UNIVERSITAT DE BARCELONA

Bachelor's Degree Final Project Thesis

Bachelor's Degree in Computer Science and Software Engineering

Faculty of Mathematics and Informatics
University of Barcelona

Machine Learning and Sentient Embodied Conversational Agents: Design and Implementation of a Virtual Tutor in the Context of Energy Efficiency and Sustainability

---

Dolça Tellols Asensi

Advisor: Dr. Pablo Almajano Francoy
Departament of Mathematics and Informatics
Barcelona, 26th June of 2018

Acknowledgements

This Final Degree Thesis has been possible thanks to the support and motivation that I have received from so many people.

Thanks to all the professors that have accompanied me during all the different academic stages and provided me with so many knowledge. Specifically, thanks to the professors of the Bachelor's Degree in Computer Science and Software Engineering and the Bachelor's Degree in Mathematics of the University of Barcelona.

Special thanks to Maite López-Sánchez and Inmaculada Rodríguez Santiago, who have been my supervisors of the departments collaboration grant and helped me a lot in the elaboration this Final Degree Thesis, and Pablo Almajano Francoy, my thesis advisor. Thanks to the three of you for everything you have taught me, for giving me research opportunities and for believe in me.

Thanks also to the professors of Waseda University (Japan), where in the framework of a General Agreement for exchange of the UB, I could enjoy an extraordinary experience at personal and academic levels. Thanks to Anna Puig Puig for helping me making the procedures that made it possible.

I also would like to thank the Rubí Brilla initiative for supporting this project, as well as the Montessori and 25 de Setembre schools from Rubí and the Escola del Mar from Barcelona. Thanks also to all the children and families who voluntarily participated making possible the evaluation of this project.

Finally, this last paragraph is to thank my friends because, without mattering if our paths met one or ten years ago or if we are near or a thousand kilometres away, with you I can always relax and recover energy in the most complicated moments.

I also want to thank my parents and my brother Oriol for their unconditional support. You have helped me being who I am and achieving my goals.

Agraïments

Aquest Treball de Final de Grau ha estat possible gràcies al suport i la motivació que he rebut de moltes persones.

Dono les gràcies a tot el professorat que m'ha acompanyat en el meu aprenentatge i aportat tants coneixements al llarg de les diferents etapes educatives, sobretot al professorat del grau d'Enginyeria Informàtica i del grau de Matemàtiques de la Universitat de Barcelona.

Vull agrair especialment a Maite López-Sánchez i a Inmaculada Rodríguez Santiago, que han estat les meves supervisores de la beca de col·laboració en departaments i que m'han ajudat molt en l'elaboració d'aquest Treball de Final de Grau, i a Pablo Almajano Francoy, el meu director del projecte. Moltes gràcies als tres per tot el que m'heu ensenyat, per donar-me oportunitats d'investigació i per creure en mi.

Gràcies també al professorat de la Universitat de Waseda (Japó), on en el marc d'un Conveni General d'intercanvi de la UB vaig poder gaudir d'una experiència personal i acadèmica extraordinària. Gràcies a Anna Puig Puig per ajudar-me a fer les gestions que la van fer possible.

Seguidament, vull manifestar el meu agraïment a la iniciativa Rubí Brilla per donar suport a aquest projecte, i també a les escoles Montessori i 25 de Setembre de Rubí i a l'Escola del Mar de Barcelona. Gracies també a tots els nens, nenes i famílies que han participat voluntàriament fent possible l'avaluació del projecte.

Aquestes últimes línies són per donar les gràcies als meus amics i amigues perquè, sense importar si els nostres camins es van creuar fa un o deu anys o si estem a prop o a milers de kilòmetres, amb vosaltres sempre puc relaxar-me i carregar energia en els moments més complicats.

També vull agrair als meus pares i al meu germà Oriol tot el seu suport incondicional. M'heu ajudat a ser qui sóc i a aconseguir els meus objectius.

Agradecimientos

Este Trabajo de Fin de Grado ha sido posible gracias al apoyo y la motivación que he recibido de muchas personas.

Doy las gracias a todo el profesorado que me ha acompañado durante mi aprendizaje y que me ha aportado tantos conocimientos a lo largo de las diferentes etapas educativas, sobre todo al profesorado del grado de Ingeniería Informática y del grado de Matemáticas de la Universidad de Barcelona.

Quiero agradecer especialmente a Maite López-Sánchez y a Inmaculada Rodríguez Santiago, que han sido mis supervisoras de la beca de colaboración con departamentos y que me han ayudado mucho en la elaboración de este Trabajo de Fin de Grado, y a Pablo Almajano Francoy, mi director del proyecto. Muchas gracias a los tres por todo lo que me habéis enseñado, por darme oportunidades de investigación y por creer en mí.

Gracias también al profesorado de la Universidad de Waseda (Japón), donde en el marco de un Convenio General de intercambio de la UB pude disfrutar de una experiencia personal y académica extraordinaria. Gracias a Anna Puig Puig por ayudarme a hacer las gestiones que la hicieron posible.

Seguidamente, quiero manifestar mi agradecimiento a la iniciativa Rubí Brilla por dar apoyo a este proyecto, y también a las escuelas Montessori y 25 de Setembre de Rubí y a l'Escola del Mar de Barcelona. Gracias también a todos los niños, niñas y familias que han participado voluntariamente haciendo posible la evaluación del proyecto.

Estas últimas líneas son para darles las gracias a mis amigos y amigas porque, sin importar si nuestros caminos se cruzaron hace uno o diez años o si estamos cerca o a miles de kilómetros, con vosotros siempre puedo relajarme y recargar energías en los momentos más complicados.

También quiero agradecer a mis padres y a mi hermano Oriol todo su apoyo incondicional. Me habéis ayudado a ser quien soy y a conseguir mis objetivos.

Abstract

Current increase of chatbots' popularity in the Human Computer Interaction field has lead to a proliferation of platforms that facilitate their design and seamless integration. However, chatbots are non-embodied conversational agents designed for communicating with the user using simple and quick goal-based interactions that fall short when it comes to engage users in longer, diverse and complex conversations.

This work presents *Sentient Embodied Conversational Agents* (SECAs), which are embodied virtual characters. SECAs can engage users in complex structured conversations and also incorporate some human-like sentient qualities like being able to perceive user's feelings and response to them.

This project includes the formalization of a SECA architecture and the implementation of a software library that facilitates the inclusion of SECAs in applications requiring proactive and sensitive agent behaviours. SECA library has a modular design to facilitate its modification. It includes a controller and different modules to model different features SECAs have (conversational capability, knowledge, memory, personality, needs and empathy).

Since educational applications constitute an example with such requirements, this work designs, implements and evaluates a virtual tutor (the Earth), which uses the presented architecture and software library, and embeds it in a gamified cultural probes application for children. This SECA's purpose is enhancing the user experience and making it more educative.

This project performs two complete iterations of the User Centered Design methodology process. The first one embeds in the application a first prototype of the agent using keyword-pattern matching techniques to analyse user input. On the other hand, the second one uses user feedback and data gathered to create an enhanced version of the agent with Machine Learning. The final version of the software library includes a Natural Language Processing Module integrating keyword-pattern matching techniques and ML with the purpose of improving SECAs understandability.

Evaluation results of the application with the Earth SECA embedded show that children are satisfied both in terms of their perception of learning and overall experience. Additionally, there are significant differences in quantity of user data gathered by the new CP application (with SECA) and its previous version (without SECA). Though there is still room for improvement, NLP Module also improved the user input analysis results with respect of the initial prototype of the application with the Earth SECA using only keyword-pattern matching techniques.

Resum

El recent augment de popularitat dels "chatbots" en el camp de la Interacció Home-Màquina ha portat a la proliferació de plataformes que faciliten el seu disseny i integració. De tota manera, els "chatbots" són agents conversacionals no-personificats, dissenyats per a comunicar-se amb l'usuari mitjançant converses senzilles, ràpides i amb objectius concrets. Per tant, es queden curts quan es volen establir converses més llargues, diverses i complexes amb els usuaris.

Aquest treball presenta els *Sentient Embodied Conversational Agents* (SECA, i.e. Agents Conversacionals Personificats Sensitius), que són personatges virtuals personificats. Els SECA són capaços d'implicar els usuaris en converses complexes i també d'incorporar algunes qualitats sensitives semblants a les dels humans com poder percebre com se senten els usuaris.

Aquest projecte inclou la formalització de l'arquitectura dels SECA i la implementació d'una llibreria software que facilita la inclusió dels SECA en aplicacions que puguin requerir comportaments proactius i sensitius per part de l'agent. La llibreria SECA té un disseny modular que facilita la seva modificació. Inclou un controlador i diversos mòduls per modelar diferents característiques que tenen els SECA (capacitat conversacional, coneixement, memòria, personalitat, necessitats i empatia).

Com les aplicacions educatives són un exemple d'aplicació amb els requeriments anteriors, aquest treball dissenya, implementa i avalua un tutor virtual (la Terra), que utilitza l'arquitectura i la llibreria software presentades, i l'incorpora en una aplicació Cultural Probes (CP) gamificada per nens i nenes amb l'objectiu de millorar l'experiència de l'usuari fent-la més educativa.

Aquest projecte realitza dues iteracions completes del procés seguit per la metodologia "User Centered Centered Design" (UCD, i.e. Disseny Centrat en l'Usuari). La primera, incorpora en l'aplicació un primer prototip de l'agent que utilitza tècniques de coincidència de patrons de paraules per analitzar el text introduït pels usuaris. Per una altra banda, la segona aprofita la retroacció dels usuaris i les dades recollides per crear una versió millorada de l'agent amb "Machine Learning" (ML, Aprenentatge Automàtic). La versió final de la llibreria software inclou un Mòdul de Processament del Llenguatge Natural (NLP), integrant tècniques de coincidència de patrons de paraules amb ML, amb l'objectiu de millorar la capacitat dels SECA d'entendre els missatges dels usuaris.

Els resultats de l'avaluació de l'aplicació amb el SECA Terra incorporat mostren com els participants estan satisfets tant pel que consideren que han après com per la seva experiència general. A més, es poden apreciar diferències significatives en la quantitat de dades dels usuaris recollides per la nova aplicació CP (amb SECA) respecte a l'anterior versió (sense SECA). Encara que hi ha marge de millora, el Mòdul de NLP ha millorat els resultats de l'anàlisi del text introduït pels usuaris respecte a l'aplicació amb el SECA Terra que només utilitzava tècniques de coincidència de patrons de paraules.

Resumen

El reciente aumento de popularidad de los "chatbots" en el campo de la Interacción Hombre-Máquina ha llevado a la proliferación de plataformas que faciliten su diseño e integración. De todos modos, los "chatbots" son agentes conversacionales no-personificados, diseñados para comunicarse con el usuario mediante conversaciones simples, rápidas y con objetivos concretos. Por tanto, se quedan cortos cuando se quieren establecer conversaciones más largas, diversas y complejas con los usuarios.

Este trabajo presenta los *Sentient Embodied Conversational Agents* (SECAs, i.e. Agentes Conversacionales Personificados Sensitivos), que son personajes virtuales personificados. Los SECA son capaces de implicar a los usuarios en conversaciones complejas y también de incorporar algunas cualidades sensitivas semejantes a las de los humanos como poder percibir como se sienten los usuarios.

Este proyecto incluye la formalización de la arquitectura de los SECA y la implementación de una librería software que facilita la inclusión de los SECA en aplicaciones que puedan requerir comportamientos proactivos y sensitivos por parte del agente. La librería SECA tiene un diseño modular que facilita su modificación. Incluye un controlador y diversos módulos para modelar diferentes características que tienen los SECA (capacidad conversacional, conocimiento, memoria, personalidad, necesidades y empatía).

Como las aplicaciones educativas son un ejemplo con los requisitos anteriores, este trabajo diseña, implementa y evalúa un tutor virtual (la Tierra), que utiliza la arquitectura y la librería software presentadas, y lo incorpora en una aplicación Cultural Probes (CP) gamificada para niños y niñas con el objetivo de mejorar la experiencia del usuario haciéndola más educativa.

Este proyecto realiza dos iteraciones completas del proceso seguido por la metodología "User Centered Design" (UCD, i.e. Diseño Centrado en el Usuario). La primera, incorpora en la aplicación un primer prototipo del agente que utiliza técnicas de coincidencia de patrones de palabras para analizar el texto introducido por los usuarios. Por otra parte, la segunda aprovecha la retroacción del usuario y los datos recogidos para crear una versión mejorada del agente con "Machine Learning" (Aprendizaje Automático). La versión final de la librería software incluye un Módulo de Procesamiento del Lenguaje Natural (NLP), integrando técnicas de coincidencia de patrones de palabras con ML, con el objetivo de mejorar la capacidad de los SECAs de entender los mensajes de los usuarios.

Los resultados de la evaluación de la aplicación con el SECA Tierra incorporado muestran cómo los participantes están satisfechos, tanto porque consideran que han aprendido como por su experiencia general. Además, se pueden apreciar diferencias significativas en la cantidad de datos de los usuarios recogidos por la nueva aplicación CP (con SECA) respecto a la anterior versión (sin SECA). Aunque hay margen de mejora, el Módulo NLP ha mejorado los resultados del análisis del texto introducido por los usuarios con respecto a la aplicación con el SECA Tierra que sólo utilizaba técnicas de coincidencia de patrones de palabras.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context and Motivation

Current increase of chatbots' popularity has lead to a proliferation of platforms that facilitate their design and seamless integration in the social web. However, chatbots are (non-embodied) conversational agents designed for communicating with the user using simple and quick goal-based interactions. Although useful in some cases, they fall short when it comes to engage users in longer, diverse and complex conversations. In these cases, the focus should be agents' embodiment and believability. An Embodied Conversational Agent (ECA) [24] is a virtual character able to establish verbal and non-verbal communication with the user. ECAs are useful for training, guiding, and giving support to users in a more comfortable way through the use of natural language. Nevertheless, ECAs are usually created ad hoc for a specific purpose, with no chance for reuse nor evolution of their functional and structural components.

This thesis continues in the line of a previous research that focused on the design of a gamified digital Cultural Probes for children (between 10 and 12 years old) [59] [54]. "Cultural Probes" (CP) are an analysis methodology that permits empathizing more with users to gather more interesting and useful data to design applications [32]. Consequently, the aim of the research was gathering information about families' behaviour related to energy consumption and improve their energy consciousness. The Cultural Probes application, named "Energy Madness", was implemented in the context of another final degree thesis [55] and it was tested with children of some schools. After finishing the previously designed CP experience, children manifested that they would like the application to have more content and they also would like to have access to the explanation of different energy related concepts that they did not know nor understand. Taking these into consideration, the idea of this final degree thesis emerged by contemplating the possibility of implementing an ECA that could be included in the CP application as a Virtual Tutor in the field of energy efficiency and sustainability.

A previous paper [20] presented the structure of a task-based Embodied Conversational Agent (ECA) –i.e., proactive agent able to communicate with users through facial expressions, natural language, gestures and movement [24]– which was also capable of showing emotions and motivation. This project advances the state of the art by designing and implementing the structure of an ECA, named as SECA (Sentient Embodied Conversational Agent), capable of performing complex structured conversations, showing emotions, having needs and being empathic. The design proposes a reusable and customizable modular architecture.

This project also uses SECA architecture to design and implement a virtual tutor for the already developed gamified CP application for Android devices designed with Unity, having the purpose of testing its performance and evaluating how users perceived its sentient and conversational capabilities.

To achieve a satisfactory interaction between the user and the SECA, this work puts a lot of effort in the analysis of users input so that they have the feeling that the agent understands them. Figure 1.1 shows the different versions developed of the application. The initial idea consisted on using machine learning to develop the SECA but, since data is necessary to train the algorithms, during this project a basic SECA using AIML and simple keyword-pattern matching functions has been deployed first to gather data from real users. And then, the data has been used to train different machine learning algorithms to improve the conversational capability of the agent. The enhanced SECA has been tested again with real users to evaluate it and compare its performance with the previous one.

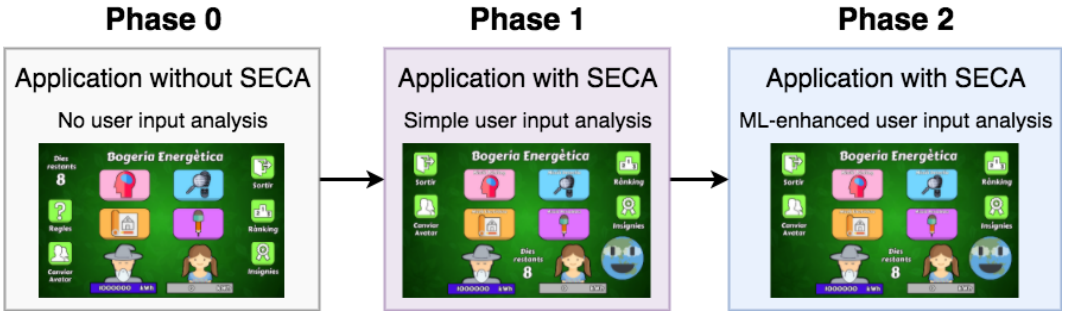

Figure 1.1: CP application versions. Phase 0: application without SECA and without user input analysis. Phase 1: application with SECA and basic NLP with AIML and keyword-pattern matching functions. Phase 2: application with SECA and enhanced NLP with AIML, keyword-pattern matching and Machine Learning functions.

This project also results from the work performed in the context of a department collaboration grant under the supervision of Dr. Inmaculada Rodríguez Santiago and Dr. Maite López-Sánchez from the WAI research group of the University of Barcelona. Moreover, this work contributes to the national research project "Collectiveware: technologies to enhance human collectives in the Smart Grid" (code: TIN2015-66863-C2-1-R) led by Dr. Juan Antonio Rodríguez-Aguilar from the Artificial Intelligence Research Institute (IIIA-CSIC).

1.2 Goals

Considering the previously mentioned context, this project has the following goals:

- Provide a general architecture and a software library to implement Sentient Embodied Conversational Agents (SECA).
- Design and implement a virtual tutor for a gamified Cultural Probes application –in the context of energy efficiency and sustainability– using the defined SECA architecture. This objective can be accomplished in two phases:
  - A first phase where the application is enhanced with a SECA that uses basic NLP to conduct conversations. The resulting prototype allows to gather conversations that can be feed to Marchine Learning algorithms in the subsequent phase.
  - A second phase that uses gathered data to enrich the SECA's NLP by means of Machine Learning conversational models.
- Enhance the previous version of the gamified CP application "Energy Madness" –through the inclusion of the virtual tutor and other improvements– to make the experience more educative and gather more valuable data.
- Evaluate the gamified CP application with the integrated virtual tutor to:
  - Assess the Conversational User eXperience (CUX) while evaluating the efficacy of the proposed SECA architecture.
  - Determine whether the Earth SECA presence impacted on the quantity of data gathered in the Cultural Probes.

In a collateral manner:

- This project wants to raise awareness on energy efficiency and sustainability in all the users (children) who participate in the evaluation of the application and their families.

Moreover, the previous goals will also allow me to achieve other secondary goals at an academic and personal level:

- Apply knowledge acquired throughout the Computer Science and Software Engineering and the Mathematics Bachelor's Degrees.
- Do research in a number of research areas such as Natural Language Processing, Machine Learning, or Human Computer Interaction, increase my knowledge and improve my capability of choosing the most appropriate technique.
- Learn about and use different Operating Systems, multi-platform software tools and libraries.
- Improve my capability to do team work in a research context, my organization skills and my written and spoken expression.

1.3 Related Work

Nowadays, many non-goal oriented chatbots, mainly designed for entertainment, are being deployed. For example, Rinna, a Microsoft conversational AI which speaks like a Japanese secondary school student [47]. On the other hand, some companies are focusing more on providing tools for implementing goal oriented chatbots, such as those facilitated by Google's DialogFlow [34], Facebook's wit.ai [19], IBM's Watson Assistant [39], and Microsoft's LUIS [46], which are suitable for creating agents that engage the user in single-turn or simple multi-turn conversations. For example, Foodie [21] is a kitchen assistant implemented with Watson technology, that recommends recipes based on user's dietary goals.

However, these have limitations when embedded in applications requiring more complex conversations and human-like properties –in favour of agent's believability and user's engagement– such as embodiment, personality, needs, or empathy. To overcome some of those limitations, Embodied Conversational Agents (ECAs) [24], i.e. proactive interface agents that can communicate with humans using facial expressions, natural language, gestures and movement, are also being implemented. Some examples include ALMA [33], which adopts the Five Factor Model (FFM) of personality [44] and the Ortony, Clore and Collins (OCC) model [49] for defining emotions; or Max [22], an ECA that implements the concept of boredom to represent the absence of stimuli from the user, which can be considered as a way of modelling the agent's interaction need. Some other examples include Greta [51], an expressive 3D female agent or Steve [53], an ECA used to assist training.

Regarding virtual tutors, some have already been designed to provide knowledge for specific fields. For example, Duolingo Bots [4], which have different personalities and are being used in the language learning field, or AutoTutor [35], which provides challenging problems for the students formulated as questions whose answers also receive some feedback. But the use of 2D/3D virtual tutors is still not common and research keeps being done like the one performed in the Emote [56] project, which developed robotic tutors for specific tasks and proved the importance of being empathic with the user during the interaction.

One of the most important components of conversational agents is their natural language conversational capability to interact with human users. The Artificial Intelligence Mark-up Language (AIML) is a well-known mechanism used to implement chatbots that has been used for example with A.L.I.C.E. [60], which establishes real conversations with users. However, AIML has limitations and other natural language processing techniques are being implemented. An example is the open-domain conversational system that wants to take advantage of Internet data through different NLP modules proposed by R. Higashinaka et. al.[37].

Taking the aforementioned related work into consideration, this project innovates by integrating all the ideas to present a generic architecture and software library conceived to design and integrate Sentient Embodied Conversational Agents capable of performing complex structured conversations in different kinds of applications.

Moreover, this project contributes to the Human Computer Interaction research area by using the proposed architecture to design and implement a virtual tutor for a digital gamified CP. The agent has the purpose of improving the learning experience associated with the CP and enhancing user experience.

Finally, a contribution is related to NLP through the training, integration and testing of different Machine Learning techniques to correctly guide the virtual tutor through the designed structured conversations while improving its conversational capabilities in the field of energy efficiency and sustainability. Though this thesis presents a concrete implementation, it also explains a general approach so that the used NLP flow and techniques can be replicated for other instantiated agents using the proposed architecture.

1.4 Thesis Structure

This thesis is structured according to the following chapters:

- Chapter 1 (Introduction). Current chapter, introduces the context, motivation, main goals, and related work of the project.
- Chapter 2 (Methodology). Explains the followed User Centered Design methodology and the detailed planning followed during the development process.
- Chapter 3 (Sentient Embodied Conversational Agents). Defines SECAS, presents a general architecture, and formalizes an implemented software library.
- Chapter 4 (Natural Language Processing Module). Presents the definition of this SECAs' module and some implementation ideas using the Catalan language as an example.
- Chapter 5 (Machine Learning and NLP). Introduces different ML algorithms and explains a general procedure to train and evaluate classification models.
- Chapter 6 (Virtual Tutor for a Gamified Cultural Probes Application). Explains in detail the analysis, design, implementation and testing of the Earth SECA virtual tutor.
- Chapter 7 (Evaluation). Detailed analysis of the results obtained during the different tests of the application.
- Chapter 8 (Conclusions and Future Work). Drawn conclussions and future work.

Moreover, the following appendices complement the thesis:

- Appendix A (Developer Manual).
- Appendix B (User Manual).
- Appendix C (Evaluation Material).
- Appendix D (FSM Diagrams within the Conversational Module).

Chapter 2

Methodology

2.1 User Centered Design

This project follows User Centered Design (UCD) methodology, which is a design process that focuses on user needs and requirements [29]. UCD is an iterative process with three different phases in each iteration. Those are:

- Analysis (A), to know users' context and their requirements.
- Prototyping (P), to perform design and implementation.
- Evaluation (E), to test with real users.

