.L. Hintzman and D.A. Caulton. Recognition memory and modality judgments: A comparison of retrieval dynamics. *Journal of Memory and Language*, 37(1):1–23, 1997.
- [17] D.L. Hintzman, D.A. Caulton, and D.J. Levitin. Retrieval dynamics in recognition and list discrimination: Further evidence of separate processes of familiarity and recall. *Memory & Cognition*, 26(3):449–462, May 1998.
- [18] Douglas R. Hofstadter and Melanie Mitchell. *The Copycat project: A model of mental fluidity and analogy-making.*, pages 31–112. Advances in connectionist and neural computation theory, Vol. 2. Ablex Publishing, Westport, CT, US, 1994.
- [19] F.A. Huppert and M. Piercy. Recognition memory in amnesic patients: Effect of temporal context and familiarity of material. *Cortex*, 12(1):3 – 20, 1976.
- [20] L.B. Jahromi. *Recognition Memory*, pages 2531–2531. Springer New York, New York, NY, 2013.
- [21] Robert Kelley and John T. Wixted. On the nature of associative information in recognition memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 27(3):701–722, 2001.
- [22] MIT Media Lab. Conceptnet. <https://conceptnet.io/>. Accessed: 2020-05-9.
- [23] Nickel A.E. Mahoney, E.J. and D.E. Hannula. *Recognition*, pages 37–43. Elsevier, 2015.
- [24] International Federation of Robotics. Robot density rises globally. <https://ifr.org/ifr-press-releases/news/robot-density-rises-globally>. Accessed: 2020-02-24.

- [25] Colleen Parks and Andrew Yonelinas. Moving beyond pure signal-detection models: Comment on wixted (2007) - postscript. *Psychological review*, 114:188–202; discussion 203, 02 2007.
- [26] R. Pröpper, F. Putze, and T. Schultz. *JAM: Java-based Associative Memory*, pages 143–155. 01 2011.
- [27] J.G. Raaijmakers and R.M. Shiffrin. Search of associative memory. *Psychological Review*, 88(2):93–134, 1981.
- [28] J. Retto. Sophia, first citizen robot of the world. 11 2017.
- [29] J. Snider, R. McCall, and S. Franklin. The lida framework as a general tool for agi. pages 133–142, 08 2011.
- [30] W.A. Suzuki. Associative learning and the hippocampus. <https://www.apa.org/science/about/psa/2005/02/suzuki/>. Accessed: 2020-04-10.
- [31] A.P. Yonelinas. Receiver-operating characteristics in recognition memory: Evidence for a dual-process model. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 20(6):1341–1354, 1994.
- [32] A.P. Yonelinas. The nature of recollection and familiarity: A review of 30 years of research. *Journal of Memory and Language*, 46(3):441–517, 2002.