Another final degree project finished a first iteration (phases A1, A2 and A3) focusing on its first design and implementation. The results of the evaluation of that iteration motivated the following steps. Figure 2.1 shows how this project addresses two new iterations:

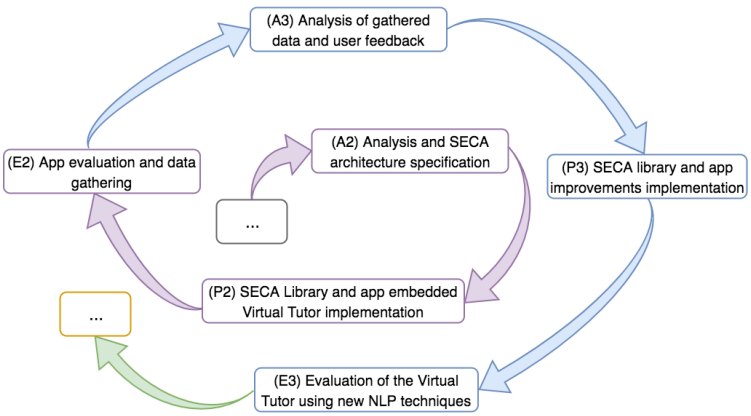

Figure 2.1: UCD Iterations.

- Iteration one (phases A2, P2 and E2) focuses on the implementation of a new application prototype that includes a SECA in the role of a virtual tutor to motivate users, raise awareness about energy efficiency and give them knowledge about concepts and topics related to the field. The NLP capability of this agent uses basic keyword-pattern matching functions and AIML and it will be evaluated with real users to gather data that will be used in the next iteration and to get some initial evaluation results.
- The following iteration (phases A3, P3 and E3), consists in the improvement of the application and the virtual tutor capabilities by analysing the previously obtained results. NLP techniques will be enhanced through the integration of some Machine Learning algorithms trained with the data gathered in the previous iteration. The final prototype will be evaluated again with real users.

The specified iterations also embed the design of the general SECA architecture (phase A2), the implementation of a general SECA software library (phase P2), its evaluation thanks to the results obtained (phase E2) and its enhancement after analysing the obtained results (phase A3) and improving its code (phase P3) to then evaluate it again (phase E3).

2.2 Planning

Figure 2.2 shows the Gantt diagram with the different phases of this project development.

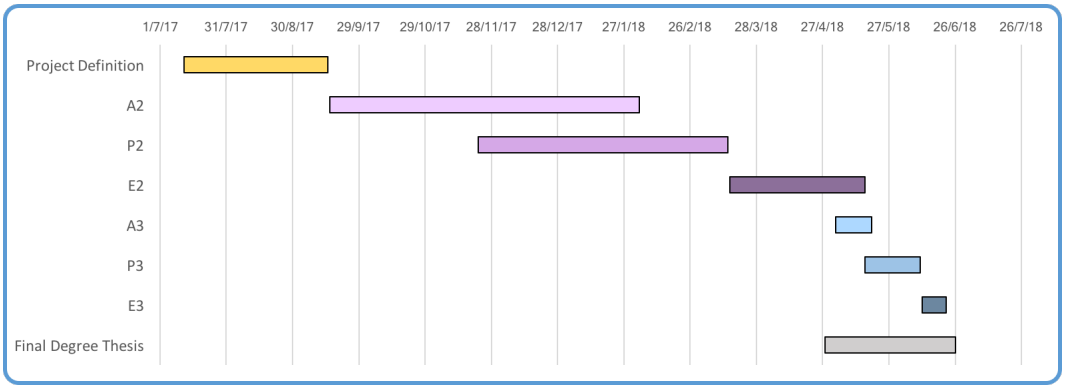

Figure 2.2: Gantt diagram.

1. Project definition, decision of the idea and main goals.
2. (A2) Design of the SECA architecture and library.
3. (A2) Design of the virtual tutor that is going to be integrated in the CP application.
4. (P2) Implementation of the SECA library in Python.
5. (P2) Set up of a Django server to host the SECA virtual tutor and connection with the Unity CP application.

6. (P2) Implementation of the first prototype of the virtual tutor using basic keyword-pattern matching functions and AIML in Python and integration with the CP Unity application "Energy Madness".
7. (E2) Pilot tests to check the application before the complete evaluation.
8. (E2) Evaluation of the first prototype of the gamified CP application, integrating the virtual tutor using basic NLP, with real users (children from some schools) thanks to the collaboration of Rubí Brilla.
9. (A3) Analysis of the gathered data to improve the application and research of different Machine Learning algorithms to be trained using the gathered data from the first evaluation to improve the NLP of the agent.
10. (P3) Implementation of a new prototype of the application. ML algorithms included to enhance the conversational capabilities of the virtual tutor and other improvements performed considering the feedback obtained from the users in the first evaluation.
11. (E3) Evaluation of the enhanced version of the application with other real users.
12. (E3) Detailed analysis of the obtained data after the new evaluation and comparison with the results obtained in the previous ones.
13. Final Degree Project Thesis preparation and writing.

Chapter 3

Sentient Embodied Conversational Agents

This chapter defines Sentient Embodied Conversational Agents (SECAs), introduces an architecture based on the client-server model and formalizes the design of a software library that allows SECAs embedding in different applications.

3.1 SECA Definition

A *Sentient Embodied Conversational Agent (SECA)* is, as its name indicates, an Embodied Conversational Agent (ECA) [24] with some human-like sentient qualities that make it capable of perceiving and "feeling" certain aspects and response to them.

Concretely, the SECA architecture presented in this project thesis, allows the implementation of agents with the following features:

- Capability to engage users in complex structured conversations with natural language by covering different types of dialogs which can start at the user's request or proactively.
- Certain knowledge to focus the conversations in specific topics.
- Memory to remember things that have been said and avoid being repetitive.
- Predefined personality based on Kshirsagar et. al.'s proposal [41] and possibility of showing different appearances depending on the current mood and emotion.
- Customizable needs based on Maslow's Hierarhy of Needs [43].
- Capability to empathize with the user.

The SECA Architecture and software Library that are going to be presented in more detail in the following sections facilitate the inclusion of these type of agents in different applications and platforms. They might be especially useful in applications requiring proactive and sensitive behaviours like the ones expected for example in educational applications.

3.2 SECA Architecture

As for any conversational agent, SECA's platform requirements are modularity, extensibility, real-time visualization, performance, and storage capacity. This project uses a client-server model which can achieve this previous specification.

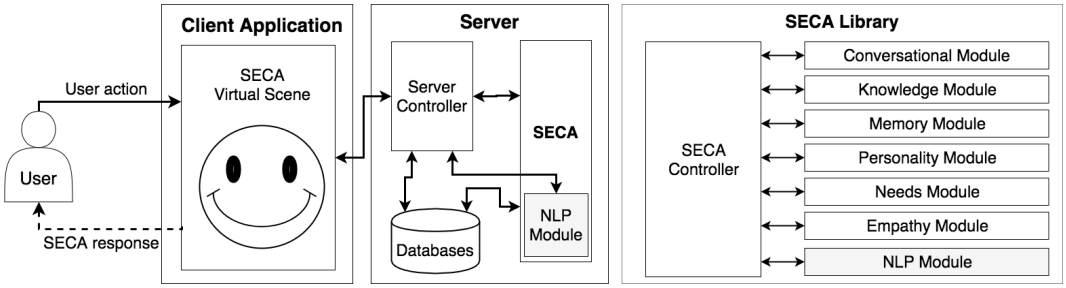

Figure 3.1: Left: Client-Server architecture. Right: SECA library structure.

Left side of Figure 3.1 depicts SECA's client-server architecture. The client application embodies the SECA within its user interface and the Server hosts the agent, with a server controller communicating the client application with the SECA and the database. Additionally, right-hand side of Fig. 3.1 details the library provided to implement SECAs, which consists of a set of modules and its own controller.

Both the architecture and library proposals are inspired on METO [20], a Motivated and Emotional Task-Oriented ECA. Particularly, this project redesigns METO's Personality Module, Needs Module and Tasks Module, being the later renamed as Conversational Module, and creates new Knowledge, Memory, Empathy and NLP Modules. Section 3.3 explains each of these modules in more detail.

Next subsections are devoted to provide some more details about main components in the server part of the SECA architecture.

3.2.1 Server Controller

The Server Controller manages the communication between the client application and the SECA. It is also in charge of managing all the interactions with the databases to store most of the application related data.

Natural Language Processing Module is necessary to support the conversational capabilities of the SECA. The Server Controller initializes just once an NLP Module that it is shared by all the instances of a certain SECA stored in the Server. It has been designed this way so that all the models that are needed are loaded just one time, making the process more efficient in terms of time and memory.

3.2.2 SECA

This module refers to an instance in the server of a certain agent using the SECA Library from the right side of Figure 3.1 which will be explained in more detail in Section 3.3.

Each SECA is associated with a certain user. It uses the NLP Module initialized in the Server and directly stores and recovers certain data from the databases to recover its current state in case the server is restarted.

3.2.3 Databases

To ensure persistence of the agents in case the server needs to be restarted, a database with enough independent tables to store the current state of each agent is necessary. Figure 3.2 shows the basic ones, which ensure:

- Personality state persistence by storing the current mood and emotion of each agent (identified with a unique ID associated with the user using that agent).
- Empathy and Memory modules persistence by storing the necessary data (explained in more detail in Section 3.3) and identifying it with a composite key considering an identified for the agent and for the specific empathy component or memory.

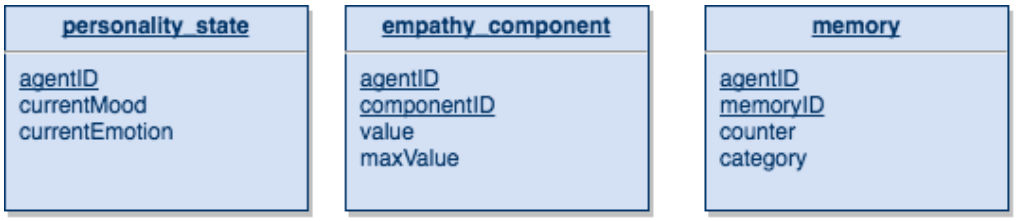

Figure 3.2: Tables to ensure personality, empathy components and memory persistence.

However, it may also be convenient to have other tables storing interaction data from the user to analyse the behaviour of the SECA or to gather data in a format suitable to train new machine learning models that improve its performance (see Figure 3.3).

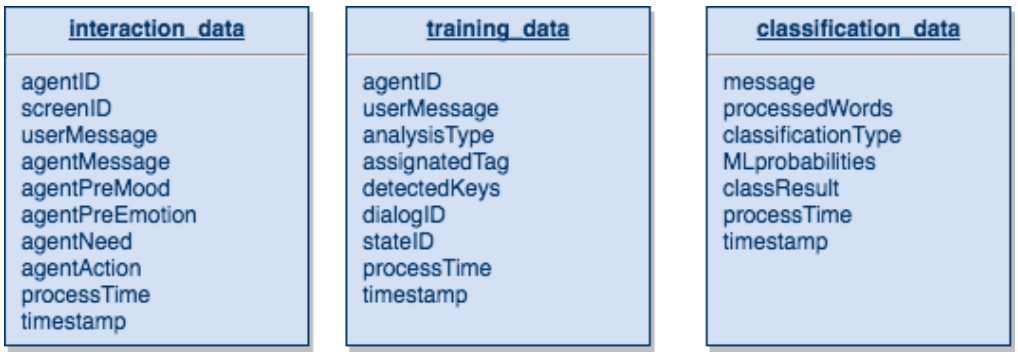

Figure 3.3: Tables to facilitate interaction, training and classification data gathering.

3.3 SECA Library

The SECA software library this project proposes consists on a structure conformed by a set of modules that take care of the different features that characterize the agent and a controller that at the same time manages all modules and controls the communication between them.

Its design is inspired on the one presented in the paper "METO: a Motivated and Emotional Task-Oriented 3D Agent" [20] which was partly developed in Python in another Final Degree Thesis [52]. However, this work designs and implements new modules to improve the agent performance and redesigns completely the agent structure to ensure independence between modules and to make it easily reusable and customizable though inheritance when creating other SECAs.

UML Class Diagram in Figure 3.4 specifies the structure and details the modules (specifying the relations between them) of the SECA Library implemented and documented as a Python module (which can be found in the source code of the project, `../Earth_Server_and_SECA_Library/chatbot/conversational`).

The following subsections formalize the controller and all the different modules that conform the library.

3.3.1 SECA Controller

The *Sentient Embodied Conversational Agent* Controller (`SECA` class in the Python library), initializes the SECA and all its modules and manages the communication between them. SECA Controller is also in charge of saving and recovering data from databases to guarantee the persistence of the agent personality, memory and empathy. To do so, it provides methods to access and update information from the different modules as Figure 3.4 illustrates. There are also two other methods that are worth mentioning:

- `getAnswer`, which given a message, the current conversation context and some parameters, returns the most appropriate answer.
- `update`, method used by `getAnswer` that, given some AIML Keywords, extracts the correspondent text and updates the agent's mood and emotion.

Notice that the actual implementation of a specific SECA requires to inherit from this class and to overwrite some methods –such as the loading and saving functions–, depending on the actual platform where the agent has been implemented and is going to be deployed. The constructor of the SECA only needs an ID, the information related with its base personality and a reference to a NLP Module to be initialized. However, other information related with the Conversational, Knowledge and Memory Modules should also be added immediately after initialization to guarantee the correct performance of the agent.

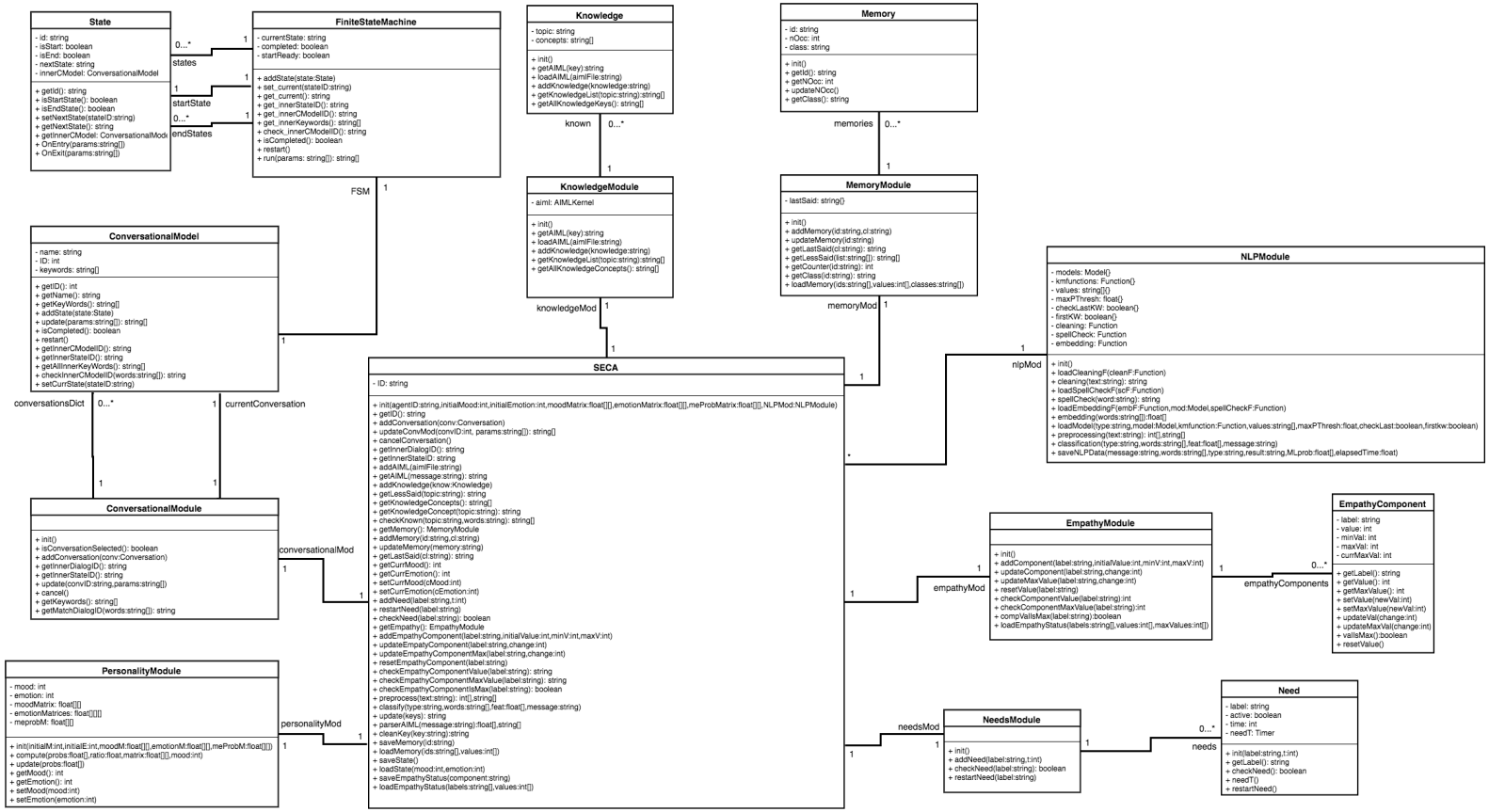

Figure 3.4: SECA Library UML Class Diagram.

3.3.2 Conversational Module

Of all the modules in the SECA Library, the conversational one allows SECAs to engage in rich conversations with users by supporting different structured conversations in natural language. These conversations facilitate user interaction in applications –such as educational apps, citizens' portals, or apps for the elderly– characterised by rich contents and/or complex tasks. Eq. (3.1) formalises this module as a tuple composed by a set of n *Conversations* ($Conv_i$) together with a reference to the current one (*currConv*). Conversations can be tailored to different application contexts by defining them as a set of specific *Dialog Types* ($DT_1, \ldots, DT_m$ in Eq. (3.2)).

$$ConversationalModule = \langle \{Conv_1, \ldots, Conv_n\}, currConv \rangle \tag{3.1}$$

$$Conv_i = \langle DT_1, \ldots, DT_m \rangle, \;\; i \in [1, n] \tag{3.2}$$

Each Dialog Type (DT) constitutes an interaction template specified by means of a Finite State Machine (FSM). DTs can be considered as the building blocks of dynamic conversations: different conversations can reuse several DTs and they can change dynamically during a conversation. In fact, conversations are also specified as hierarchical Finite State Machines (FSMs) where states can be DT's FSMs.

Consequently, the interaction structure is different, for example, if the dialog revolves around the user asking a FAQ or if the SECA aims at arousing user's interest. That is why it is also possible to distinguish between proactive DTs –where the agent starts the dialog– from those on demand. Figure 3.5 illustrates a conversation where the SECA can start DT_1 and DT_2 in a proactive way while it will only start DT_3 upon user's request.

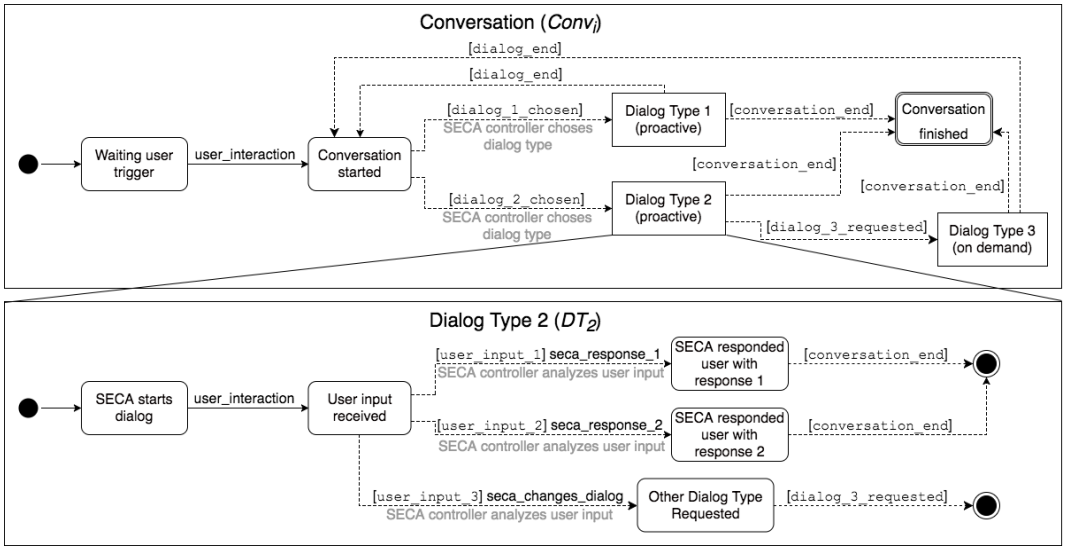

Figure 3.5: FSM diagrams representing a Conversation and one of its Dialog Types. Continuous lines indicate user input and discontinuous lines denote SECA's internal processing.

As previously mentioned, states in the FSM can correspond to basic conversational states or dialog types and, thus, the design supports modular and nested FSM specifications. As for state transitions, they are:

1. Dynamic, since there is a function which determines the next state on run-time.

2. Context dependent, so that the current conversation, current state, and last user input are considered when choosing the next state.

3. Able to produce a natural conversation flow, since the SECA's messages addressed to the user are computed considering, not only the most recent user input, but also the data stored in the Knowledge, Memory, Personality, and Empathy Modules.

SECA's behaviour is mostly performed by the `getAnswer` method from the SECA Controller. This function is invoked using information from most modules due to the complexity of the conversation flow. On the one hand, some transitions —such as the one between "Conversation Started" and "Dialog Type 1" in the conversation of Fig. 3.5 – require the SECA to be proactive and choose next state, somehow randomly, but without repeating contents nor abusing of the same interaction structures (i.e., DT's). On the other hand, alternative transitions –as the one between "User input received" and "Other dialog type requested" in Fig. 3.5)– invoke the function to analyse the user input and to be able to detect specific on demand DT changes.

Since the needed behaviour may vary slightly depending on the agent, `getAnswer` needs to be overwritten. However, an implementation example is provided in Chapter 6, where this project uses the SECA software Library to implement a virtual tutor.

Regarding the implementation of the Conversational Module found in the Library (see Fig. 3.1), it contains a dictionary storing conversations identified by a unique integer and an identifier to know which conversation is being performed by the agent. It also has a set of methods that can be used to access information of the different conversations and to update them.

As explained, conversations are hierarchical and they are represented by FSM whose states can be dialog types, which are also represented by FSM. Though conceptually they represent different things, their structure is the same and that is the reason why the `ConversationalModel` class has been provided in the Library to facilitate the implementation of both through inheritance. Each Conversational Model is identified with a name, a numeric ID and an optional list of keywords, and it is represented in the form of a Finite State Machine where each state can also have a Conversational Model inside (nesting possibility).

This project implements FSM having a certain code as base [11]. When the FSM is updated, and there is a state transition, methods `OnExit` and `OnEntry` are executed when the current state is exited and the next state is entered respectively. These methods must be overwritten to provide certain state transitions information depending on the conversation or dialog type to be implemented.

3.3.3 Knowledge Module

This module manages the static knowledge associated to the targeted conversations. Eq. (3.3) formalises it as a tuple of *extAIML* and a set of predefined *Knowledge components* (K_j). Moreover, Eq. (3.4) specifies each K_j as a set of s_j concepts related to a $topic_j$. Thus, for example, the $topic_1$ = `colour` could be related to the set of concepts {`purple, blue`} when defining K_1.

$$KnowledgeModule = \langle extAIML, \{K_1, ..., K_r\} \rangle \tag{3.3}$$

$$K_j = \langle \{concept_1, ..., concept_{s_j}\},\ topic_j \rangle,\ \ j \in [1, r] \tag{3.4}$$

extAIML is an extension of AIML that borrows part of the specifications from METO AIML [20] to include personality and emotions' parameters that can have an impact in the personality changes of the agent thanks to the implementation of the Personality Module (see Section 3.3.5).

When using the SECA library, all possible text answers the SECA can provide given an AIML keyword must be specified in the `.aiml` files that will be loaded to the AIML Kernel, using the following format:

```
<category>
    <pattern>AIML KEYWORD</pattern>
    <template>
        [Related_to_emotion_1, Rel._to_em._2, ..., Rel._to_em._n]
        Text used when the agent changes to emotion 1.%
        Text used when the agent changes to emotion 2.%
        Text used when the agent changes to emotion 3.%
        ...
        Text used when the agent changes to emotion n.%
    </template>
</category>
```

Since each `Related_to_emotion_x` value denotes the probability of influencing the SECA to express emotion x, according to the definition of probability:

$$\sum_{i=1}^{n} Related\_to\_emotion\_i = 1,\ \ Related\_to\_emotion\_x \in [0, 1] \tag{3.5}$$

The SECA Controller has a pair of functions to clean AIML keys and parse the content obtained from the AIML files.

3.3.4 Memory Module

Memory Module is in charge of storing dynamic information and avoiding repetitive conversations. Particularly, it manages both Long-Term (*LTM*) and Short-Term memories (*STM*). Eq. (3.6) specifies *LTM* as a set of *t Memories*.

$$MemoryModule = \langle LTM = \{M_1, ..., M_t\}, f_{STM} \rangle \tag{3.6}$$

Each memory M_l is defined as a tuple composed by (see Eqs. (3.7), (3.8) and (3.9)):

1. An *id*, which can identify a specific dialog type or concept.
2. A *class* identifying whether that id corresponds to a DT or to a concept (i.e., Knowledge).
3. And *nOcc*, which is its number of occurrences and it is initialised to 0.

$$M_l = \langle id, class, nOcc \rangle, \;\; l \in [1, t] \tag{3.7}$$

$$id \in Ids = \{DT_1, ..., DT_m\} \cup \{concept_1, ..., concept_s\} \tag{3.8}$$

$$class \in \{\texttt{DT}, \texttt{Knowledge}\}, \;\; nOcc \in \mathbb{Z}_{\geq 0} \tag{3.9}$$

And the STM function serves as a mechanism to retrieve the most recent used id of a given class (i.e., `DT` or `Knowledge`):

$$f_{stm} : \{\texttt{DT}, \texttt{Knowledge}\} \rightarrow Ids, \;\; f_{stm}(class) = id \tag{3.10}$$

In the library, this module stores all memories in a dictionary (LTM) and also contains another dictionary that stores the last memory that the agent has for each class (STM).

This module also has a set of methods to retrieve memory related data. One of the most useful ones (`getLessSaid`) returns all the less used concepts of a certain class. Consequently, this method is used when choosing a DT or concept randomly while avoiding being repetitive.

3.3.5 Personality Module

Personality Module provides personality traits to the SECA. Its idea follows Kshirsagar et al.'s work [41] in defining Personality, Moods, and Emotions as three interdependent layers and is based on METO [20]. However, this project extends the idea presented in METO so that SECAs present a fixed personality (specified when initialized) allowing a different number of moods ($numM$) and emotions ($numE$) that change thanks to several transition probability matrices that are stored in the Personality Module.

In this work current Mood (*currMood*) influences the emotions layer (lower one) and consequently, the following current emotion (*currEmotion*) change. Assuming emotions last a shorter amount of time and are more tied to actions, in the specification they are calculated in the first place so that user input can affect them and they can also slightly affect the following mood change [38].

Personality Module (see Eq. (3.11)) needs the following transition matrices to perform the current mood and emotion update:

1. One $numM \times numM$ *Mood Transition Matrix* (MTM) that changes the current mood with respect of the previous one.
2. $numM$ (one for each mood) $numE \times numE$ *Emotion Transition Matrices* (ETM_μ, $\mu = 1 \dots numM$) to change the current emotion depending on the previous one.
3. One $numE$ x $numM$ *meProbability Matrix*($meProbM$) that for each emotion, has a vector of probabilities that slightly affects the mood changes of the agent.

Moreover, when user interactions trigger the SECA Controller to update its current mood and emotion (*currMood* and *currEmotion*), a vector of `Related_to_emotion_x` probability values (P) specified at the *extAIML* from the Knowledge Module, is obtained and intervenes in the update of the current mood and emotion values (see Eq. (3.12)).

$$PersonalityMod = \langle currMood, currEmotion, MTM, ETM_1, ...ETM_{numM}, meProbM \rangle \quad (3.11)$$

$$P = \langle P_1, ..., P_{numE} \rangle, \;\; P_i \in [0,1] \;\; where \sum_{i=1}^{numE} P_i = 1 \quad (3.12)$$

Tables 3.1, 3.2 and 3.3 show in a detailed general way the contents that probability matrices store.

Table 3.1: Mood Transition Matrix example.

		newMood			
		M_1	M_2	...	M_{numM}
currMood	M_1	$P(M_1\|M_1)$	$P(M_2\|M_1)$	...	$P(M_{numM}\|M_1)$
	M_2	$P(M_1\|M_2)$	$P(M_2\|M_2)$	...	$P(M_{numM}\|M_2)$
	...	...	...	$P(newM\|currM)$	...
	M_{numM}	$P(M_1\|M_{numM})$	$P(M_2\|M_{numM})$	...	$P(M_{numM}\|M_{numM})$

Table 3.2: Emotion Transition Matrix example.

		newEmotion			
		E_1	E_2	...	E_{numE}
currEmotion	E_1	$P(E_1\|E_1)$	$P(E_2\|E_1)$	...	$P(E_{numE}\|E_1)$
	E_2	$P(E_1\|E_2)$	$P(E_2\|E_2)$	...	$P(E_{numE}\|E_2)$
	...	...	...	$P(newE\|currE)$	...
	E_{numE}	$P(E_1\|E_{numE})$	$P(E_2\|E_{numE})$	...	$P(E_{numE}\|E_{numE})$

Table 3.3: meProbability Matrix example.

		Mood Impact			
		M_1	M_2	...	M_{numM}
currEmotion	E_1	$P(M_1\|E_1)$	$P(M_2\|E_1)$	...	$P(M_{numM}\|E_1)$
	E_2	$P(M_1\|E_2)$	$P(M_2\|E_2)$	...	$P(M_{numM}\|E_2)$
	...	...	...	$P(newM\|currE)$	...
	E_{numE}	$P(M_1\|E_{numE})$	$P(M_2\|E_{numE})$	...	$P(M_{numM}\|E_{numE})$

Eqs. (3.13) and (3.14) detail the operations that update the current personality values of SECAs when users perform an action (send a message). Since user actions might have a quick impact in the agent, the module calculates first the update of the agent's current emotion taking P into account with a certain weight ($w_e = 0.4$) and then, modifies its mood considering that the new emotion might slightly ($w_e = 0.2$) affect its change.

$$P(newE_r) = (1 - w_e) * P(newE_r|currEmotion) + w_e * P(newE_r) \quad (3.13)$$

$$P(newM_s) = (1 - w_m) * P(newM_s|currMood) + w_m * P(newM_s|currEmotion) \quad (3.14)$$

3.3.6 Needs Module

This module manages and brings to light the different needs the agents may have. Based on Maslow's Hierarchy of Needs [43] –which assumes self-realization requires the fulfilment of some basic and social needs– seeks to develop an emotional bond with the user.

Thus, for example, SECAs can show proactive behaviour due to their communication needs. Other needs may be related to specific goals –such as completing certain tasks– of the application that embeds the agent.

In general, Eqs. (3.15) and (3.16) formalise the module as a set of p *Need*s, where each N_k is defined in terms of: a *label* identifying it; a *state* signalling if this need is accomplished (i.e. `inactive`) or not (`active`); and a *time* counter which is reset upon reactivation.

$$NeedsModule = \langle N_1, ..., N_p \rangle, \;\; N_k = \langle label, state, time \rangle, \;\; k \in [1, p] \tag{3.15}$$

$$typeOf(label) = \texttt{string}, \; state \in \{\texttt{active,inactive}\}, \;\; time \in \mathbb{R}_{\geq 0} \tag{3.16}$$

In the software library, this module has a dictionary of needs identified by their label and each need, apart from the label and a Boolean to know its state, has also associated a custom timer which, every *time* seconds, invokes a function that changes the state of the need to `active` (by setting the correspondent Boolean to `True`). For example, SECAs could have an attention need that becomes `active` after 10 seconds of having no interaction.

This module also has a method to restart the need by setting its state to `inactive` (i.e. changing the correspondent Boolean value to `False`) and starting again its timer.

3.3.7 Empathy Module

Empathy Module tries to guess user's thoughts and feelings from their interaction. As for the needs, empathy is considered with the aim of establishing stronger bonds with the user [28]. Equations (3.17) and (3.18) formalise SECA's *Empathy* Module as a set of q "user state" indicators (E_j) whose bounded (between $minV$ and $currentMax$) numerical values are monitored along the interaction.

$$EmpathyModule = \{E_1, ..., E_q\} \tag{3.17}$$

$$E_j = \langle label, value, minV, \tau, currentMax \rangle, \; j \in [1, q] \tag{3.18}$$

$$currentMax \in [\tau, \infty] \;\; where \;\; minV, \tau, currentMax \in \mathbb{Z} \tag{3.19}$$

$$-\infty < minV < \tau < currentMax < +\infty \tag{3.20}$$

$$typeOf(label) = \texttt{string}, \; value \in [minV, currentMax] \tag{3.21}$$

This work assumes that E_j's maximum values might be different depending on the user. Consequently, Eqs. (3.19), (3.20) and (3.21) illustrate how $currentMax$ varies in a range defined thanks to a certain τ indicated when initializing each E_j. Variations can occur real-time depending on the interactions of the user with the SECA.

All in all, SECAs can adapt their behaviour depending on empathy values. Thus, for instance, having a `Tiredness` empathy component whose value increases as conversations get longer may urge the SECA to end conversations earlier. And since some users might get tired later than others, `Tiredness`'s maximum value could be consequently adjusted.

In the provided SECA library, this module stores different empathy components in a dictionary, using their label as a key.

3.3.8 NLP Module

The Natural Language Processing (NLP) module contains functions devoted to preprocessing and analysing user input. As for any standard NLP flow, preprocessing includes text cleaning, spell-checking, and word embedding. Next Chapter 4 further details both the techniques and the NLP flow used in this work.

Chapter 4

Natural Language Processing Module

4.1 Real Time Multi-Problem Multi-Language NLP

Natural Language Processing (NLP) is the field in charge of manipulating natural language using computers for their understanding and for being able to give useful responses [23]. Since this project implements a conversational agent, NLP becomes necessary to analyse the user input. Though the processing flow be the same for any language, each one has some particularities that make the operations performed in each of the steps quite unique.

SECAs' Conversational Module defines predefined structured conversations so that it expects certain user inputs at certain steps. Taking this into consideration, after preprocessing user input the problem is "reduced" to correctly performing some classification of the received message so that the agent can understand what type of message it is and provide the most appropriate answer ($extAIML$ from the Knowledge Module contains the set of predetermined possible SECA answers).

The NLP Module implemented in the Python software library takes this into consideration and provides methods to load different models and functions for each step of the processing flow so that it can be used to implement SECAs understanding any language. Figure 4.1 illustrates the processing flow that can be performed using the NLP Module and that ensures a classification result for a certain user input.

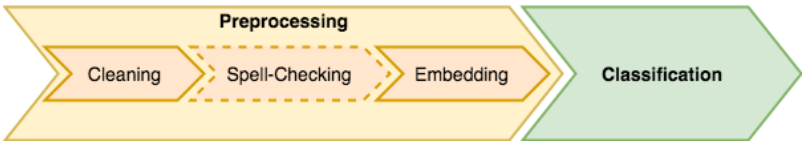

Figure 4.1: NLP flow.

The steps depicted in Figure 4.1 (Preprocessing and Classification) will be explained in more detail in the following sections.

Since SECA architecture is going to be instantiated as a virtual tutor for children who speak Catalan, the operations performed when analysing user input from these users will be used as an example to explain the NLP flow followed. Because there are not many works yet in this language, the techniques this project uses may serve as guidelines to implement further Catalan NLP applications.

Though there exist libraries like NLTK (Natural Language Tookit) for Python, in this project the problem is addressed from scratch to think about the different phases of Natural Language Processing and to tailor the processing to the problem, which requires a reduced number of operations.

Finally, another factor considered during the design of the whole process is the processing time, since the proposed NLP is embedded in an on-line –i.e., real time– conversational application. That is why the NLP Module that this thesis presents opts for simple operations and sometimes discards more complex operations that take longer execution time.

4.2 Preprocessing

The first phase of the NLP flow is preprocessing. User's raw text is very difficult to classify without applying some preprocessing, mainly because of punctuation presence, other non-meaningful characters, or spell-checking mistakes that users commit. That is why each time the user communicates with the SECA, the NLP Module preprocesses the inputted sentence by undergoing: cleaning, spell-checking, and embedding, before classifying it. Preprocessing is necessary to standardize text and to facilitate further manipulation and analysis.

In order to perform preprocessing using the provided SECA library, it is very important to previously load the necessary models and functions. The following section provides the requisites to perform each step using the Catalan language as an example.

4.2.1 Cleaning

To perform cleaning, a specific function that returns a list of cleaned words must be loaded to the NLP Module using the `loadCleaningF` method. In the case of Catalan, this project proposes a cleaning function at word level (`cleanCAT` in the `Cleaning.py` file).

After removing some special characters, `cleanCAT` performs *tokenization*, which consists on breaking the stream of text into words (in this case), phrases or other meaningful elements. Since Catalan is a space-delimited language, words can easily be processed individually by splitting text using white-spaces. However, sometimes users forget white-spaces when using other sentence or topic delimiters like ",", ".", "?" or "!". Therefore, the function also considers those four punctuation marks for the tokenization process.

After that, the following cleaning is applied to each word:

- Removal of one character length words (notice that this corresponds to a simplified *stop words removal*).
- Removal of the reduced form of Catalan pronouns and the correspondent punctuation signs like "l"', "'ns", "-me" etc (simplified *stemming*).
- Removal of other punctuation signs ("!", "?", ".", etc.).
- Removal of repetitive characters (in this manner, "holaaa" becomes "hola"). Though, the function allows "rr", "ll" and "ss" because they are possible digraphs in Catalan.
- Consideration of some common abbreviations (for example, "pk" becomes "perque").
- Word transformation to upper-case.
- Word transformation to Unicode "utf-8" to work with special characters.

4.2.2 Spell Checking

Spelling mistakes are common when using keyboards of electronic devices to input written text because of many reasons (keys proximity, use of abbreviations, etc.).

Though it would make sense to perform spell checking over all inputted words, processing time is very important in this kind of applications because the user is interacting real-time with the conversational agent and output should be given as quickly as possible. That is why, the NLP Module of the software library, only performs spell checking over words whose embedding cannot be provided. Then, it performs embedding again over the spell-checked words.

In the case of Catalan, the function loaded to perform spell-checking is the Spelling Corrector implemented by Peter Norvig in Python [48]. It achieves an accuracy of around 80-90%, corrects approximately 1 word in 0.1 seconds and contemplates spelling mistakes like missing characters, transposed characters and wrong splitting. This work modifies the code provided in his page to contemplate all characters from the Catalan language (addition of "àèéíïòóúüç" characters).

This spell-checking function works by using some probabilities to guess which is the most possible correction of a word given some candidates. When checking a word, it considers:

- The probability of word w appearing as a word of Catalan text, $P(w)$.
- And the probability of writing w when the user wanted to write c (which is a candidate word for correction), $P(w|c)$.

The operations it performs are:

$$argmax_{c \in candidates} P(c|w) \tag{4.1}$$

Using Bayes's Theorem becomes:

$$argmax_{c \in candidates} \frac{P(c)P(w|c)}{P(w)} \tag{4.2}$$

Which can be simplified given that $P(w)$ is the same for all candidates:

$$argmax_{c \in candidates} P(c)P(w|c) \tag{4.3}$$

The code manually generates candidates using force brute algorithms which calculate all possible edits at distance one or two from the initial word. As an example, supposing the given word is "energa", some of the candidates generated by the code are "en" and "erga" by splitting, "enera" by deleting, "neerga" by transposing, "energe" by replacing and "energia" by inserting. The algorithm considers that the group of candidates at distance one are more probable that the ones at distance two and deduces that $P(w|c)$ is always the same given that every candidate from a certain distance group has the same probability. Then, everything is reduced to calculate:

$$argmax_{c \in candidates} P(c) \tag{4.4}$$

To find the previous probability, this project uses the modification of Peter's Norvig code proposed by a Kaggle user [13] that takes profit of Word2Vec ordering of words to approximate candidates apparition probabilities.

Word2Vec [42] is a technique developed by T. Mikolov et. al which finds vector representations of words from very large datasets in a short period of time. Consequently, NLP Module also uses this technique to find embeddings (see Section 4.2.3).

To use a Word2Vec model in the code, it needs to be trained using an appropriate dataset. In the case of Catalan, this thesis uses some github code [18] to generate a vector model through a dataset of all wikipedia articles written in Catalan until 04/20/2018 [2] and `WikiCorpus` and `Word2Vec` functions from Python's `Gensim` library.

All in all, the function implemented in this project for the Catalan language case, that used the previously explained source codes as reference, is `getCorrection` from the `SpellChecker` class of the `SpellChecker.py` file.

4.2.3 Embedding

Given the list of words obtained after performing cleaning and spell-checking (in certain cases), the final preprocessing step consists on obtaining an embedding for the given user input to facilitate the following classification step.

The purpose of word embedding algorithms is finding an N length real-valued vector representation of each word so that similar words get similar values and are placed in a closer position in an N dimensional space. And, since Word2Vec provides this functionality and has been proven to give good results, it will be used in the `w2vFeatures` (`Embedding.py`) method loaded as an Embedding Function in the Catalan NLP Module.

The problem with Word2Vec is that when a word is not found in the model, it cannot provide an embedding. This is a very common problem when a model is trained with correct data that has almost no spelling mistakes compared with the input text. And that is why NLP Module performs spell-checking over the words that have not been found in the model. Both spell-checking and embedding, use the same Word2Vec model.

To obtain embeddings, Word2Vec works at word level. However, classification usually needs to be performed at sentence-level. Consequently, after obtaining the word embedding for the different words of the user input, the proposed code calculates the average of all the word embeddings to get the sentence embedding which will be used later for classification. The algorithm calculates the averaged features vector using only words, found in the model, whose embedding was provided.

Embedding functions to be used in the NLP Module from the provided SECA library are assumed to return a two items list consisting of a list of features and a list of words including both originals (in case corrections are wrong) and corrections.

4.3 Classification

As previously mentioned, classification comes after processing users' input. To perform it in the NLP Module from the provided SECA library, it requires the loading of some classification problem models. A *classification problem* represents a process that, from an input, detects the class where the input belongs given a set of possible classes. Sentiment Analysis constitutes an example of a classification problem which requires detecting if an input corresponds to an affirmation, a negation or other.

4.3.1 Classification Problem Models

The different elements that need to be specified when a classification problem model is loaded to the module are (hereafter we illustrate those components by considering the SA: Sentiment Analysis classification problem):

- *Tag* to identify the classification problem that is going to solve the model (for instance, `SA`).
- *Machine_Learning_model* used for classification (for example, a previously trained SVM Model). This will be further explained in Chapter 5.
- *Keyword_matching_function* used for classification and which returns the prediction and a Boolean indicating if the result is relevant or not.
- List of possible *class_values* (for example, `[Affirmation, Negation, Other]`).
- *Threshold_value* to decide if the result obtained with machine learning is reliable.
- *further_classification*, a Boolean indicating if some further classification needs to be tried in case the classification result is an ambiguous class (e.g., `Other`).
- *keyword_matching_should_be_tested_before*, another Boolean to indicate if a simple keyword-pattern matching should be tried before using the machine learning model for classification. This is useful in case the ML model cannot achieve good results after being trained with certain data.

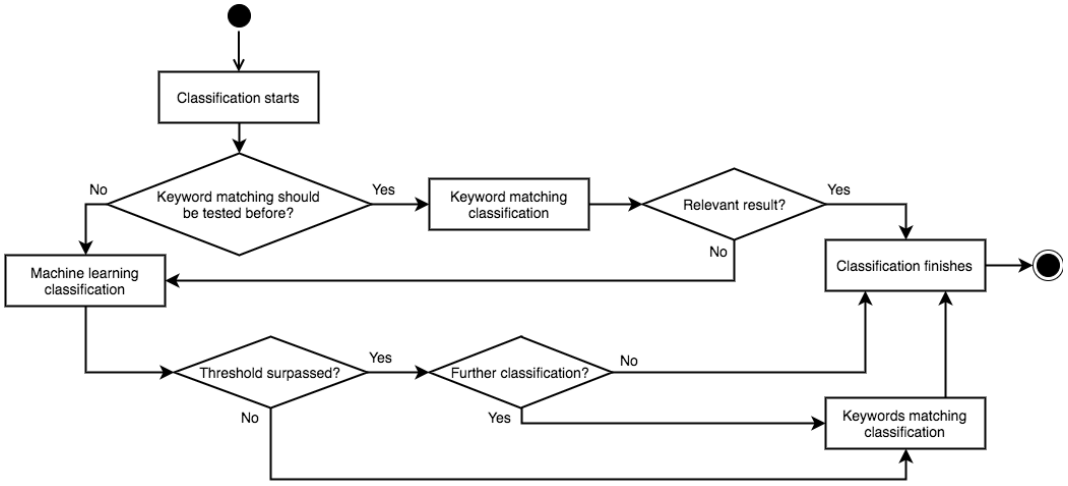

Figure 4.2: Classification flow using keyword-pattern matching functions with Machine Learning integration.

4.3.2 Keyword Matching with Machine Learning integration

This work proposes an hybrid system using keyword matching and machine learning to perform classification in the NLP Module provided in the SECA software library. The reason behind it is because depending on the classification problem, it might be hard to specify enough keywords to perform a good classification and on the other hand, in some cases the data available to perform good classifications based on Machine Learning might be insufficient.

Figure 4.2 describes the flow followed when classification is performed using the Booleans $further\_classification$ and *keyword_matching_should_be_tested_before* specified when each classification problem model is loaded to the NLP Module, the *Threshold_value* and the relevancy Boolean that return all *Keyword_matching_function*s.

`classification` method from the NLP Module directly returns a value corresponding to one of the possible class values.

4.4 NLP Example

This is an example of NLP processing of some input in Catalan language to illustrate the NLP process that this project proposes.

- User input:
  - "Què signifca energa solar?"
- Preprocessing:
  - Cleaning output:
    - `[u'QUE', u'SIGNIFCA', u'ENERGA', u'SOLAR']`, where `u` stands for "unicode" text.

  - Embedding and Spell checking for words whose embedding was not found. Console output:
    * `Word signifca not found in model`
    * `Word signifca has been corrected to significa`
    * `Word energa not found in model`
    * `Word energa has been corrected to energia`
  - Average of word embeddings and new word list including the proposed corrections after spell-checking (preprocessing function output):
    * `[[-1.2273177355527878, 0.6118546929210424, -2.0599005334079266, ...-2.2642883956432343, -0.528718039393425,2.7149559259414673], [u'QUE',u'SIGNIFCA',u'SIGNIFICA',u'ENERGA',u'ENERGIA',u'SOLAR']]`. It stores all generated words during the process so that, in case of using the keyword-pattern matching function later, all possibilities are contemplated (because maybe the spell-checker could not provide an appropriate word correction).

- Classification (further details in Chapters 5 and 6):
  - Input:
    * Preprocessing output i.e. features vector and list of keywords.
  - Topic Matching (TM) classification problem with the following settings (see Tables 6.4 and 6.8):
    * Class values = `[ask, concept, CP, Other]`
    * *further_classification* = `True`, so that if ML predicts `Other` class, keyword-pattern matching function tries classification too. By doing so, the problem with unbalanced supports (i.e. certain classes having more available data than others when training occurs) tries to be fixed.
    * *keywords_matching_should_be_tested_before* = `False`, so that classification directly starts with ML since the model being used proved to have high prediction accuracy during the training.
    * *Threshold value = 0.8*. Because the precision score of the ML model is also high, the probability of being the chosen predicted class the right one is high, and consequently the chosen threshold value is quite permissive.
  - Steps performed
    * ML result:
      · Right prediction probability value = 0.990414937642
      · Class: `concept`

- Output:
  - `concept`

Chapter 5

Machine Learning and NLP

5.1 Introduction

In Computer Science, more specifically in the Artificial Intelligence field, there is the *Machine Learning* (ML) discipline, which seeks allowing computers to learn by themselves. In the case of Natural Language Processing, ML algorithms suppose a powerful tool to generate models that might provide good results when used in classification problems (see Section 4.3).

However, depending on the classification problem, some algorithms might give better results than others and that is why, this project developed a system performing Grid Search with Cross Validation to select the most appropriate hyper-parameters to train each Machine Learning model with the available training data.

Though the proposed system can be used in general, in this project it has been used with the following selection of different types of ML algorithms to train data in Catalan language:

- Support Vector Machines [36].
- Random Forest [12].
- Gradient Tree Boosting [7].
- Logistic Regression [6].
- Multi-Layer Perceptron [9].

Note that this work has not considered the usage of Deep Learning because of the small amount of training data available.

Section 5.2 briefly introduces each of these Machine Learning algorithms and Section 5.3 details the steps followed during the training process of the models, that then can be used in the NLP Module.

The provided code is implemented in Python using the `scikit-learn` library.

5.2 Classification Algorithms

All considered Machine Learning classification algorithms are supervised, i.e. training data must be provided with the correspondent class tag.

In the case of text, the training data used by the different algorithms is a vector of numerical features, which in this project is obtained in the embedding phase of the pre-processing explained in Subsection 4.2.3.

Moreover, all algorithms allow multi-class classification (i.e. having more than two different class tag values).

5.2.1 Support Vector Machines

Support Vector Machines (SVM) [36] [57] are a supervised Machine Learning algorithm used for classification.

Given a binary classification problem, SVM algorithms try to find the optimal hyperplane that separates the data from the two classes i.e. the hyperplane with the maximal margin of separation between the two classes.

In certain cases, data needs to be mapped to a infinite-dimensional feature space to solve the previously explained optimization problem.

Python's `scikit-learn` library implements SVM [14] using the one-vs-one scheme for multi-class classification problems with the `SVC` class.

5.2.2 Random Forest

Random Forest (RF) algorithm is an ensemble method [5] (i.e. combines several base estimators to improve generalizability and robustness) from the family of averaging methods, which means that builds several estimators independently and then averages their prediction results.

This method averages randomized decision trees [3], which are a non-parametric supervised learning method used for classification and regression whose goal is creating a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

Python's `scikit-learn` library implements it with the `RandomForestClassifier` class [12].

5.2.3 Gradient Tree Boosting

Gradient Tree Boosting (GB), it is an ensemble method as RF that can be used for classification and regression. However, it belongs to the boosting methods family, which means that builds base estimators sequentially and tries to reduce the bias of the combined estimator.

Through combing different weak estimators, it aims to find a powerful ensemble.

`GradientBoostingClassifier` class [7] from Python's `scikit-learn` library supports multi-class classification.

5.2.4 Logistic Regression

Logistic Regression (LR) is a generalized linear model for classification [6]. LR uses a logistic function to model the probabilities which describe the possible outcomes of a single trial.

In `scikit-learn`, Logistic Regression [8] uses one-vs-rest scheme by default when solving multi-class classification problems with the `LogisticRegression` class.

5.2.5 Multi-Layer Perceptron

Multi-Layer Perceptron (MLP) [10] is a supervised neural network model that trains a function with an n-dimensional input and a one dimensional output. It can train a non-linear function approximator for classification given some training data (features and target class). Between the input and output layer, there can be different hidden layers that transform the values from the previous layer applying a weighted linear summation and a non-linear activation function.

`MLPClassifier` class [9] from Python's `scikit-learn` library implements a multi-layer perceptron which uses the back-propagation [1] algorithm.

5.3 Training

5.3.1 Data Preparation

Since most models are supervised, the code in this project prepares training data before using it. Given a `.csv` file with some text data, `PrepareTrainingData` function of Python source codes prepares data to be used to train the models of the different classification problems:

1. Loads CSV file with raw text data items and performs class labelling i.e. determines to which class belongs each message depending on the classification problem (requirement of supervised learning). Note that this step discards some messages from the original data file if they are not relevant for the classification.
   To proceed with class labelling, each data item in the `.csv` should have a column with a certain tags (which might be previously typed in a manual way) that identifies the type of message.
2. Preprocesses the raw text data using the same functions loaded in the NLP Module so that models work with text data processed in the same way real-time obtained data will be i.e. performs cleaning, spell-checking (if necessary) and embedding.
3. Generates a new CSV file containing the features vector, the class, and the original message for each data item. This file is the one that will be used for the training.

5.3.2 Grid Search and Cross Validation

Grid Search [16] consists on, given different possibilities for each of the hyper-parameters that can be tuned in the different ML algorithms (i.e. parameters that are not learnt automatically within estimators), testing and evaluating all their combinations so that the most appropriate ones can be chosen depending on the desired result. It becomes necessary because of the vast amount of possibilities there are for the different hyper-parameters.

On the other hand, Cross Validation [30] focuses more on the way of using the provided training data. When large amounts of training data are available and it is randomly distributed, it can be divided into training and testing set (holdout method). However, this is not always the case and even with large amount of data, information left for testing might have been more useful for training in some cases (overfitting of data may happen). K-Fold Cross Validation provides a solution for this by generating different subsets of data that can be used for training and testing alternatively. This method consists on generating k subsets of data so that the holdout method is applied k times using k-1 subsets for the training and the other one for the testing. The final results are averaged. Figure 5.1 illustrates an example of K-Fold Cross Validation.

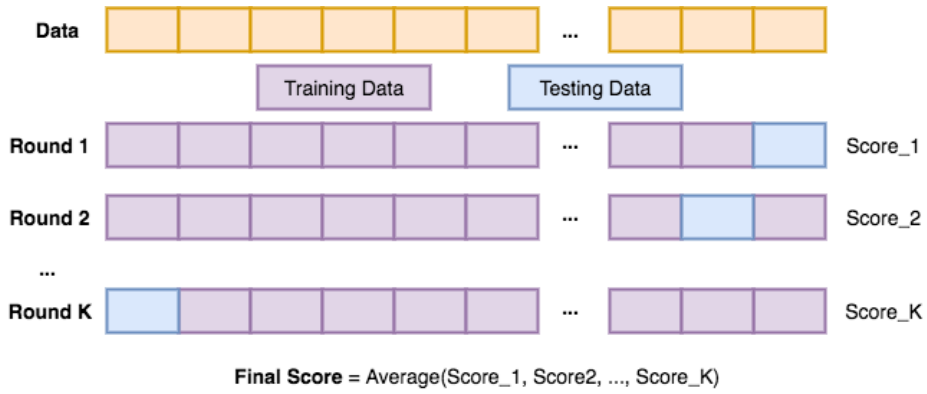

Figure 5.1: K-Fold Cross Validation.

Given a CSV file with the correct data format specified in Subsection 5.3.1 (rows with vector of features, class, and initial message) the operations performed are the same for all classification problems. The code provided in the `ML_GridSearch.py` Python source code contains necessary methods to apply grid search with cross validation to the different ML algorithms presented in Section 5.2 considering different hyper-parameters.

5.3.3 Evaluation Metrics

The provided Python code performs Grid-Search with 10-Fold Cross Validation while tuning parameters considering three different scores: accuracy, precision and recall [40].

Table 5.1 illustrates some concepts that are necessary to understand what the mentioned scores are evaluating. The table considers that multi-class classification is performed, where a certain value has a targeted class and there are other possible classes (1 or more).

Table 5.1: Evaluation concepts.

		Predicted Class	
		Targeted Class	**Other Class**
Real Class	**Targeted Class**	True Positive	False Negative
	Other Class	False Positive	True Negative

- True Positives (TP), i.e. value from the targeted class correctly predicted.
- True Negatives (TN), i.e. value from other classes correctly predicted.
- False Positives (FP), i.e. value from another class misclassified as targeted class.
- False Negatives (FN), i.e. targeted class value misclassified as another class value.

Knowing these definitions, previous score values are:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{Correct\ predictions}{Total\ predictions} \tag{5.1}$$

$$Precision = \frac{TP}{TP + FP} = \frac{Targeted\ class\ correct\ predictions}{Total\ predictions\ of\ targeted\ class} \tag{5.2}$$

$$Recall = \frac{TP}{TP + FN} = \frac{Targeted\ class\ correct\ predictions}{Total\ targeted\ class\ elements} \tag{5.3}$$

From the formulas, it can be deduced that accuracy is the most intuitive value. Usually higher accuracy means better model but, if the number of elements belonging to each class (support) is very different, then it becomes necessary to look at other scores. Precision is important when we want to avoid false positives and recall is important to avoid false negatives.

After performing 10-Fold Cross Validation over a combination of hyper-parameters with the provided code, it generates a classification report like the following one (example of three classes SVM classification where, though the third class has bigger support than the other ones, they are quite balanced):

```
### Parameters: kernel = rbf, C = 1.0000, gamma = 0.0100
###### Mean after 1 times random 10-KFold Cross Validation = 0.9400
              precision    recall    f1-score    support

 Afirmation    0.96         0.90      0.93        515
   Negation    0.95         0.89      0.92        424
      Other    0.93         0.97      0.95        1346

avg / total    0.94         0.94      0.94        2285
```

Where *mean* is the sum of the provided score values for each class divided by the number of classes, while *avg* is the weighted arithmetic mean of the score values given for each class with the corresponding support as weight.

The following formula can be used to calculate f1-score:

$$f1 - score = 2 * \frac{precision * recall}{precision + recall} \tag{5.4}$$

After tuning hyper-parameters over a certain score for all combinations, the code prints out the following information for easier analysis of the data:

```
Highest accuracy score is 0.9527.
Obtained with parameters: kernel = rbf, C = 10.0000, gamma = 0.0100
```

Given the provided evaluation report, all the available data is then used to train the most appropriate model for the correspondent classification problem.

Chapter 6

Virtual Tutor for a Gamified Cultural Probes Application

6.1 Analysis

Cultural Probes (CP) is a well-known technique used to learn about users [32], usually by using a physical package containing the necessary objects to perform some tasks that will allow a more personal data gathering.

Figure 6.1: Pictures of the initial gamified CP Application.

Previous works designed a gamified digital CP for children to gather data about the habits related to energy of children and their families [59] [54] and another Final Degree Thesis implemented in Unity this application [55].

After users read the story designed for the CP (see top-left in Fig. 6.1), they can register and choose an avatar. Then a Dashboard screen shows up (see top-right in Fig. 6.1) and displays four missions available for them:

- *Psychologist's mission*, where children play the role of a psychologist and ask their family about the electrical appliances they use the most and what is their mood while using them (bottom-left of Fig. 6.1 shows the explanation provided before starting this mission).
- *Detective's mission*, where children take the role of a detective and observe the energy related behaviour of their family to finally think about it and decide if it is good or not.
- *Electrician's mission*, where children act as electricians and indicate the electrical appliances they have in each room of their house (see an example in bottom-right of Fig. 6.1). Then they have to indicate how much time use each appliance on labour days and on weekend days and finally, reflect about the difference in use depending if it is labour day or weekend day.
- *Journalist's mission*, where children take the role of a journalist and interview their family about energy related issues.

While doing the missions, users obtain points and badges that can check from the Dashboard Screen. There is also a Ranking Screen with the scores of the different players and users have the possibility of changing their avatar during the game. Users have 8 days to complete the CP and 2 days to complete each mission that start counting after starting each of them. After completing all missions, the game is considered finished.

The evaluation of the gamified CP application showed that users wanted to have more content in the application and wished to learn about concepts related to energy and the application that they did not know nor understand.

Following the User Centered Design iterative methodology, this project embodies a SECA in the application as a virtual tutor providing real-time natural language conversations, with the following goals:

- Enhancing children's User eXperience using the application and making it more educative and engaging.
- Evaluating the previously presented SECA architecture.

The type of users who interact with the SECA are volunteer children between 10-12 years old. To participate in the CP and use the application, all their parents or tutors must provide a compulsory consent form (see Appendix C.1). In this form, they are informed about the contents of the application, data anonymity and the use of it only for research purposes.

This project comprises two iterations of the UCD (see Section 2.1). In the following sections, there are some comments referring to the first prototype of the application (used to gather real conversational data to train ML models and to get a first idea of how the SECA was being perceived by the users). However, most design and implementation details refer to the final version of the application were the SECA is already using the NLP Module integrating keyword-pattern matching and Machine Learning.

The following sections present details about the virtual tutor SECA's design, implementation and integration in the application.

6.2 Design

6.2.1 Earth (Virtual Tutor) Design

The CP application focuses on energy efficiency and environmental sustainability. That considered, the appearance of the SECA virtual tutor is a 2D Earth character with some personification elements like eyes and mouth. As Figure 6.3 depicts, this project presents different Earth illustrations that simulate different moods and emotions (see Section 6.2.3).

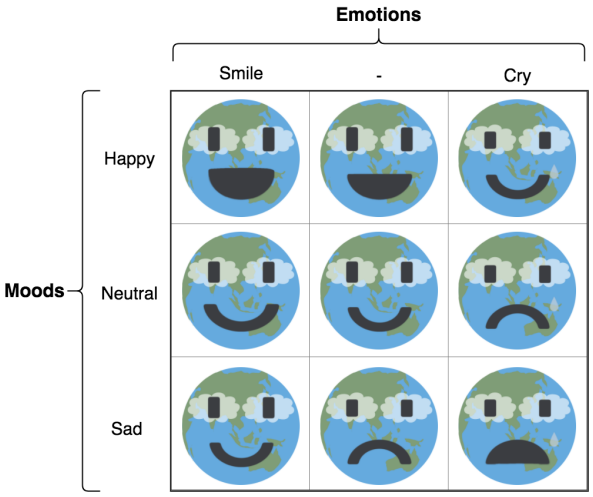

Figure 6.2: Earth states.

The work also designed more illustrations serving as frames to generate smooth animations that transit between the different Earth emotional states. As an example, Figure 6.3 depicts frames to generate the Earth's blinking animation in a certain mood.

Figure 6.3: Earth's blinking animation frames.

6.2.2 Application Integration

This project integrates the Earth SECA in the "Energy Madness" digital Cultural Probes (CP) application. The resulting prototype keeps gathering relevant data about energy knowledge, attitudes, and consumption habits of children and their families. However, the inclusion of the Earth as a new element of the interface, that will act as a companion virtual tutor of the user, enhances its possibilities.

The Earth appears in the following screens of the application[1]:

1. `Introduction Screen`. In the new version of the application, the Earth explains the CP and introduces itself (see Figure 6.4). In this spot, the user is not able to converse actively with it yet.

Figure 6.4: `Introduction Screen` screen-shots (left: initial version, right: new version).

2. `Registration Screen`. During the registration process, the Earth does some explanation (see Figure 6.5). However, the user is not able to converse actively with it yet.

Figure 6.5: `Registration Screen` screen-shots (left: initial version, right: new version).

[1]The detailed explanation of all the available screens of the application is in the Final Degree Thesis which focused on the implementation of the initial application [55]

3. `Dashboard Screen`. The Earth is always present in this spot so that the user can interact and converse with it by clicking on it (see top of Fig. 6.6). Moreover, the Earth informs the user when the game is over (see bottom of Fig. 6.6).

Figure 6.6: `Dashboard Screen` screen-shots (left: initial version, right: new version).

4. `Accept Mission Screen`. In the new version of the application, the Earth explains the missions (see Figure. 6.7).

Figure 6.7: `Accept Mission Screen` screen-shots (left: initial version, right: new version).

5. `Mission 1 Screen`. During the Psychologist's mission, the Earth sometimes shows up to ask children more detailed explanations of their answers.

6. `Mission 2 Screen`. In the Detective's mission, the Earth shows up to ask for more information about the green behaviour after finishing each member's task.

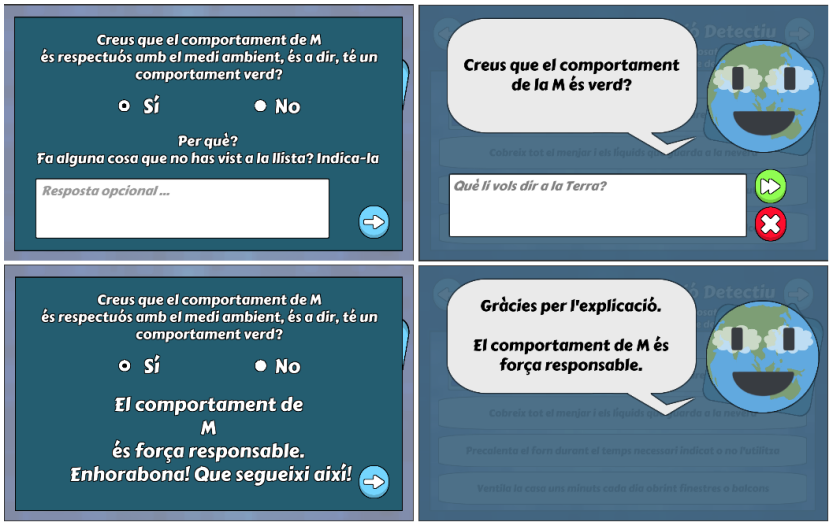

Figure 6.8: `Mission 2 Screen` screen-shots (left: initial application, right: new version).

7. `Mission 3 Screen`. During the house map setting in the Electrician's mission, the Earth shows up sometimes if many electrical appliances are in one room to make the user think about it.
8. `Mission 3 P2 Screen`. When user indicates an appliance use time in the Electrician's mission, the Earth shows up sometimes if the use time value is high.
9. `Select Day Screen`. When the Electrician's mission finishes the Earth shows up to make the user reflect about the different amounts of time the electrical appliances in the house are used on labour days and on weekend days.

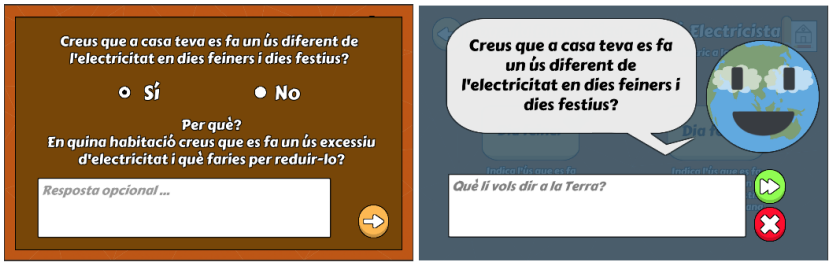

Figure 6.9: `Select Day Screen` screen-shots (left: initial version, right: new version).

10. All screens available after registration process. Whenever children shake their devices, the Earth appears and the they can converse with it in the same way as if they clicked it in the Dashboard screen.

Note that the interface that shows up when users converse with the Earth (see, for example, right side of Figures 6.8 or 6.9) allows:

- Appreciating personality changes of the Earth.
- Easily reading the Earth's messages.
- Easily inputting some text message for the Earth.
- Closing the interface any-time to stop talking with the Earth.

6.2.3 Earth SECA Modules Specification

This work designs different specifications to provide the Earth with conversational capabilities and elements to make it closer to the users. All the specified features will then be incorporated when the SECA library is used to initialize the Earth virtual tutor.

Conversational Module

The design this project proposes makes the Earth proactive in most cases and encourages children to reflect about energy related topics. The design of different Conversations ($Conv_i$, see Eq.3.2 in subsection 3.3.2) and Dialog Types (DT_j) using Finite State Machines contemplates all conversational experiences users may have when using the application.

Conversations (6 in total) and their DTs (13 different in total), are the following (`CA` –Conversational Agent– indicates proactive while `U` –User– refers to on demand):

1. `Dashboard` ($Conv_1$): Accessed from the Dashboard Screen or any-time users shake the device that executes the application.
   (a) `Information CA` (DT_1): Proactive DT i.e. the Earth chooses when to start it. Provides children with information related to energy efficiency and sustainability and shares advice.
   (b) `Meaning CA` (DT_2): Proactive DT. The Earth asks children if they know the meaning of an energy related concept and helps reviewing it.
   (c) `Concept CA` (DT_3): Proactive DT. The Earth asks children if there is any concept they do not know. If that's the case, it gives them the definition.
   (d) `CP CA` (DT_4): Proactive DT. The Earth asks children if there is any question they have about the CP. If that's the case, it gives them the answer.
   (e) `Free CA` (DT_5): Proactive DT. The Earth starts a topic free conversation, this means that it accesses AIML knowledge directly (without further input text analysis) to provide some text.
   (f) `Concept U` (DT_6): On demand DT version of DT_3. The user asks the Earth about the definition of an unknown concept.
   (g) `CP U` (DT_7): On demand DT version of DT_4. The user asks the Earth a question about the CP.
   (h) `Ask U` (DT_8): On demand DT. The user wants to obtain a question from the Earth. The diagram of this DT is equivalent to the Meaning CA one.

2. `Psychologist` ($Conv_2$): Starts occasionally after the user saves a combination of mood-appliance for a given family member in the Psychologist's mission.
   (a) `Combination CA` (DT_9): Proactive DT. The Earth asks the reason why the user chose a certain combination of mood-appliance.
   (b) `Concept U` (DT_6).
   (c) `CP U` (DT_7).
   (d) `Ask U` (DT_8).
3. `Detective` ($Conv_3$): Starts after the user saves the introduced information for a certain family member in the Detective's mission.
   (a) `Behaviour CA` (DT_{10}): Proactive DT. The Earth asks the child if he/she considers a family member's behaviour is correct and the reason why.
   (b) `Concept U` (DT_6).
   (c) `CP U` (DT_7).
   (d) `Ask U` (DT_8).
4. `Electrician Room` ($Conv_4$): Starts sometimes after the user saves a room with many electric appliances in first part of the Electrician's mission.
   (a) `Room CA` (DT_{11}): Proactive DT. The Earth asks the child why he/she thinks they have so many appliances in a certain room or what could they do to change that.
   (b) `Concept U` (DT_6).
   (c) `CP U` (DT_7).
   (d) `Ask U` (DT_8).
5. `Electrician Time` ($Conv_5$): Starts sometimes after the user saves an electric appliance with a long use time in the second part of the Electrician's mission.
   (a) `Time CA` (DT_{12}): Proactive DT. The Earth asks the child why he/she thinks they use a certain appliance so much or what could they do to change this.
   (b) `Concept U` (DT_6).
   (c) `CP U` (DT_7).
   (d) `Ask U` (DT_8).
6. `Electrician End` ($Conv_6$): Starts just before the Electrician's mission finishes.
   (a) `Use CA` (DT_{13}): Proactive DT. The Earth asks the child if he/she thinks the energy use of his/her family changes from labour days to weekend days and the reason why.
   (b) `Concept U` (DT_6).
   (c) `CP U` (DT_7).
   (d) `Ask U` (DT_8).

This project presents designs of Finite State Machine Diagrams for each Conversation and Dialog Type (all of them can be checked in Appendix D). As an example, Figure 6.10 illustrates the FSM Diagram for a Conversation ($Conv_3$) and a Dialog Type (DT_6) of the Earth. The format of the diagrams is the same as the one in Figure 3.5 though these diagrams also include example messages so that they can be more easily understood (notice that "Earth text" appears in green, while "user text" is purple).

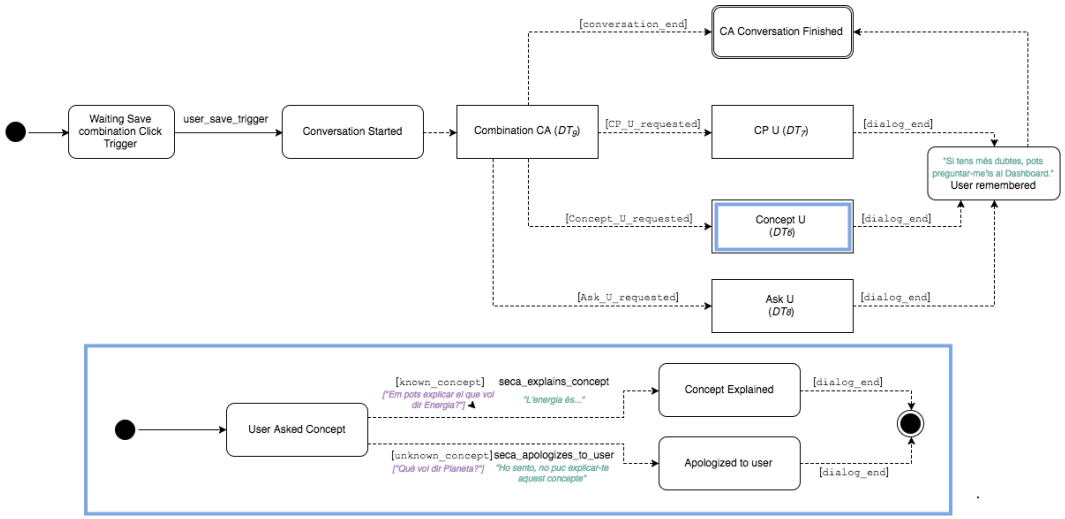

Figure 6.10: Conversation $Conv_3$ and Dialog Type DT_6 of the Earth.

Knowledge Module

Fluent and useful conversations and DTs require the Earth to have knowledge about vocabulary, practices related with energy and sustainability and the CP in the language of the application, which in this case, will be Catalan [31][27][50][26].

The following equations (see Subsection 3.3.3) briefly illustrate Earth's Knowledge Module contents:

$$KnowledgeModule = \langle extAIML, \{K_1, K_2, K_3, K_4, K_5\}\rangle \tag{6.1}$$

$$K_1 = \langle \{saving\_day, energy\_day, earth\_day, night\_use, ...\}, information\rangle \tag{6.2}$$

$$K_2 = \langle \{energy, energy\_source, fossil\_fuels, ..., kw, light, ...\}, concept\rangle \tag{6.3}$$

$$K_3 = \langle \{hydraulic\_energy, eolic\_energy, nuclear\_energy, ...\}, energy\_concept\rangle \tag{6.4}$$

$$K_4 = \langle \{detective, psychologist, electrician, journalist, missions, win\}, mission\rangle \tag{6.5}$$

$$K_5 = \langle \{fridge, oven, microwave, washer\_machine...\}, device\rangle \tag{6.6}$$

There is a total of 62 different concepts distributed in the 5 different topics.

Memory Module

In case of the Earth, Memory Module has as long-term memory all previously defined *DT*s and all just previously specified *concept*s in the Knowledge Module so that conversations do not become repetitive. It also has a short-term memory function as formalized in Subsection 3.3.4.

Personality Module

The Earth's personality this project proposes has the moods and emotions shown in Figure 6.3 so that it has an agreeable personality. Transition matrices values (see Tables 6.1, 6.2 and 6.3) make the Earth more focused on extreme emotions (happy/sad) than on neutral ones so that the impact on the user experience of the Earth's affective changes is more apparent.

$$Personality = \{agreeable\} \tag{6.7}$$

$$Moods = \{happy, neutral, sad\} \tag{6.8}$$

$$Emotions = \{smile, neutral, cry\} \tag{6.9}$$

Table 6.1: Earth's Mood Transition Matrix.

		newMood		
		happy	*neutral*	*sad*
currMood	*happy*	0.8	0.15	0.05
	neutral	0.25	0.5	0.25
	sad	0.05	0.15	0.8

Table 6.2: Earth's Emotion Transition Matrices.

happy

		newEmotion		
		smile	*neutral*	*cry*
curr Emot.	*smile*	0.8	0.15	0.05
	neutral	0.5	0.4	0.1
	cry	0.1	0.6	0.3

neutr.

		newEmotion		
		smile	*neutral*	*cry*
curr Emot.	*smile*	0.4	0.5	0.1
	neutral	0.3	0.4	0.3
	cry	0.1	0.5	0.4

sad

		newEmotion		
		smile	*neutral*	*cry*
curr Emot.	*smile*	0.1	0.6	0.3
	neutral	0.1	0.4	0.5
	cry	0.05	0.15	0.8

Table 6.3: Earth's meProbability Transition Matrix.

		Mood Impact		
		happy	*neutral*	*sad*
currentEmotion	*smile*	0.8	0.15	0.05
	neutral	0.25	0.5	0.25
	cry	0.05	0.15	0.8

Needs Module

Earth SECA has two different needs. The first one is the need for "Attention", as the necessity of social interaction that aims to increase user engagement. It becomes active when children are on the Dashboard Screen without interacting with the Earth for a while (see Figure 6.11). And the second one, "End Game", corresponds to the necessity of the agent to help accomplishing CP goals and makes appear the Earth in the Dashboard after some hours if the game is not finished to remind users about it.

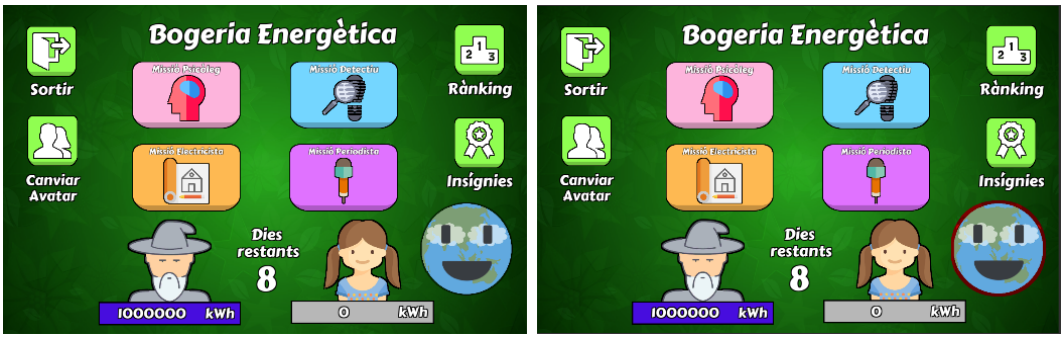

Figure 6.11: Earth's "Attention" need inactive (left) and active (right).

$$NeedsModule = \langle N_1, N_2 \rangle \tag{6.10}$$

$$N_1 = \langle attention, inactive, 8sec. \rangle, \quad N_2 = \langle end\_game, inactive, 7200sec. \rangle \tag{6.11}$$

Empathy Module

The Earth SECA has two empathy components to be more empathic with users in some way (see Eq. (6.12), (6.13) and (6.14)). On the one hand user's "motivation" controls the frequency of apparitions of the SECA during the application (it appears more often if users have a higher motivation value) and the Earth adjusts its value according to the durability of the conversational interactions. On the other hand, user's "tiredness" urges the Earth to end conversations by itself if they become too long. Though the initial $currentMax$ value for the last component is 4, it can change dynamically for each user if conversations finish earlier or later thanks to the detailed initialization values which work as explained in Subsection 3.3.7.

$$EmpathyModule = \langle E_1, E_2 \rangle \tag{6.12}$$

$$E_1 = \langle user\_motivation, 50, 0, 100, 100 \rangle \tag{6.13}$$

$$E_2 = \langle user\_tiredness, 0, 4, 4 \rangle \tag{6.14}$$

NLP Module

Regarding the NLP Module, when the server receives user input for the Earth, some classification needs to be performed in some cases to understand what users are saying or requesting. This thesis considers the following classification problems for the Earth:

- *Sentiment Analysis (SA)*. This classification becomes necessary in states of dialog types when the Earth needs to know if the user is making an affirmation, a negation or neither of them.

- *Explanation Detection (ED)*. This classification problem aims at determining if the user input contains an explanation or not. It is necessary for example in states of dialog types where users are expected to provide specific reasons (e.g., `Combination CA` or `Electrician Room`) or definition (e.g., `Meaning CA`).

- *Topic Matching (TM)*. This classification problem is essential to perform the designed on demand *DT* transitions. Given a user input, it predicts if the user is proactively requesting a change to `Concept U`, `CP U`, `Ask U` or neither of them.

- *Keyword Search CP (KSCP)*. When the user is supposed to be asking a question about the CP, this classification problem tries to guess what the question asked is about (for instance, about the Psychologist's mission or about winning the game).

- *Keyword Search Concept 1 (KSC1)*. This classification problem determines if the message has a concept related to energy or not. Consequently, states where supposedly the user is providing a concept to ask for its meaning, use it.

- *Keyword Search Concept 2 (KSC2)*. In case KSC1 decides that a concept is related to energy (see Eq. (6.4) and Table 6.5), KSC2 can be used afterwards to determine which concept is it.

- *Keyword Search Concept 3 (KSC3)*. In case KSC1 decides that a concept is not related to energy, KSC3 can be used afterwards to determine which concept is it. Figure 6.5 illustrates the different concepts that can classify.

To train the most appropriate model for each classification problem, this project performed the process detailed in Section 5.3. Tables 6.4 and 6.5 illustrate the prepared training data (real user input from a first test using AIML and simple keyword-pattern matching functions, see Fig. 1.1). For each each classification problem, the Tables illustrate the different class values for the multi-class classification problem, the number of available messages for each class, and the total number of messages used to train each classifier.

Table 6.4: SA, ED, TM and KSCP Classification Problems Data.

Classification Problem	Classes	Data (Relevant Text Messages)	Total Data
SA	Affirmation	515	2285
	Negation	424	
	Other	1346	
ED	Explanation	808	2285
	Other	1477	
TM	Ask	21	2285
	CP	78	
	Concept	117	
	Other	2069	
KSCP	Detective	15	78
	Electrician	10	
	Win	17	
	Missions	13	
	Journalist	10	
	Phycologist	13	

Table 6.5: KSC1, KSC2 and KSC3 Classification Problems Data.

Classification Problem	Classes	Data (Relevant Text Messages)	Total Data
KSC1	Energy related	73	117
	Other	44	
KSC2	Energy	21	73
	Electric energy	7	
	Eolic energy	4	
	Geothermal energy	2	
	Hydraulic energy	3	
	Seawater energy	3	
	Nuclear energy	4	
	Solar energy	11	
	Non-renewable energy source	6	
	Renewable energy source	10	
	Energy types	2	
KSC3	Climate change	3	44
	Fossil fuel	2	
	Contamination	1	
	Greenhouse effect	3	
	Environmental impact	1	
	Kw	2	
	Light	4	
	Environment	2	
	Standby	1	
	Earth	7	
	Unknown	18	
	Machine performance	0	
	Power of a machine	0	
	Sustainability	0	
	Green behaviour	0	
	Screensaver	0	
	Low consumption	0	
	Rubí Brilla	0	
	Energy efficiency	0	

Then, Grid Search with 10-Fold Cross Validation provided a huge number of results from the combination of the different parameters possibilities that Table 6.6 illustrates. After indicating the number of parameter combinations, Table 6.6 indicates the total number of tests performed for each ML algorithm, taking into account that the code performs:

- 10-Fold Cross Validation, i.e. the number of experiments is multiplied by 10.
- Experiments to tune three different scores (Accuracy, Precision and Recall), so that the number of experiments is multiplied by 3.
- Tests for the different classification problems (SA, TM, KSCP, KSC1, KSC2 and KSC3), so that the number of experiments is multiplied by 6. Note that for the SVM algorithms, the first row multiplies the result by 5 and the second row value stays as is it. The training separated the ED case since it requested special training parameters because of convergence problems.

All in all, this work performed a total of 26640 experiments.

Table 6.6: Grid Search with 10-Fold Cross Validation experiments performed.

ML Algorithm	Tested Parameters	Combination Tests Performed	Total Experiments
Support Vector Machine	kernel = ['rbf','linear'] C = [1, 10, 100, 1000] gamma = [[1e-2,1e-3,1e-4],None]	16	2.880
Support Vector Machine (ED)	kernel = ['rbf'] C = [1, 10, 100, 1000] gamma = [[1e-2,1e-3,1e-4]]	12	360
Random Forest	n_estimators = [10,20,50,100] criterion = ['gini','entropy']	8	1.440
Gradient Boosting	loss = ['deviance'] n_estimators = [10,50,100,200] learning_rate = [0.1,0.2]	8	1.440
Logistic Regression	penalty = ['l2'] C = [1,10,100] solver = ['newton-cg','lbfgs']	6	1.080
Multi-layer Perceptron	hidden_layer_sizes = [100,250,500] max_iter = iters = [1000] activation = ['identity','logistic','tanh','relu'] solver = ['lbfgs','sgd','adam'] alpha = [1e-2,1e-3,1e-4]	108	19.440

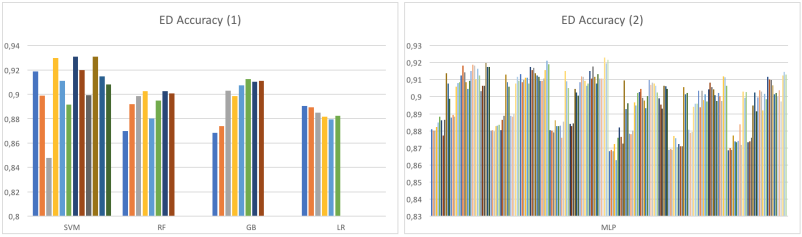

Figure 6.12: Example of some accuracy results obtained when performing Grid Search with 10-Fold Cross Validation experiments for the Explanation Detection (ED) classification problem. Each bar represents a combination of parameters for a certain method.

The results of the experiments (see Table 6.12) and the purpose of each classification problem led to choose the models indicated in Table 6.7 for the Earth's NLP Module. Considering in almost all cases the parameters combination that achieved a highest accuracy score and in the case of the Sentiment Analysis problem, the parameters which also achieved highest precision scores (more than one combination achieved a high accuracy).

Table 6.7: Chosen Machine Learning Model for each Classification Problem.

Classification Problem	Chosen Model	Parameters	Achieved Scores (Average)	Reason
SA	SVM	kernel = 'rbf' C = 10 gamma = 0.01	Accuracy = 0.9527 Precision = 0.95 Recall = 0.95	Highest accuracy and highest precision when choosing Affirmation and Negation
ED	SVM	kernel = 'rbf' C = 100 gamma = 0.01	Accuracy = 0.9308 Precision = 0.93 Recall = 0.93	Highest accuracy
TM	MLP	hidden_layer_sizes = 250 max_iter = iters = 1000 activation = 'relu' solver = 'adam' alpha = 0.001	Accuracy = 0.9540 Precision = 0.95 Recall = 0.95	Highest accuracy
KSCP	MLP	hidden_layer_sizes = 100 max_iter = iters = 1000 activation = 'tanh' solver = 'adam' alpha = 0.001	Accuracy = 0.8625 Precision = 0.86 Recall = 0.86	Highest accuracy
KSC1	SVM	kernel = 'rbf' C = 10 gamma = 0.001	Accuracy = 0.8705 Precision = 0.87 Recall = 0.87	Highest accuracy
KSC2	MLP	hidden_layer_sizes = 100 max_iter = iters = 1000 activation = 'identity' solver = 'lbfgs' alpha = 0.0001	Accuracy = 0.6857 Precision = 0.63 Recall = 0.68	Highest accuracy
KSC3	MLP	hidden_layer_sizes = 250 max_iter = iters = 1000 activation = 'relu' solver = 'sgd' alpha = 0.01	Accuracy = 0.6800 Precision = 0.65 Recall = 0.68	Highest accuracy

In addition to a ML Model (see Table 6.7) and a keyword-pattern matching function, this work chose the values in Table 6.8 to define each of the classification problems defined for the Earth SECA. Classification problems whose ML model has low accuracy (KSC2 and KSC3) have a True *keyword_matching_should_be_tested_before* Boolean. Regarding *further_classification*, Topic Matching has its value to True so that if ML detects no *DT* change, keyword-pattern matching functions also try to find one.

Table 6.8: Additional chosen parameters for each Classification Problem.

Classification Problem	*Threshold_value*	*further_classification* Boolean	*keyword_matching_should_be_tested_before* Boolean
SA	0.9	False	False
ED	0.9	False	False
TM	0.8	True	False
KSCP	0.8	False	False
KSC1	0.9	False	False
KSC2	0.7	False	True
KSC3	0.99	False	True

6.3 Implementation

6.3.1 Client-Server Architecture implementation

This project follows the client-server architecture presented in Section 3.2 to implement the Earth SECA and integrate it in the gamified CP application for Android devices implemented in Unity [55].

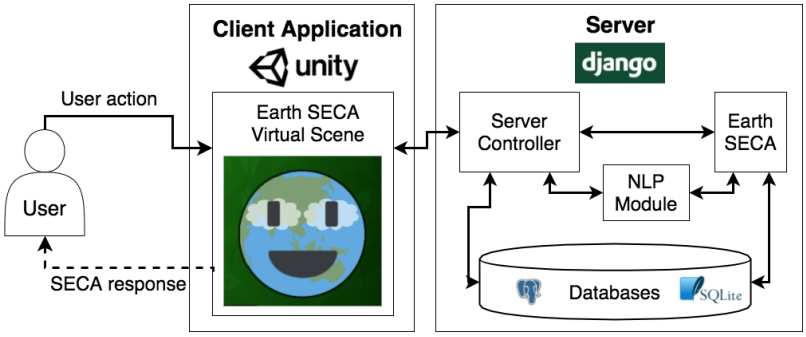

Figure 6.13: Earth SECA Architecture (logos copyrighted by Unity, Django, Postgres and SQLite).

The aforementioned server infrastructure uses Django Python Framework, while the client application uses Unity (see Figure 6.13). REST (Representational State Transfer) API, based on HTTP, is in charge of the server communications. Moreover, there are two Databases, one in PostgreSQL which stores CP application related data and another one using SQLite3 and Django models to store all the information related to the SECA. The server also requires different Python libraries like `scikit-learn`, `gensim`, `numpy` and `python-aiml` to work properly (see Appendix A.2.1).

Server details

The server implementation, using Django, presented in this project and available in the correspondent source code, has the file structure depicted in Figure 6.14. It follows a model-view-template architecture and besides the Earth SECA related code, it also has some specific server files. This project worked mostly with the following ones:

- `models.py`, which contains all the models of the SQLite3 database.
- `serializers.py`, containing the classes that serialize the models defined in `models.py`.
- `urls.py`, which defines the access points to the server functionalities (HTTP requests) that allow the communication between the Server and Client Application.
- `views.py` (Server Controller), which initializes the Earth SECAs and the NLP Module, and contains all the functions that the links defined in `urls.py` invoke.

The Django server runs on an Ubuntu server running Apache using WSGI (Web Server Gateway Interface). Though the University manages the server content, it is hosted by an external provider. The use of a Python Virtual Environment [17] in the Server allows creating an isolated environment within the server and facilitates easy replication in other servers. Appendix A contains requirements and instructions to maintain the server.

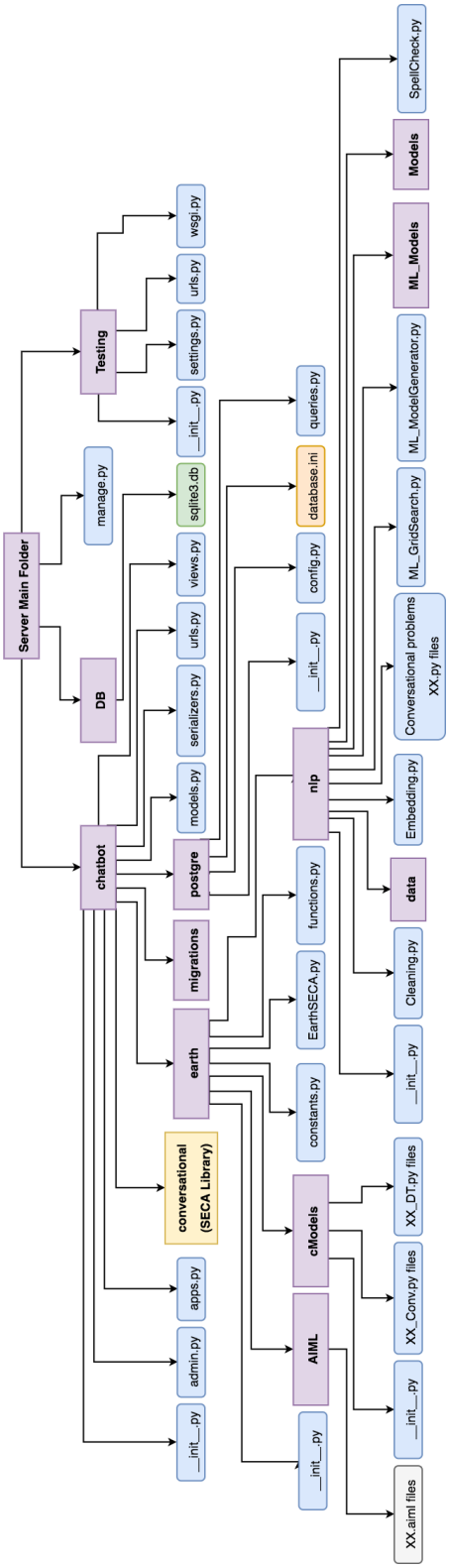

Figure 6.14: Django Server File Structure.

Client Application details

The Unity client application "Energy Madness" embeds the Earth SECA and is in charge of visualizing and synchronizing its different input and output modalities, i.e. dialog utterances, animations showing Earth's personality and needs, sound effects, and haptic feedback.

To do so, this project proposes a new Unity Prefab (*asset type that allows you to store a GameObject object complete with components and properties*), to facilitate the inclusion of the Earth in the different screens of the application. Figure 6.15 shows the game objects contained in the `EarthModule` prefab (in the same order so that firstly defined elements show at the back): `Blue Panel`, `EarthBig`, `BubbleE`, `EarthText`, `UserT`, `SendText`, `ExitXat`.

Figure 6.15: `EarthModule` prefab instantiation in an application Screen.

Apart from modifying almost all Screen related scripts to incorporate the Earth in them, this project also incorporates new scripts necessary for the correct integration of the Earth SECA in the application, and modifies others:

- `Earth.cs`, file with an Earth class containing enough methods to control the state of the Earth in the different screens (for example, to know if `EarthModule` prefab is visible or not).
- `GameControl.cs`, this file already existed though now incorporates the initialization of an `Earth` for each user and contains a new class `Coroutine` [25] to manage Server calls correctly avoiding data loss.
- `ServerControl.cs`, new source code that contains all necessary methods to stablish communication with the Server.
- `BDControl.cs`, file modified to contemplate the new Server connection.
- `HappySound.cs` and `SadSound.cs`, scripts that allow sound effects when "Smile" or "Cry" emotions happen.

Moreover, the project contains Unity animations for all different Earth personality states transitions.

6.3.2 Earth SECA

This project implements Earth SECA Controller as a class (`EarthSECA`) inheriting from the general SECA Controller class from the library explained in Section 3.3.

To do so, `EarthSECA` overwrites the following methods:

- `__init__` (SECA constructor), to initialize the SECA with all the necessary elements.
- `getAnswer`, which given some user input and certain parameters, updates the SECA and returns the Earth answer.
- All methods for loading and saving Django models data.

Earth SECA Constructor

The constructor method requires the agent identifier (`agentID`) and an instance of the NLP Module. When invoked it uses the constructor of the superclass to initialize the Personality Module and loads all the necessary components (conversations, knowledge, memories, empathy components and needs) to the different Modules.

The implementation of the Earth conversations takes into consideration the following:

- There is one class for each Conversation that inherits from the `ConversationalModel` class.
- The constructor of each Conversation initializes all its states (classes that inherit from the `State` class) indicating if they are final states and if they have a nested Dialog Type. Finally it indicates which one is the initial state. Each state class overwrites its `OnEntry` and `OnExit` methods.
  - `OnEntry` method returns `None` or list of keywords that `getAnswer` method uses to know the necessary internal processing. In case the state has a nested DT, it is updated.
  - `OnExit` method sets the following state, which may depend on some parameters obtained from the controller when calling the `update` method of the Conversational Module. In case the state has a nested DT, this method checks if the nested DT is completed or not. In case it is not completed the next state keeps being the same one and, in case it is completed, sets the next state depending on the finishing parameters obtained from the nested DT.

Earth SECA `getAnswer` method

Though the constructor is necessary, the `getAnswer` method of the Earth SECA ensures the correct performance of the agent. Given a message, a conversation identifier and some parameters that come from Unity with application data information, it updates the SECA and returns the answer message. This method, performs the following steps:

1. Updates the Conversational Module and receives a list of keys (K1) from the `OnEntry` method of the current state.
2. Processes the obtained list of keys (from index 0 and so on) and performs some actions depending on the value of the key that is being processed:

(a) `AIML`: stores the keyword after this one as a provisional pattern, that might be used later by the Knowledge module, in a new list (K2).

(b) `ANALYSE_INPUT`: analyses user input in a certain way determined by the next keywords in the K1 list to understand what users are saying. Updates the Conversational Module afterwards with the analysis result as parameter.

(c) `NEXT_STATE`: updates again the Conversational Module to the next state.

(d) `END_PARAM`: sends a special signal to Unity indicating that the conversation with the Earth must end.

(e) `RANDOM_DT_NEEDED`: chooses the following state with nested *DT* somewhat randomly after taking Memory Module information into consideration to avoid being repetitive.

3. Processes the stored provisional patterns in K2 and modifies them depending on the current context (conversation, memory and knowledge) to get the definitive AIML keys (K3).
4. Calls SECA's update method with the definitive AIML keys to update the Earth's new personality values and get the message that will send to Unity.
5. If necessary, also sends to Unity a special trigger to close the Earth.

When this method analyses user input (`ANALYSE_INPUT`), all the NLP flow explained in Section 4 takes place. Though all user input undergoes preprocessing in the same way, depending on the current state, different classification problems may be solved. The classification results lead to an identifier that will be send as parameter when the Conversational Module is updated after the analysis to assign the conversation next state (see identifiers in the FSM diagrams of Appendix D).

6.3.3 Data Persistence and Databases

To store the data gathered in the CP, a whole PostgreSQL database structure was already designed in the Final Degree Thesis that implemented the first prototype of the application [55]. However, this database was in a poor performance Server and the connection, apart from being slow, sometimes led to data loss during transmission. That is why, this project has migrated the whole database model to the server that manages the University to improve data gathering and internet connections.

To do so, this project implements some methods in the Python Django Framework to communicate with the PostgreSQL database and to update and modify it when necessary (see `Postgre` folder in Figure 6.14).

Regarding the Earth's data, this project has designed a whole new database using Django models saved in an SQLite3 database. In addition to the general models that facilitate SECAs personality, empathy and memory persistence (see Fig. 3.2) and the ones that facilitate SECA interaction related data gathering (see Fig. 3.3), there are other models to simplify the data sent in certain connections (since serialized models data can be sent through the HTTP requests) and application related data gathering (see Fig. 6.16).

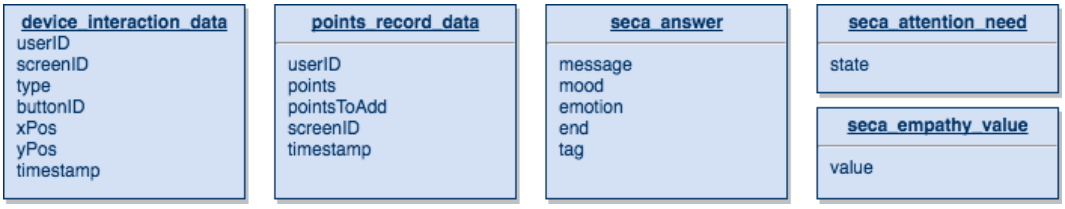

Figure 6.16: Models to facilitate Unity connection and store application related data.

- `device_interaction_data`, allows storing all the screen touches and shakes users make when using the application.
- `points_record_data`, allows storing all application point changes to detect abnormal behaviours.
- `seca_answer`, is serialized to send data to Unity after receiving and processing some user input.
- `seca_attention_need`, is serialized and used in Unity to check the state (active or not) of the Earth's "Attention" need.
- `seca_empathy_value`, is serialized and used in Unity to check the value of a certain Empathy Component.

As mentioned in Section 6.3.2, to save and load personality status, memory and empathy data of the Earth's using Django models, `EarthSECA` class overwrites the correspondent methods from the SECA Controller superclass.

Finally, highlight that there are some methods in the `views.py` Python Django Framework file that facilitate the download of the necessary data from the different databases in an easily readable format (CSV). Appendices A and B, specify all the commands to manage and use the different databases.

6.3.4 Other Client Application improvements

Apart from including the Earth SECA in the different screens of the gamified CP application and of migrating the whole PostgreSQL database, this project performed other improvements on the initial Unity application.

1. It simplified the text provided in the introduction of the CP by making the Earth being the one explaining the story so that it is more engaging for the users. Moreover, now text uses `TextMesh Pro` from Unity's assets store [15], to show icons instead of words in certain cases making the text easier to read and the story behind it more understandable (see Figure 6.17).
2. In the original application, when there was a problem with the database (usually when internet connection was lost), the game shut down. Now, it checks the device Internet connection every time a connection with the server needs to be performed. In case there is no connection, the user receives a message (see Figure 6.18). When connection is recovered, the application retries the transmission until it is correctly performed.

Figure 6.17: `TextMesh Pro` use in the Introduction Screen.

Figure 6.18: Check Internet connection message.

3. In the `Changing Avatar Screen`, there was a problem that did not allow users to select a new avatar after changing it for the first time. This happened in some devices because in some cases the avatar identifier was too long and not correctly saved. Reducing the length of the identifiers saved in memory solved this problem.
4. If the user left the `Ranking Screen` before the scores where loaded, in the initial application, the game crashed. Apart from solving this problem, the current application shows a message to users to let them know that scores may take some time to load (see Figure 6.19).

Figure 6.19: Loading Ranking Screen.

5. In the `Mission 3 Screen` of the Electrician Mission, when objects were being put inside the different rooms, if an object from the left list tried to be removed, the application crashed. Problem solved by taking away the possibility of removing objects from the left list
6. In the initial application prototype, when a text referred users themselves, instead of using the second person pronoun and the correspondent verbal forms, everything was said in third person. Earth SECA fixed this.

6.4 Testing

6.4.1 Methodology

This project tested with users two different prototypes of the gamified CP application with the embedded Earth SECA. The first one had an Earth using AIML and simple keyword-pattern functions while the second one incorporates an agent with the more complex NLP Module integrating ML presented in this thesis (see Table 6.9).

Table 6.9: Summary of the different evaluation tests.

Test	Goals	Participants
Pilot Test	Define questionnaires Fix small problems Check Server concurrency	4
Test of the app with an Earth SECA using keyword-pattern matching techniques	Evaluate the Earth SECA Gather user input data to train ML algorithms	30
Test of the app with an Earth SECA with a more complex NLP Module integrating ML	Evaluate the Earth SECA Check if the NLP Module makes an improvement	15

Children who participated did it voluntarily and their parents signed a consent form where they were informed about the Cultural Probes, data anonymity and the use of data only for research purposes (see Appendix C.1). The evaluation followed standards and ethical guidelines.

After getting the signed consent form, the evaluation consisted on the following steps:

1. Explaining the CP application to the volunteer children.
2. Ensuring that all volunteer children answered a questionnaire before using the application (*pre-test questionnaire*). The design of this questionnaire allows learning about their previous experiences in conversational interactions (see Appendix C.2).
3. Letting children use the application during a period of maximum 8 days at their own pace at home.
4. Ensuring that all volunteer children answered another questionnaire after using the application *post-test questionnaire* to know their impressions on the Earth SECA (see Appendix C.3).

A pilot test with a reduced number of users (4) tested a first prototype of the application before the first evaluation. The goals were:

- Defining the final versions of the questionnaires that children had to answer.
- Finding possible mistakes in the application.
- Verifying that the simultaneous access to the Server where the Earth was deployed supposed no problem.

After defining the final versions of the questionnaires and confirming that the whole system worked correctly, it started the first big round of testing with 30 volunteer children from two schools, who were between 11 and 12 years old and spoke Catalan. The goals of this first test were:

- Gathering natural language user data to train the ML algorithms that then would be used in the following prototype version with the enhanced NLP Module.
- Assessing the impact of the SECA agent on user experience (perception of learning, understanding, etc).

Finally, this work evaluated the new version of the Earth SECA having an enhanced NLP Module, with 15 other children between 10-12 years old from different schools and that also spoke Catalan. The main goals of this final evaluation were:

- Obtaining feedback and user data to verify the efficacy of the Earth SECA.
- Comparing the results obtained with the previous Earth and the new one including the enhanced NLP Module integrating ML techniques.

6.4.2 Gathered Data

All the previously explained tests, facilitated gathering either qualitative and quantitative data [45]. Though the gamified CP application itself collects information related to the energy habits families have, the evaluation of the application has already been performed in a previous Final Degree Thesis [55]. That is why, though this project also gathered CP related data when children used the application, this thesis focuses on the obtained data useful to evaluate the SECA architecture and the impact it had on the application.

On the one hand, this project performed gathering of *quantitative data*, which is the one gathered in numerical form and that can be ordered and measured:

- During the use of the application (for example, the number of interactions).
- Thanks to some questions from the pre-test and post-test questionnaires whose answers are convertible to a Likert scale from 1 to 5.

On the other hand, gathering of *qualitative data*, which is the one that cannot be converted to numbers, was possible thanks to different open questions from the two questionnaires and thanks to the content of the interactions they had with the Earth.

Regarding the data gathered during the use of the application, the Server source code provided in this project allows generating different CSV files (more information in Appendix B.3) with the following data:

- Postgres database data comprising different CSV files with the user information, and all the energy related information they sent during their use of the application.
- Device Interaction data, which stores all the clicks of the user when using the application, with a time-stamp and indicating if a determinate button was clicked.
- SECA Interaction data, storing all interactions between users and the Earth including information like the exchanged messages or the mood and emotion of the Earth.

- Training data, useful for ML purposes containing user messages, when they were received (dialog type and state from the FSM) and their assigned tag.
- Classification data, to check in detail all classifications performed by the NLP Module.
- Points Record data, to analyse how points are updated in DB and detect the reason behind some possible atypical behaviour.

As for the questionnaires, the one to be answered before using the application (see Appendix C.2), apart from demographic data, wants to know if the users have previous experience with the Earth SECA and if they seem predisposed to talk to one or not. The questionnaire to be answered after using the application (see Appendix C.3) again, apart from demographic data, wants to know how the users evaluate different aspects of the experience they had with the SECA (if they enjoyed it, if they learnt, if they preferred it in certain moments, etc.).

The following chapter presents and analyses the Evaluation results.

Chapter 7

Evaluation

As previously stated in Section 1.2, the main goals of this project's evaluation are:

1. Assessing the Conversational User eXperience (CUX) while evaluating the efficacy of the proposed SECA architecture.
2. Determining whether the Earth SECA presence impacted on the quantity of data gathered in the Cultural Probes.

The results obtained with both versions of the application (see Fig. 1.1) will be compared during the whole analysis to see if the NLP Module presence and some new adjustments had an overall positive effect.

- v1 - Simple NLP (SECA with AIML and keyword-pattern matching functions).
- v2 - Enhanced NLP (SECA with NLP Module using AIML, keyword-pattern matching functions and Machine Learning).

Demographic data and previous experience with Conversational Agents data

Children from the two evaluation tests performed have the demographic distributions that Figure 7.1 depicts:

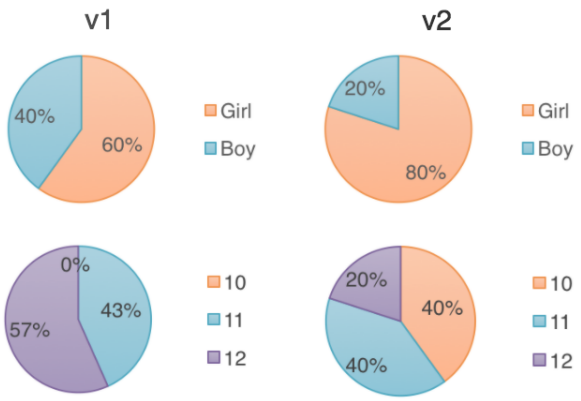

Figure 7.1: Test participants demographic data.

The pre-test questionnaire shows how most children had talked with a conversational agent before. However, there were also some who did not know what a chatbot is (see Figure 7.2).

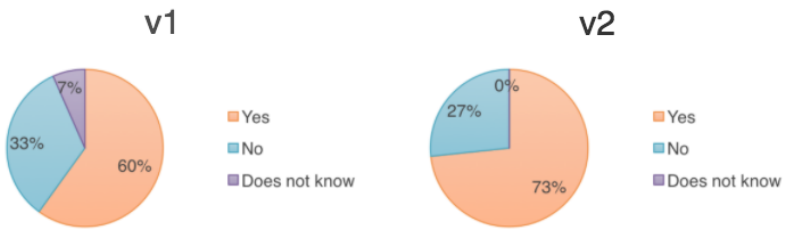

Figure 7.2: "Q1. Have you ever talked with a chatbot, i.e. a virtual character with whom you can converse?" answers.

Children who answered yes, had mostly talked with conversational agents like Siri or Google's assistant. And the majority considered that their experience was positive (see Figure 7.3).

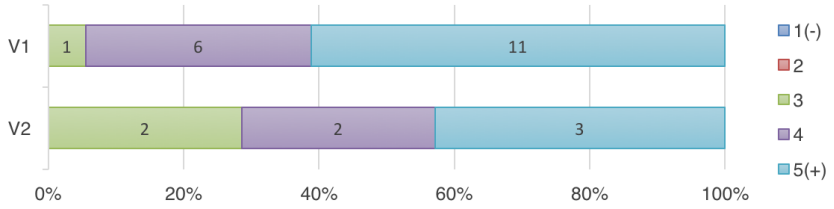

Figure 7.3: "Q2. How was the experience of talking with the chatbot?" answers (being 1 the most negative and 5 the most positive).

Most of the percentage of children who had not talked with a chat-bot or did not know what it was, would like to talk to one (see Figure 7.4) because they think it could be fun, interesting or because it is a new experience.

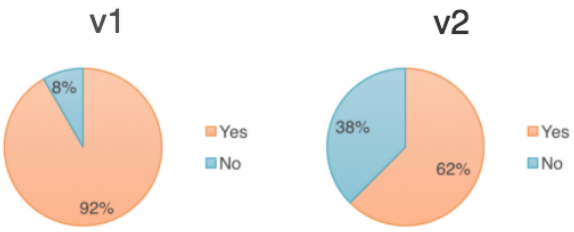

Figure 7.4: "Q3. Would you like to talk with a chat-bot?" answers.

Since children from different genders, ages and with various previous experience with conversational agents participated, the results might be more general and less biased.

Conversational User eXperience (CUX) data

The post-test questionnaire children answered after using the application provides a lot of valuable data to know what they thought about the Earth SECA and at the same time to analyse if the proposed architecture is appropriate, for example, if the agent understands what users are saying, if users detect the emotions it shows or if agent needs make any difference.

On the one hand, the perception children have about the *overall experience* of talking with the Earth, shows that it was clearly satisfactory (see Figure 7.5). Moreover, the percentage of positive answers seems to increase in the version using the enhanced NLP Module (v2) with respect to v1.

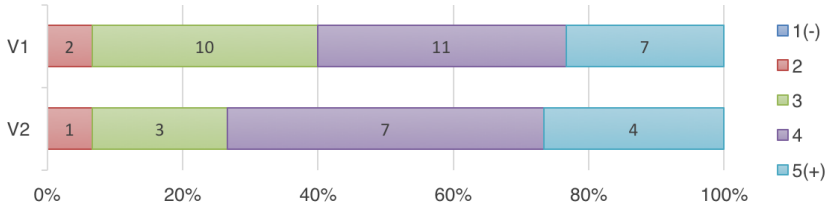

Figure 7.5: "Q1. What was your overall impression when talking with the Earth?" answers (being 1 the most negative and 5 the most positive).

Children who did not enjoy the experience argue, for example, that "the Earth did not understand" them or that "it was too insistent", while the ones who enjoyed it consider that "it explained new information" or "was interesting".

The other questions of the post-test questionnaire evaluate how user's perceived each of the different SECA Modules.

Q2 and Q3 analyse the impact the Earth SECA *Personality Module* had on users (see Figure 7.6).

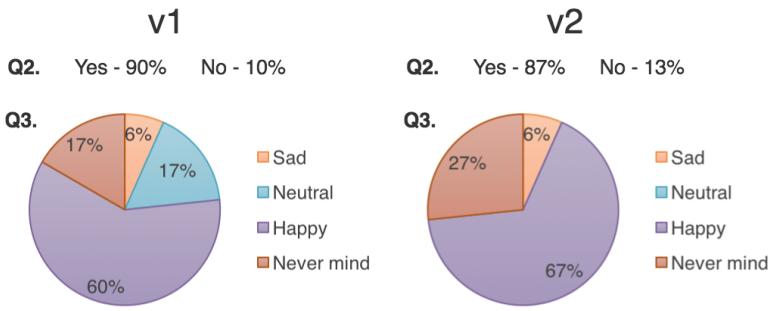

Figure 7.6: "Q2. Did you noticed that the Earth showed emotions (happiness and sadness)?" and "Q3. When did you prefer to talk with the Earth?" answers.

From Q2 it can be deduced that almost all children noticed that the Earth showed emotions. Moreover, they clearly preferred to talk with the agent when it was in a certain mood. Though most of them preferred to talk with the Earth when it was "Happy" because they felt that what they were saying was fine or they "felt better", it is not applicable to everyone. There are children, for example, who preferred to talk to it when it was sad, though they also explained that the reason was to "make the Earth happy".

Consequently, a possible future work would be making changes in the Earth's Personality Module transitions matrices to make more probable the transitions to certain moods or even to adapt the transition matrices to each user since every one has different preferences.

Q4 evaluates if children considered that they had learnt thanks to all the concepts introduced in the Earth's *Knowledge Module* and the effectiveness of its *Memory Module* to avoid repetition. Furthermore, this question is useful to determine the accomplishment of one of the goals of including the Earth in the CP application, which was enhancing users' experience by making the application more educative. Thanks to the provided energy related concept definitions and ideas to reduce energy consumption most users perceived they learned (see Figure 7.7).

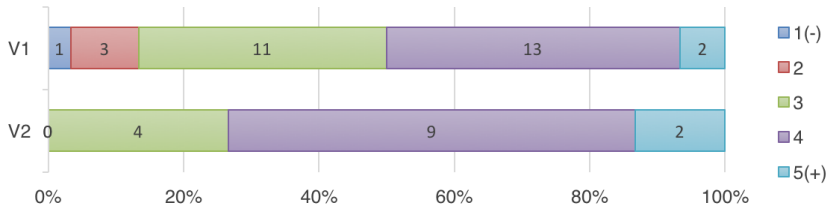

Figure 7.7: "Q4. Do you think you have learned thanks to the Earth?" answers (being 1 the less and 5 the most).

To evaluate the SECA's *Conversational Module* and *NLP Module*, there is a lot of data. Q5 illustrates how there is a clear difference in the perception user's had of the Earth understanding in each version.

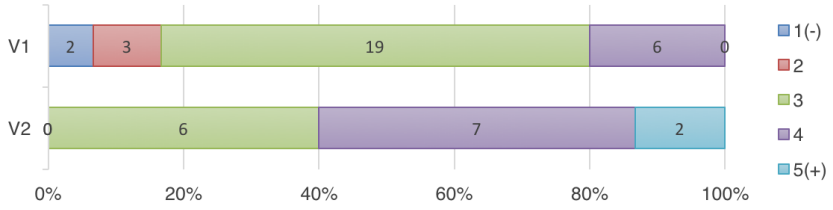

Figure 7.8: "Q5. Do you think the Earth understood what you were saying?"answers (being 1 the less and 5 the most).

Regarding v1, where the Earth SECA just used AIML and basic keyword-pattern matching functions, 16.67% of children complained about the Earth misunderstanding with negative scores of 1 and 2 and no one considered that the agent completely understood everything. This result is corroborated by a 22.31% of Response Error Rate (see Table 7.1), since 505 out of 2263 agent's responses were not correct. This result proves that simple keyword-pattern matching techniques are inadequate to provide the agent with enough understanding.

On the other hand, when children answered about the agent with the enhanced NLP Module incorporating ML (v2), their perception clearly seemed to improve. No user considered misunderstanding was bad enough to give scores of 1 and 2 and 13% of children even considered that the Earth always understood them. In this case, objective data (see Table 7.1) also reflects the improvement with almost half a RER of 11.46% (80 out of 698 agent responses were not correct).

Table 7.1: Earth interactions related data.

Data	V1 (30 participants)	V2 (15 participants)
Total number of messages	6010	1986
Number of user messages	2263	698
Average number of answers (standard deviation)	75.43 ($\sigma = 40.70$)	46.53 ($\sigma = 20.02$)
Response Error Rate (RER)	22,31%	11,46%
Number of Earth messages	3747	1288
Number of started conversations ($Conv_i$)	1484	590
Number of started dialogs (DT_j)	2165	744
Total number of times users called the Earth	591	180
Total number of transitions to on demand Dialog Types	58	13

All in all, Table 7.1 provides overall data on user-SECA interactions, which resulted in a total of 7996 utterances. From these, the high number of user utterances is remarkable, although the large standard deviation of its average shows an imbalanced engagement of kids in the conversational experience. Analysing post-test questionnaires of those users who uttered less than 50 messages and found that 72,73% of them considered that the Earth scarcely understood them. Indeed, they literally said "Sometimes I asked a question and the Earth did not know how to answer it". As for the number of conversations, they are quite similar to the number of dialogs, which means conversations were rather short. Several factors may explain this similarity: the design of conversations (5 out 6 conversations had only one proactive *DT* and three on-demand *DT*s); agent's emphatic behaviour detected user's tiredness and ended conversations early; and the level of engagement of some kids.

Regarding the Earth misunderstanding, this work considered two types of incorrect answers:

- *NLP Mistake*, which are the erroneous answers the SECA provides that could be corrected with an improvement of the techniques used in the NLP Module. For example, if the user asks for the definition of "solar energy" and the Earth provides the definition of "energy".
- *Unexpected or unintelligible message*, which refers to answers that could not be provided correctly because there was no AIML, the text was unintelligible or the structured conversation or dialog type did not contemplate that option. For example, if users ask non-expected free topic questions like "What is your favourite colour?" or ask about non expected energy related questions like "What uses more energy, the fridge or the computer?".

Taking that into consideration, the percentage of mistakes coming from NLP also seem to have improved slightly (see Table 7.2). However, there is still room for improvement.

Table 7.2: Earth incorrect answers analysis.

Type of Mistake	**Number of Messages (Proportion)**	
	V1	**V2**
NLP Mistake	245(48.51%)	37(46.25%)
Unexpected or unintelligible message	260 (51.49%)	43 (53.75%)
Total Mistakes (TM)	505	80
RER (TM/Total messages)	22.31%	11.46%

Table 7.3 illustrates the results of all the different classifications that took place during the second testing round using the new NLP Module. Algorithms with a higher number of available training data -"SA", "ED" and "TM"- (see Tables 6.4 and 6.5), perform better than the others. However, the results for KSCP and KSC3 show that they might have worked better if Keyword matching had been performed more often (Predominant Method column illustrates the predominant chosen method of classification in each case i.e. ML –Machine Learning– or Keywords –keyword-patten matching–). KSC1 and KSC2 prove how, when ML models does not work as expected, is good to take keyword-pattern matching into consideration by adjusting the *Threshold_value* and the *further_classification* and *keyword_matching_should_be_tested_before* Boolean values of each classification problem (see Section 4.3).

Table 7.3: NLP Module with Keyword matching and Machine Learning integration results.

Classification Problem	**Analysed Messages**	**Accuracy**	**Predominant Method (% used times)**
SA	282	96.81%	ML (91.13%)
ED	494	93.93%	ML (71.46%)
TM	712	97.75%	ML (97.47%)
KSCP	47	44.68%	ML (68.09%)
KSC1	59	100%	Keywords (96.61%)
KSC2	4	100%	Keywords (100%)
KSC3	46	65.22%	ML (65.22%)

Q6 (see Fig.7.9) of the post-test questionnaire evaluates the efficacy and proves the necessity of the *Empathy Module*.

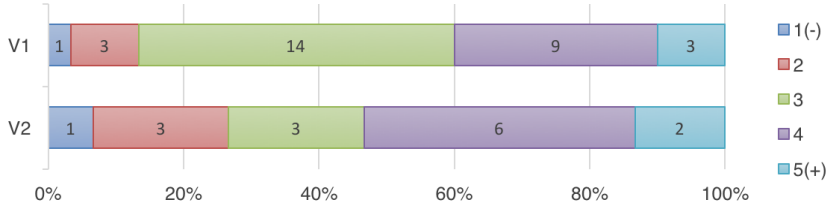

Figure 7.9: "Q6. Did you like that the Earth appeared sometimes to talk to you?" answers (being 1 the less and 5 the most).

Though Table 7.1 shows that children required the attention of the Earth plenty of times (either by clicking it in the Dashboard screen or by shaking their device), the results of the post-test questionnaire that Figure 7.9 illustrates, indicate that not all children enjoyed the fact that the Earth appeared during their use of the application. Consequently, it makes sense to adjust the times the Earth appears depending on the child (some of them really enjoyed it while others did not like it), proving the necessity of the Empathy Module and its further adjustment.

Regarding the *Needs Module*, in the first and second test, children clicked the Earth 178 and 46 times respectively while it showed its "Attention" need. This results that the 30.12% (v1) and the 25.56% (v2) of times users called the Earth, might be because of the "Attention" need effect. This might have had an impact in the learning experience of the children, since they had access to more knowledge the more they talked with the Earth.

SECA agent impact on the quantity of data gathered in the Cultural Probes

Table 7.4: Open questions user response comparison.

Application Version	Number of open questions asked	Number of open questions NOT answered	Response rate
Without the Earth	110	28	74.55%
With the Earth SECA	200	18	91.00%

Finally, data obtained in a previous version of the application, which did not embed a conversational agent [55], allows evaluating whether the Earth SECA affected the quantity of data gathered in the CP application. The response ratio of open questions asked to the user, without agent and with the Earth agent is 74.55% and 91.00%, respectively.

This difference proves to be significant after performing a chi-squared test that results in a $p - value = 0.000096 < 0.05$. This indicates that the agent had a clear positive effect on the active participation of children in the CP.

CP completion rate has stayed the same with respect to the previous version of the application without Earth SECA. In that case, 20 of the 25 participants completed it (80%), while 36 out of 45 participants (80%) of the CP with the Earth have finished all the application tasks. This means that the "End Game" need is necessary and might require some adjustments so that children become more motivated to finish the application.

Chapter 8

Conclusions and Future Work

This project introduces Sentient Embodied Conversational Agents (SECAs) as virtual characters capable of engaging users in complex structured conversations and also incorporate some human-like sentient qualities like being able to perceive user's feelings and response to them. This work also formalizes their architecture and provides a software library that facilitates their inclusion in applications requiring proactive and sensitive agent behaviours therefore, fulfilling one of the main goals of the project.

SECA library implementation, apart from including Conversational, Knowledge and Memory modules to communicate with users, disposes of the Personality, Needs and Empathy modules to make the agent sentient and increase the bound with them.

The implemented SECA library also includes a Natural Language Processing Module devoted to enhance the user input analysis.

This thesis designs, implements and embeds a virtual tutor for a gamified Cultural Probes application for children using the proposed SECA architecture (the Earth). The first evaluation of the SECA library, using just simple keyword-pattern matching techniques to understand users' input, served to detect its strong and weak points. At the same time it permitted gathering real user data to train some Machine Learning algorithms to be used in the final version of the SECA Library with an enhanced NLP Module.

In order to validate empirically the proposal, the machine learning enhanced version of the Earth SECA was also tested. Evaluation results show children's overall satisfaction. Moreover, user experience was also enhanced successfully since most children consider they learned while interacting with the Earth consequently achieving the goal of making the application more educative.

Specifically, results corroborate that endowing the agents with human-like features such as personality, needs, or empathy increases user bonding because, for example, they enjoyed talking with the Earth knowing that it showed emotions and called it when it showed its "Attention" need. Moreover, NLP Module has successfully reduced Response Error Rate from 22.31% to 11.46% and open questions answer rate has also increased.

In fact, there is also a significant improvement of the quantity of user data gathered by our CP when compared with a previous version lacking of the Earth SECA, therefore achieving another goal of this project.

Apart from achieving the main goals of the project, it also allowed the achievement of different personal goals like increasing my knowledge related to Conversational Agents, Machine Learning and Natural Language Processing.

Finally, it is worth mentioning that the Sentient Embodied Conversational Agent architecture designed, implemented and evaluated in this project has been accepted as a paper for the 21st International Conference of the Catalan Association of Artificial Intelligence (CCIA 2018) [58].

8.1 Future work

Though the architecture seems to work well, its modular design leaves room for improvement and as future work there is still chance to enhance many of its modules for the Earth specific case:

- New conversations and dialog types could be added to increase the presence of the Earth in the CP application.
- Knowledge module could be enhanced with more topics because, for example, there were children who would have like to talk with the Earth about energy related jokes.
- Memory Module could incorporate new techniques so that the agent can remember and refer to previous interactions on user demand. There were children who, for instance, did not remember what the Earth had said and requested repetition.
- Personality Module could tie moods and emotions to user actions and not only text so that for example, when CP missions are completed, the Earth becomes happier.
- Empathy components values could be adjusted so that the Earth sudden apparitions frequency is more adequate for each user.
- More needs could be modelled.

Regarding NLP, it presents many difficulties and, though the new system reduced Response Error Rate, there is still room for improvement. There are other methods that can be applied like Deep Learning, which seems to be giving good results. However, though a Recurrent Neural Network tried to be used at first for the spell-checking step, it was discarded because the method used in the correct project provided better results.

The results obtained when evaluating the Machine Learning models also show that the more data used for training, the better the results. So future versions of the Earth should take all gathered data into consideration to train again the algorithms. This process could also improve by testing more parameters when using Grid Search with Cross Validation.

Bibliography

[1] Backpropagation algorithm. http://ufldl.stanford.edu/wiki/index.php/Backpropagation_Algorithm. Accessed: 2018-06-23.

[2] cawiki dump progress on 20180401. https://dumps.wikimedia.org/cawiki/20180401/. Accessed: 2018-06-25.

[3] Decision trees. http://scikit-learn.org/stable/modules/tree.html#tree. Accessed: 2018-06-23.

[4] Duolingo bots. http://bots.duolingo.com. Accessed: 2018-05-10.

[5] Ensemble methods. http://scikit-learn.org/stable/modules/ensemble.html. Accessed: 2018-06-23.

[6] Generalized linear models. http://scikit-learn.org/stable/modules/linear_model.html#logistic-regression. Accessed: 2018-06-23.

[7] Gradient boosting classifier. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html#sklearn.ensemble.GradientBoostingClassifier. Accessed: 2018-06-23.

[8] Logistic regression. http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. Accessed: 2018-06-23.

[9] Mlp classifier. http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html. Accessed: 2018-06-23.

[10] Neural network models (supervised). http://scikit-learn.org/stable/modules/neural_networks_supervised.html. Accessed: 2018-06-23.

[11] Python advanced: Finite state machine in python. https://www.python-course.eu/finite_state_machine.php. Accessed: 2018-06-24.

[12] Random forest classifier. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier. Accessed: 2018-06-23.

[13] Spell checker using word2vec. https://www.kaggle.com/cpmpml/spell-checker-using-word2vec. Accessed: 2018-06-25.

[14] Support vector classification. http://scikit-learn.org/stable/modules/svm.html. Accessed: 2018-06-23.

[15] Textmesh pro. https://assetstore.unity.com/packages/essentials/beta-projects/textmesh-pro-84126. Accessed: 2018-06-22.

[16] Tuning the hyper-parameters of an estimator. http://scikit-learn.org/stable/modules/grid_search.html. Accessed: 2018-06-20.

[17] Virtualenv – virtualenv 16.0.0 documentation. https://virtualenv.pypa.io/en/stable/. Accessed: 2018-06-26.

[18] Wiki word2vec. https://github.com/hgrif/wiki-word2vec. Accessed: 2018-06-25.

[19] Wit.ai. https://wit.ai. Accessed: 2018-05-10.

[20] P. Almajano, D. Tellols, I. Rodríguez, and M. López-Sánchez. Meto: A motivated and emotional task-oriented 3d agent. In *Recent Advances in Artificial Intelligence Research and Development*, volume 300, pages 263–268. IOS Press, 2017.

[21] P. Angara, M. Jiménez, K. Agarwal, and H. Jain et al. Foodie fooderson a conversational agent for the smart kitchen. In *Conference on Computer Science and Software Engineering*, pages 247–253, 2017.

[22] C. Becker, S. Kopp, and I. Wachsmuth. *Simulating the Emotion Dynamics of a Multimodal Conversational Agent*, pages 154–165. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[23] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[24] J. Cassell. Embodied conversational agents: representation and intelligence in user interfaces. *AI magazine*, 22(4):67, 2001.

[25] J. Dunstan. Too many corroutines: A queue solution. https://jacksondunstan.com/articles/3241. Accessed: 2018-06-23.

[26] Endesa. Consejos de ahorro de energía en el hogar.

[27] E. Esteban. 10 consejos para enseñar a tus hijos a ahorrar energía. https://www.guiainfantil.com/blog/educacion/valores/10-consejos-para-ensenar-a-tus-hijos-a-ahorrar-energia/. Accessed: 2018-06-26.

[28] N. D. Feshbach and S. Feshbach. Empathy and education. *The social neuroscience of empathy*, 85:98, 2009.

[29] Interaction Design Foundation. User centered design. https://www.interaction-design.org/literature/topics/user-centered-design. Accessed: 2018-06-20.

[30] Prashant G. Cross-validation in machine learning. https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f. Accessed: 2018-06-20.

[31] A. García-Carmona and A. M Criado. Enseñanza de la energía en la etapa 6-12 años: un planteamiento desde el ámbito curricular de las máquinas. *Enseñanza de las Ciencias*, 31(3):0087–102, 2013.

[32] B. Gaver, T. Dunne, and E. Pacenti. Design: Cultural probes. *interactions*, 6(1):21–29, January 1999.

[33] P. Gebhard. Alma: A layered model of affect. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '05, pages 29–36, New York, NY, USA, 2005. ACM.

[34] Google. Dialogflow. https://dialogflow.com. Accessed: 2018-05-10.

[35] A. C Graesser, S. Lu, G. T. Jackson, H. H. Mitchell, M. Ventura, A. Olney, and M. M Louwerse. Autotutor: A tutor with dialogue in natural language. *Behavior Research Methods, Instruments, & Computers*, 36(2):180–192, 2004.

[36] M. A. Hearst, S. T Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[37] R. Higashinaka, K. Imamura, T. Meguro, C. Miyazaki, N. Kobayashi, H. Sugiyama, T. Hirano, T. Makino, and Y. Matsuo. Towards an open-domain conversational system fully based on natural language processing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 928–939, 2014.

[38] D. Hume. Emotions and moods.

[39] IBM. Watson. https://www.ibm.com/watson/. Accessed: 2018-05-10.

[40] Renuka J. Accuracy, precision, recall & f1 score: Interpretation of performance measures. http://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/. Accessed: 2018-06-20.

[41] S. Kshirsagar. A multilayer personality model. In *Proceedings of the 2nd international symposium on Smart graphics*, pages 107–115. ACM, 2002.

[42] Tomas M., Kai C., Greg C., and Jeffrey D. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[43] A. H Maslow. A theory of human motivation. *Psychological review*, 50(4):370, 1943.

[44] Robert R. McCrae and Oliver P. John. An introduction to the five-factor model and its applications. *Journal of personality*, 60(2):175–215, 1992.

[45] S. McLeod. Qualitative vs quantitative data. https://www.simplypsychology.org/qualitative-quantitative.html, 2008. Accessed: 2018-06-22.

[46] Microsoft. Luis. https://www.luis.ai/home. Accessed: 2018-05-10.

[47] Microsoft. Rinna. https://www.rinna.jp. Accessed: 2018-06-20.

[48] P. Norvig. How to write a spelling corrector. *http://norvig.com/spell-correct.html*, 2007.

[49] A. Ortony, G. L Clore, and A. Collins. *The cognitive structure of emotions*. Cambridge university press, 1990.

[50] Ser Padres. Cómo ahorrar energía en casa.

[51] I. Poggi, C. Pelachaud, F. de Rosis, V. Carofiglio, and B. De Carolis. *Greta. A Believable Embodied Conversational Agent*, pages 3–25. Springer Netherlands, Dordrecht, 2005.

[52] G. Rabionet Martínez. Conversational web with meto agent library: Implementation and evaluation. 2018.

[53] J. Rickel and W. Lewis Johnson. Steve (video session): A pedagogical agent for virtual reality. In *Proceedings of the Second International Conference on Autonomous Agents*, AGENTS '98, pages 332–333, New York, NY, USA, 1998. ACM.

[54] K. Samsó, I. Rodríguez, A. Puig, D. Tellols, F. Escribano, and S. Alloza. From cultural probes tasks to gamified virtual energy missions. In *Proc. British Computer Society HCI Conf.*, page 79, 2017.

[55] K. Samsó Muñoz. Design, implementation and evaluation of a gamified digital cultural probe in the context of energy consumption. 2017.

[56] S. Serholt, W. Barendregt, T. Ribeiro, G. Castellano, A. Paiva, A. Kappas, R. Aylett, and F. Nabais. Emote: Embodied-perceptive tutors for empathy-based learning in a game environment. In *European Conference on Games Based Learning*, page 790. Academic Conferences International Limited, 2013.

[57] I. Steinwart and A. Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

[58] D. Tellols, M. López-Sánchez, I. Rodríguez, and P. Almajano. Sentient embodied conversational agents: Architecture and evaluation. *"To appear"*, 2018.

[59] D. Tellols, K. Samsó, I. Rodríguez, and A. Puig. Cultural probes for the gamification of energy awareness. ACM WomENcourage, 2016. https://womencourage.acm.org/archive/2016/poster_abstracts/womENcourage_2016_paper_5.pdf, Accessed: 2018-06-26.

[60] R. S. Wallace. A.l.i.c.e. http://alice.pandorabots.com. Accessed: 2018-05-10.

Appendices

Appendix A

Developer Manual

A.1 Working Environment

Hardware used during the whole development process:

- iMac (21.5', 2010)
  - macOS Sierra 10.12.6
  - Processor: 3,2 GHz Intel Core i3
  - Memory: 4GB 1333MHz DDR3
  - Graphics: ATI Radeon HD 5670 512 MB
- MacBook Pro (Retina 13', 2015)
  - macOS Sierra 10.12.6
  - Processor: 3,1 GHz Intel Core i7
  - Memory: 16GB 1867MHz DDR3
  - Graphics: Intel Iris Graphics 6100 1536MB

Device used for testing the CP application:

- Samsung Galaxy Tab A6 (10.1')
  - Processor: 1.6GHz Octa Core Processor
  - Memory: 2GB RAM

As for the software, this project required the use of:

- "draw.io" for the design of the different diagrams.
- Different text editors to code:
  - "MonoDevelop-Unity", version 5.9.6 (to code in C#).
  - "TextWangler", version 5.5.2 (to code in Python and to edit other text files).
  - "Wing IDE" 1015.1.12-1 (to test Python code).

- "Unity", version 2017.1.1f1 Personal (to integrate the virtual tutor in the application).
- Different tools to work with the Python Server where the SECA is deployed:
  - "Django".
  - "Virtual Environment".
  - "FileZilla", version 3.31.0 (to connect with the server, upload and download data).
  - "Postman", version 5.5.3 (to send requests to test the server).
  - "Terminal", version 2.7.3 (to test and work with the server).

Some online platforms to store the code and other data (diagrams, questionnaires, etc.) in the cloud were used:

- "BitBucket".
- "Dropbox".
- "GoogleDrive".

A.2 Django Server Management

A.2.1 Server deployment

During development, the server can run on a local network:

```
cd LOCAL_DIR/Earth_Server_and_SECA_Library
cd ./manage.py runserver
```

However, the Django project source files need to run in a server so that all application users can access them.

For the testing, this project deployed its source code in a server managed by the University using Apache. This server is accessible via Terminal using an `ssh` protocol and the appropriate credentials.

The points to take into consideration when deploying the application in a server are:

1. It is recommended to create an specific virtual environment with the requirements to run the Django project. These are:

```
Django==1.11.10
Unidecode==1.0.22
argparse==1.2.1
boto==2.48.0
boto3==1.7.33
botocore==1.10.33
bz2file==0.98
certifi==2018.4.16
chardet==3.0.4
configparser==3.5.0
distribute==0.7.3
djangorestframework==3.7.7
docutils==0.14
futures==3.2.0
gensim==3.4.0
idna==2.6
jmespath==0.9.3
numpy==1.14.2
psycopg2==2.7.4
python-aiml==0.9.1
python-dateutil==2.7.3
pytz==2018.3
requests==2.18.4
s3transfer==0.1.13
setuptools-scm==2.1.0
six==1.11.0
smart-open==1.5.7
unicodecsv==0.14.1
urllib3==1.22
wsgiref==0.1.2
```

Important: Though the previous requirements can be installed by putting them in a text file and executing the `pip freeze` command in console, "scipy==1.0.1" and "scikit-learn==0.18.1" are also required though they cannot be installed using the same method. To complete their installation, install both modules outside the virtual environment and then manually copy-paste all the installation folders inside the appropriate directory of the v. environment (`VIRTUAL_ENV_DIR/lib/python2.7/site-packages/PACKAGE_NAME`).

2. Apache needs a new configuration file in "sites-enables" to contemplate the new application.

Further steps to follow every time a new version of the project wants to be deployed in a server:

3. Before uploading the project, generate locally the `sqlite3` database file and all the `.csv` data files (it does not matter if they are empty at this time).
4. Upload the project source code to the server using, for example, “FileZilla”.
5. Update directories in the `/testing/wsgi.py"` file of the project.
6. Change permissions for `sqlite3` database and all `.csv` files using the following command when being in the appropriate directory:

```
chmod 777 FILE_NAME
```

7. Access the virtual environment being in the appropriate directory and executing:

```
source VIRTUAL_ENV_DIR/bin/activate
```

8. Make migrations and migrate the server in case models changed:

```
./manage.py makemigrations
./manage.py migrate
./manage.py makemigrations chatbot
./manage.py migrate chatbot
```

9. Restart the apache server:

```
sudo /etc/init.d/apache2 restart
```

10. Exit the virtual environment using `deactivate`.

A.2.2 Server maintenance instructions

- To run the server, go to its main directory and execute the command:

```
./manage.py runserver IP:PORT
```

- To use a new server host, open the `/testing/settings.py` file and add its IP to the `ALLOWED_HOSTS` list available inside.
- If the project location in the host server changes, open the `/testing/wsgi.py` file and update directories.
- If a new folder wants to be added to the project, it must include an "__init__.py" file (can be a copy of an existing one).
- To create a new connection link that invokes a function from the server through a POST or GET method (HTTP request):
  - Specify the function inside a class of the `chatbot/views.py` file.

```
class CAView(viewsets.ModelViewSet):
    @detail_route(method=['post'])
    def function(self, request):
        ...
        return Response()
```

  - Add the url to the `urlpatterns = []` variable in the `chatbot/urls.py` file.

```
urlpatterns = [...,
        url(r'^new_url$', views.CAView.as_view({'post':'function'}),
            name='function'),
    ...]
```

- To add a new model to the Django sqlite3 database:
  - Define it in the `chatbot/models.py` file.
  - Define its serializer in the `/chatbot/serializers.py` file.
  - Register it in the `/chatbot/admin.py` file.
  - From the server console, execute:

```
./manage.py makemigrations
./manage.py migrate
./manage.py makemigrations chatbot
./manage.py migrate chatbot
```

- If Postgres database location changes, new configuration must be specified in the `chatbot/postgre/database.ini` file:

```
[postgresql]
host=NEW_HOST_IP
database=DATABASE_NAME
user=DATABASE_USER
password=DATABASE_PASSWORD
```

A.3 Unity Application Project Management

When working with the Unity project source code of the gamified Cultural Probes application, the following indications might be useful:

- Server must be running when using the application. When the Server component is deployed on a local or private network, the appropriate address must be indicated in the `Awake` method of the `BDControl` class of the `BDControl.cs` file and the `ServerControl` class of the `ServerControl.cs` file:

```
...
initializeURLs("http://IP:PORT/chatbot/");
...
```

- In the `Awake` method of the `GameControl` class from the `GameControl.cs` source file, there is a set of lines that delete the local saved data file. These lines should be commented when the application wants to be deployed though it might be useful to have them when testing code.

```
...
if (File.Exists (Application.persistentDataPath + "/gameInfo.dat")) {
    Debug.Log ("deleting");
    File.Delete (Application.persistentDataPath + "/gameInfo.dat");
}
...
```

- To incorporate the Earth in a certain screen, apart from adding the correspondent prefab to the Scene, the attached source code must include some essential lines of code (any Scene including the Earth's source file serves as reference):
  - Attributes to manage all the Earth components.
  - Attributes to manage device shake detection mechanism.
  - Initialization lines in the `Start` or `Awake` method (depends on the Scene).
  - Code to check Earth "Attention" need and device shaking in the `Update` method.
  - `toBigEarth` and `closeBigEarth` methods to "open" and "close" the Earth interaction screen (might be slightly different depending on the screen).
  - `updateBubble(int access)` to send user input to the server to then update the Earth's bubble with the answer message received.
  - `waitAndClose()` method that closes the Earth automatically after some seconds when called.
- When performing computer testing, device shaking cannot be tested and in the Introduction Screen, when the Earth is explaining the story, device shaking is requested to continue. To solve this, there are some lines in the `Update` method of the `IntroductionScreen` class of the "IntroductionScreen.cs" file that skip this step and facilitate computer testing. Those lines must be commented to deploy the application.

```
...
bubble11E.GetComponentInChildren<TextMeshProUGUI> ().text =
                                  "Molt bé! Molta sort en l'operació!";
nextB.SetActive (true);
needCheckS = false;
currentBubble = 13;
...
```

Appendix B

User Manual

B.1 CP Application installation instructions

The gamified Cultural Probes application "Bogeria Energètica" ("Energy Madness") can be installed in devices with the following specifications:

- Android operative system with version above 4.1 (JellyBean).

To install the application, follow these steps:

1. Download the `.apk` file of the application.
2. If a message saying "Blocked installation" pops up:
   (a) Click "Configuration".
   (b) Look for the option "Unknown sources" and enable it.
   (c) Click "Accept".
3. Click "Install".

If the installation is successful, it will appear in the "Applications" panel of the device.

B.2 CP participants registration instructions

To control user access to the CP, participants need to be previously registered to the Postgres database to access the application.

Since there is a ranking and different groups of users might be required to use the application the following POST instruction (that can be sent using "Postman") facilitates the registration of different user groups:

```
http://IP:PORT/chatbot/db_add_mail
Body
    email: user_name@domain
    name: USER_GROUP Complete_Name
```

Where `USER_GROUP` must be the same for all users appearing in the same ranking.

B.3 Server data download instructions

To download all databases data in CSV format from the server, the following steps can be followed:

1. Using "Postman", send the following GET requests to the server:
   (a) Postgres Data

   `http://IP:PORT/chatbot/get_p_data`

   (b) CA Interaction Data

   `http://IP:PORT/chatbot/get_cai_data`

   (c) Device Interaction Data

   `http://IP:PORT/chatbot/get_di_data`

   (d) Training Data

   `http://IP:PORT/chatbot/get_t_data`

   (e) Classification Data

   `http://IP:PORT/chatbot/get_c_data`

   (f) Points Record Data

   `http://IP:PORT/chatbot/get_pr_data`

2. Using "FileZilla", connect to the server and download all the generated .CSV files that will be inside the `DB` folder of the project.

Appendix C

Evaluation Material

C.1 Consent Form

Projecte: Cultural Probes en el context de l'energia
FULL INFORMATIU PELS PARES/MARES/TUTORS/ES

Introducció

- En aquest estudi es demana al seu fill/a de participar voluntàriament en un estudi d'investigació. Ha estat seleccionat com a possible participant perquè té una edat d'entre 10 i 12 anys, té mòbil o tableta pròpia amb sistema operatiu Android.
- Li demanem que llegeixi aquest document i faci qualsevol pregunta [1] que pugui tenir abans de permetre que el seu fill/a participi en aquest estudi.

Objectius

- Aquest estudi té com a objectiu conèixer millor les necessitats que les famílies tenen en relació al consum i gestió de l'energia elèctrica. Per aconseguir aquest objectiu es farà servir un mètode anomenat "Cultural Probes", que consisteix en donar unes activitats als nens i nenes per a fer voluntàriament a casa durant un període limitat de temps.
- La informació obtinguda en aquest temps servirà per al disseny d'aplicacions informàtiques que ajudin a les famílies a gestionar el seu consum elèctric i canviar o adaptar els seus hàbits relacionats amb l'energia.

[1] Per qualsevol dubte sobre l'estudi el pares poden contactar per correu amb els investigadors del projecte: inmarodriguez@ub.edu, maite_lopez@ub.edu, almajano@ub.edu. Departament de Matemàtiques i Informàtica de la Universitat de Barcelona.

Procediment

- El seu fill/a tindrà instal·lada una aplicació al seu dispositiu (mòbil o tableta amb sistema operatiu Android) durant vuit dies. Després d'aquest període, l'aplicació quedarà inactiva, llista per a ésser eliminada del dispositiu.
- Aquesta aplicació demana fer un seguit d'activitats que requereixen la interacció del seu/seva fill/a amb pares/mares i germans/nes. L'aplicació també incorpora un agent conversacional amb qui el vostre fill/a pot parlar sobre temes relacionats amb l'energia i les activitats pròpies de la Cultural Probes. Aquestes activitats es fan voluntàriament quan es tinguin uns minuts d'esbarjo al llarg del dia.

Confidencialitat

- Tota la informació serà tractada de forma anònima, cap informació serà retinguda ni utilitzada amb altres finalitats que aquelles d'establir conclusions de recerca i compartir-les amb la comunitat educativa, famílies i altres investigadors.

Beneficis

- La participació en aquest estudi pot fer que la seva família sigui més conscient dels bons i mals hàbits de consum elèctric.

Consentiment

- La seva firma a sota indica que vostè ha decidit permetre al seu fill/a participar en aquest estudi, i que vostè ha entès la informació proporcionada abans.

(Pare/Mare/Tutor/a del/la menor)

Nom i cognoms: Dni:

Signatura: Data:

Investigadors de l'estudi:
Dra. Inmaculada Rodríguez Santiago
Dra. Maite López-Sanchez
Dr. Pablo Almajano
Departament Matemàtiques i Informàtica
Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

C.2 Previous Questionnaire

(Questions marked with * are mandatory)

1. Email address (the one you will use to register in the application)*
2. Gender*
   (a) Girl
   (b) Boy
3. Age*
   (a) 10
   (b) 11
   (c) 12
   (d) Other
4. Do you know if yours and your family's energy related behaviour is respectful with the environment?*
   (a) No
   (b) Yes
5. Are you aware of your household electricity usage?*
   (a) I am not aware
   (b) I am a little bit aware
   (c) I am very aware
6. Q1. Have you ever talked with a chatbot, i.e., a virtual character with whom you can converse?*
   (a) No
   (b) I don't know
   (c) Yes

In case Q1. Answer is yes:

7. With what chatbot did you talk to and/or for what?*
8. Q2. How was the experience of talking with the chatbot?*
   (a) I hated it
   (b) I didn't like it
   (c) I liked it a little
   (d) I liked it
   (e) I liked it a lot
9. Why?

In case Q1. Answer is no or I don't know:

10. Q3. Would you like to talk with a chatbot?*
    (a) No
    (b) Yes
11. Why?

C.3 Final Questionnaire

(Questions marked with * are mandatory)

1. Email address (the one you used to register in the application)*
2. Gender*
   (a) Girl
   (b) Boy
3. Age*
   (a) 10
   (b) 11
   (c) 12
   (d) Other
4. Q1. What was your overall impression when talking with the Earth?*
   (a) I hated it
   (b) I didn't like it
   (c) I liked it a little
   (d) I liked it
   (e) I liked it a lot
5. Why?
6. Q2. Did you notices that the Earth showed emotions (happiness and sadness)?*
   (a) No
   (b) Yes
7. Q3. When did you prefer to talk with the Earth?*
   (a) When it was SAD
   (b) When it was NEUTRAL
   (c) When it was HAPPY
   (d) Never mind
8. Why?
9. Q4. Do you think you have learned thanks to the Earth?*
   (a) Nothing
   (b) No
   (c) A little
   (d) Yes
   (e) Yes, a lot

10. Q5. Do you think the Earth understood what you were saying?*
    (a) Never
    (b) Almost never
    (c) Sometimes
    (d) Almost always
    (e) Always
11. Why?
12. Q6. Did you like that the Earth appeared sometimes to talk to you?*
    (a) Not at all
    (b) No
    (c) Sometimes
    (d) Yes
    (e) Yes, a lot
13. Why?
14. Q7. Is there any topic would you like to talk about with the Earth?*
15. Q8. Do you have any additional comment about the Earth (what you liked the most, the less...)?
16. Do you know if yours and your family's energy related behaviour is respectful with the environment?*
    (a) No
    (b) Yes
17. Are you aware of your household electricity usage?*
    (a) I am not aware
    (b) I am a little bit aware
    (c) I am very aware
18. Do you think the habits of your family will change after using the "Energy Madness" application?*
    (a) They won't change
    (b) They will change a little bit
    (c) They will change a lot
19. Do you have any additional comment about the application (what you like the most, the less...)?

Appendix D

FSM Diagrams within the Conversational Module

In the following pages there are all the designed Finite State Machine (FSM) diagrams representing each of the Conversations (*Conv*) and Dialog Types (*DT*s) designed for the Earth SECA.

The style followed to design all of them is the one used in Figure 3.5 including additional example messages so that they can be more easily understood ("Earth text" appears in green, while "user text" is purple), as specified in Section 6.2.3.

D.1 Dashboard ($Conv_1$) FSM Diagram

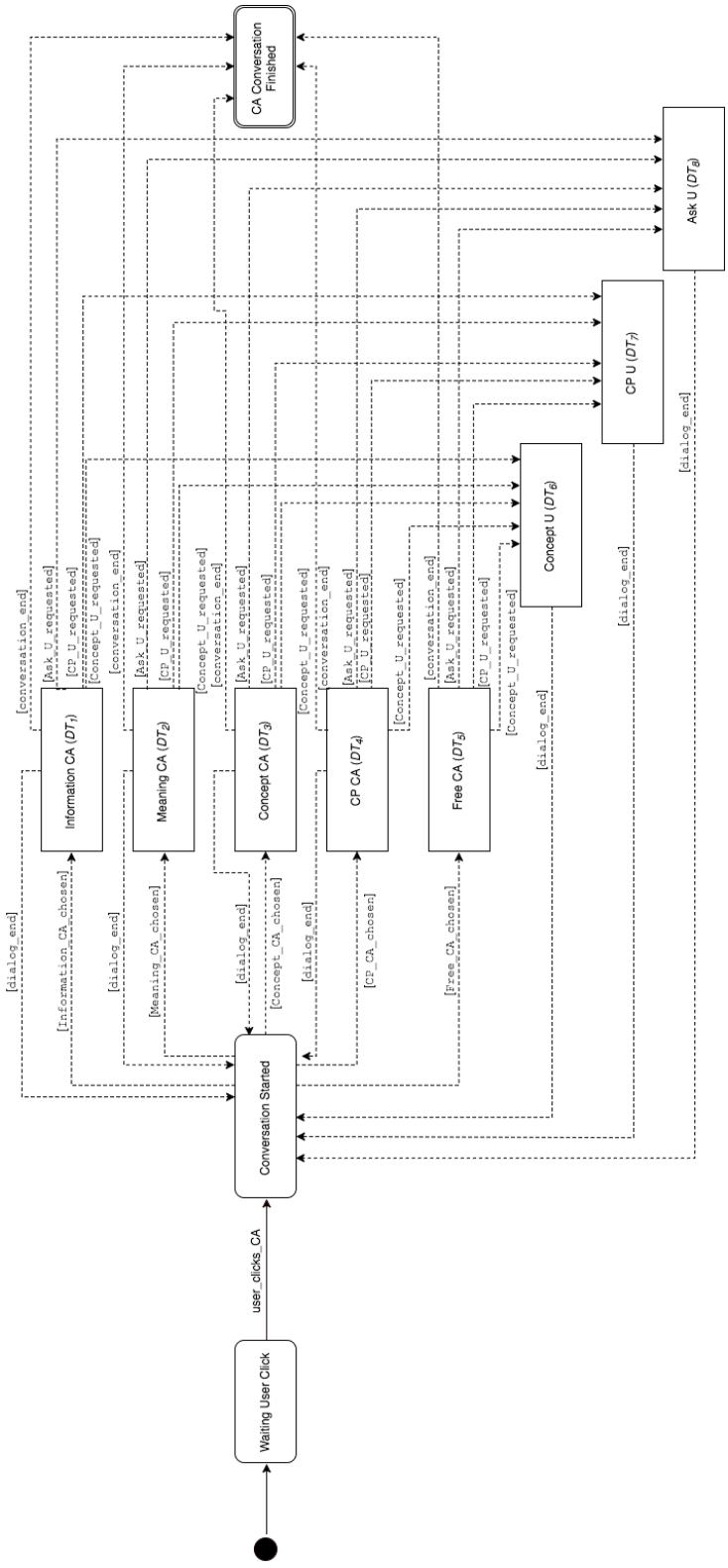

Figure D.1: Dashboard ($Conv_1$) FSM Diagram.

D.2 Psychologist ($Conv_2$) FSM Diagram

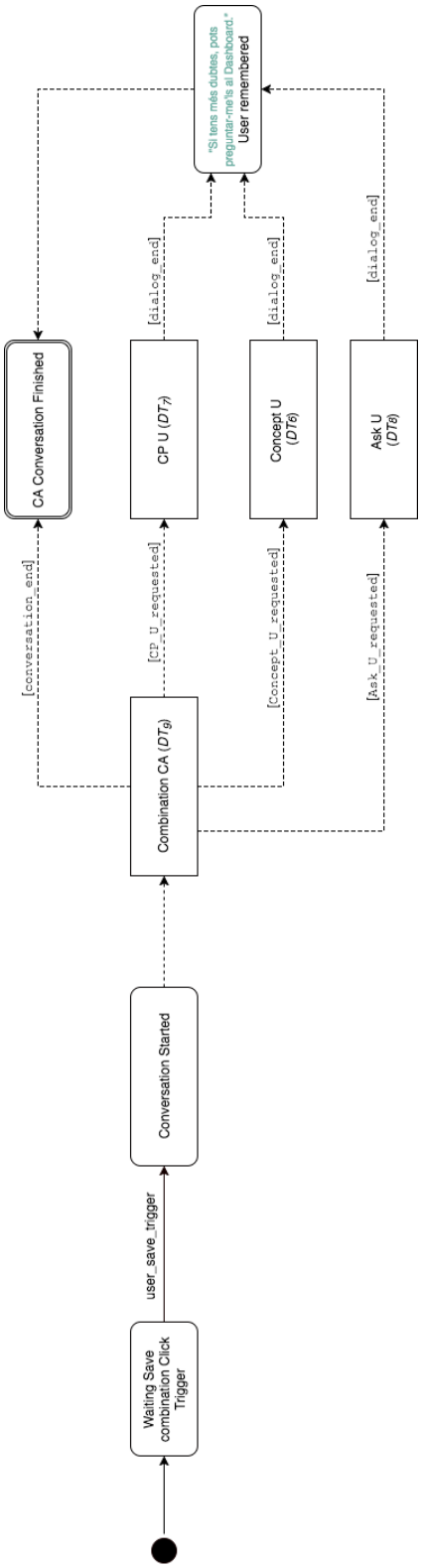

Figure D.2: Psychologist ($Conv_2$) FSM Diagram.

D.3 Detective ($Conv_3$) FSM Diagram

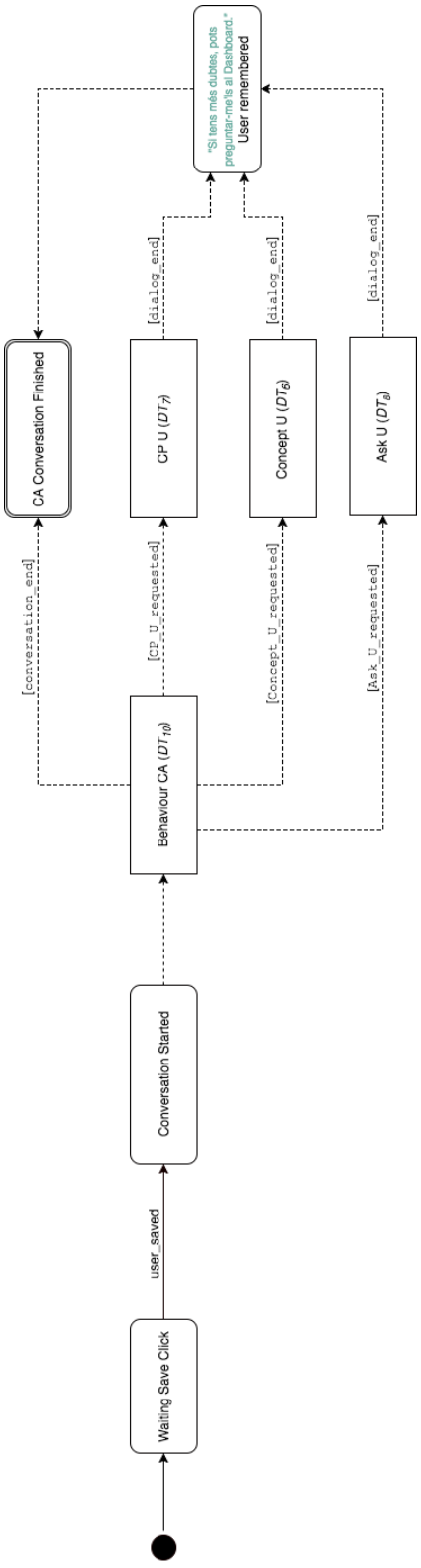

Figure D.3: Detective ($Conv_3$) FSM Diagram.

D.4 Electrician Room ($Conv_4$) FSM Diagram

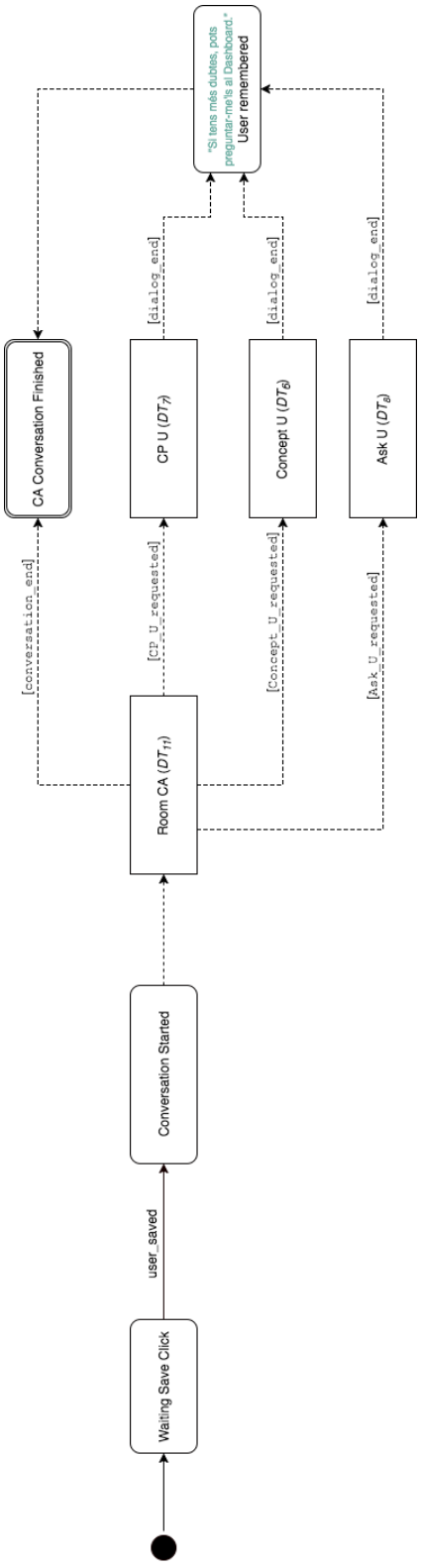

Figure D.4: Electrician Room ($Conv_4$) FSM Diagram.

D.5 Electrician Time ($Conv_5$) FSM Diagram

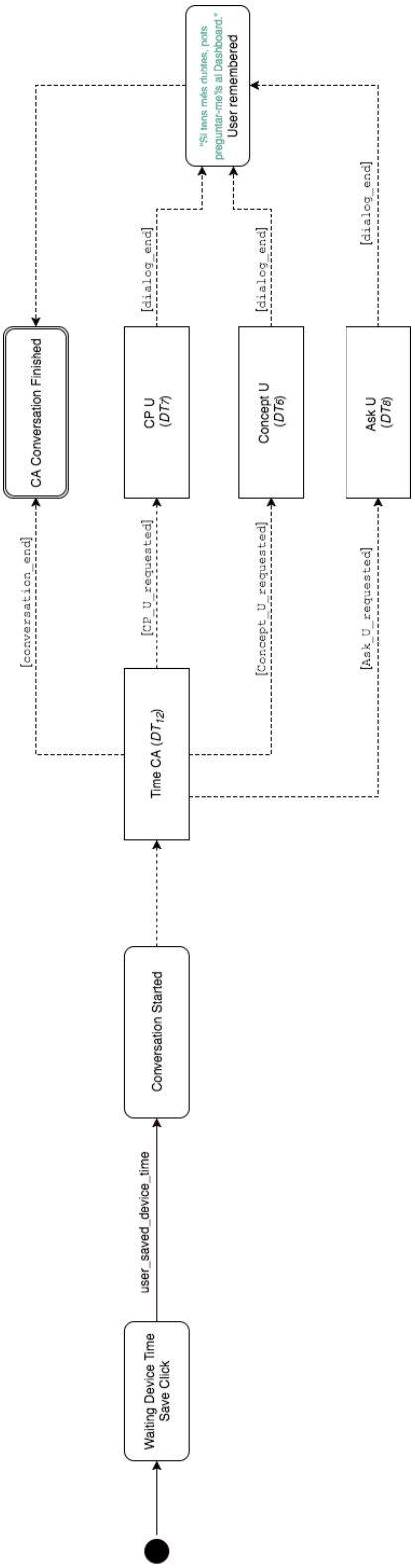

Figure D.5: Electrician Time ($Conv_5$) FSM Diagram.

D.6 Electrician End ($Conv_6$) FSM Diagram

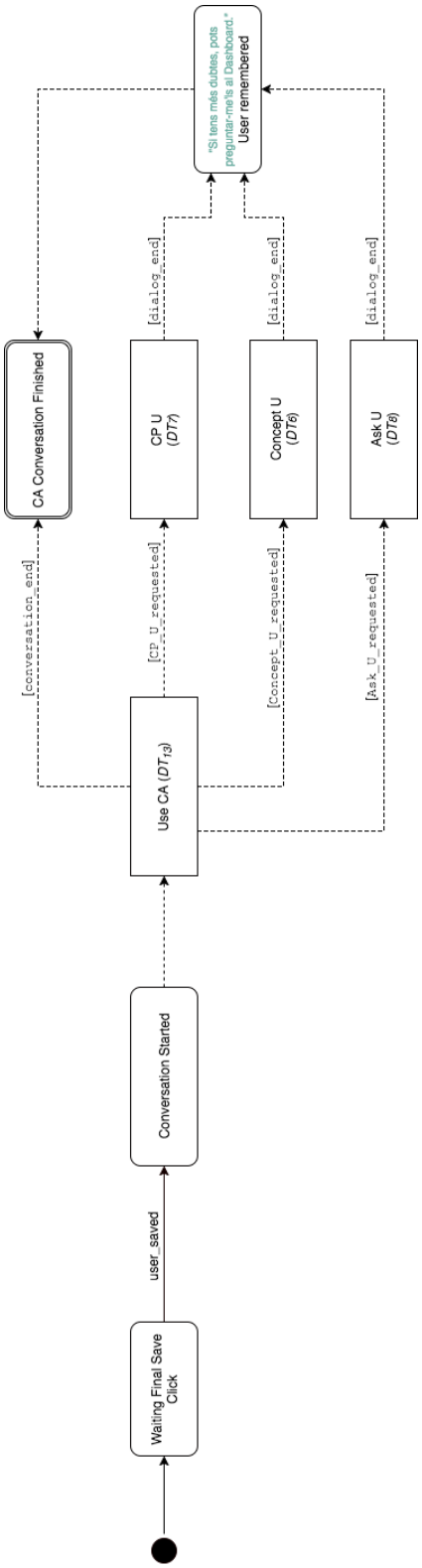

Figure D.6: Electrician End ($Conv_6$) FSM Diagram.

D.7 Information CA (DT_1) FSM Diagram

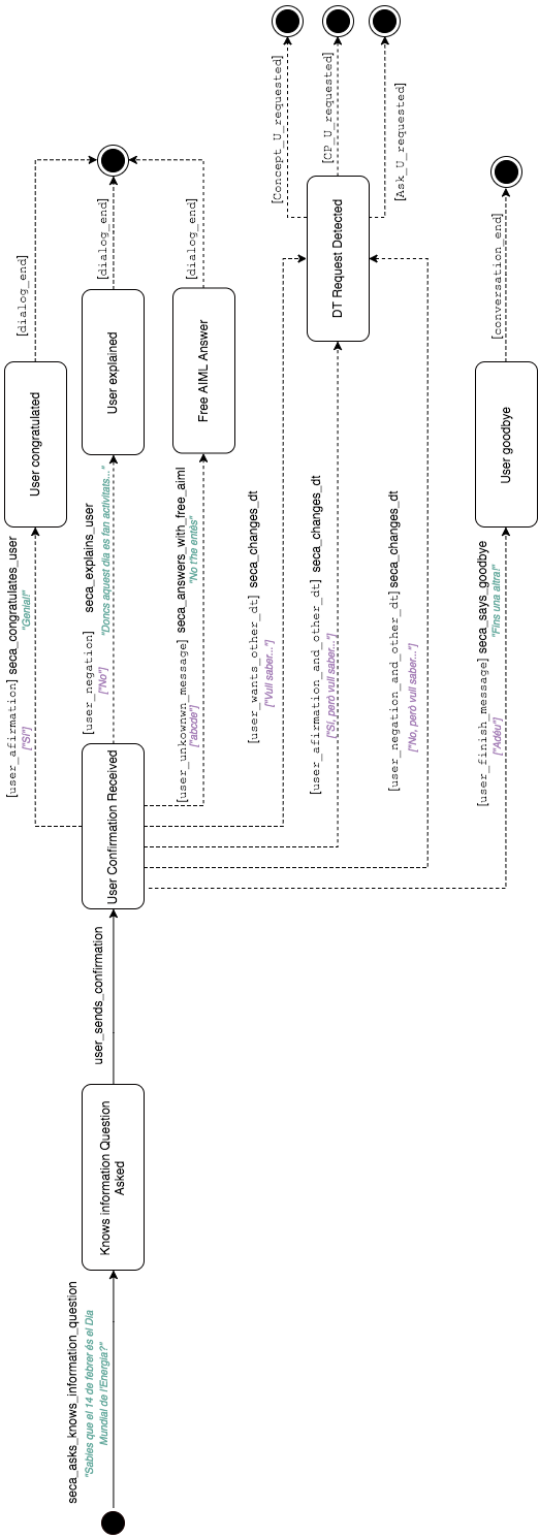

Figure D.7: Information CA (DT_1) FSM Diagram.

D.8 Meaning CA (DT_2) and Ask U (DT_8) FSM Diagram

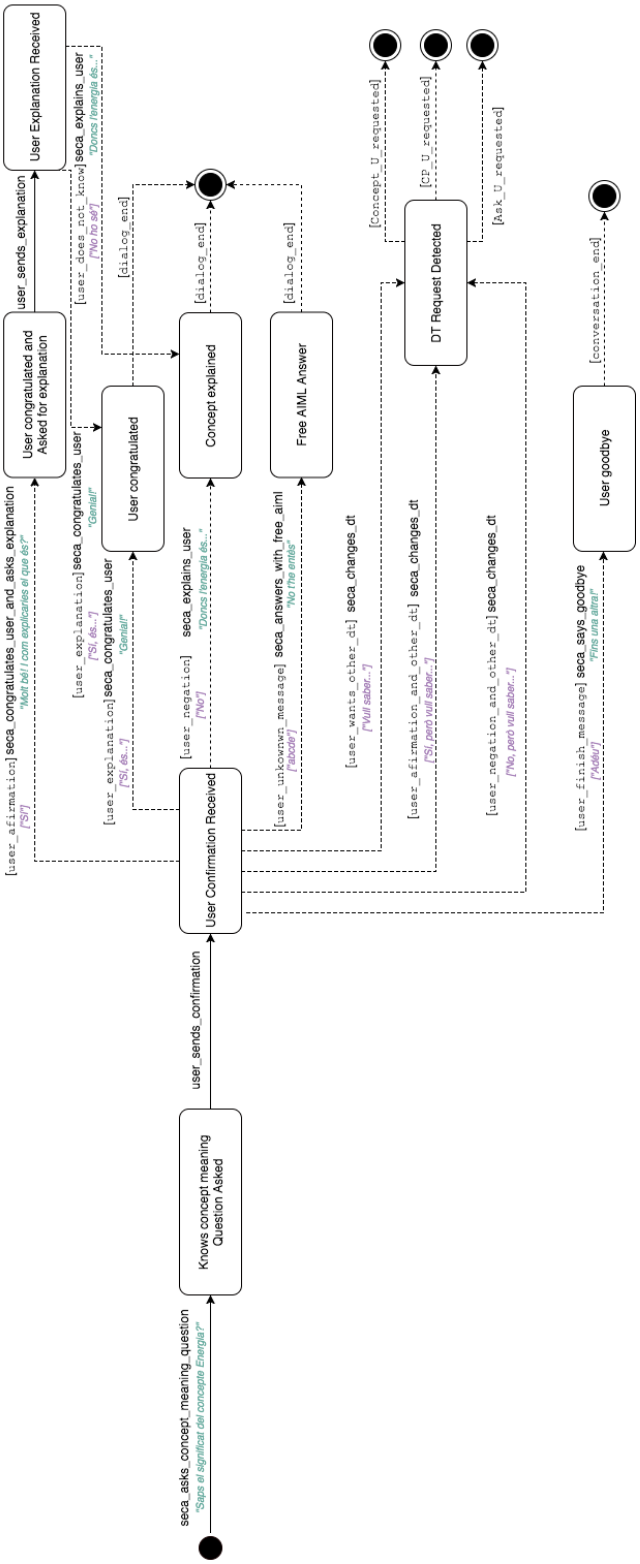

Figure D.8: Meaning CA (DT_2) and Ask U (DT_8) FSM Diagram.

D.9 Concept CA (DT_3) FSM Diagram

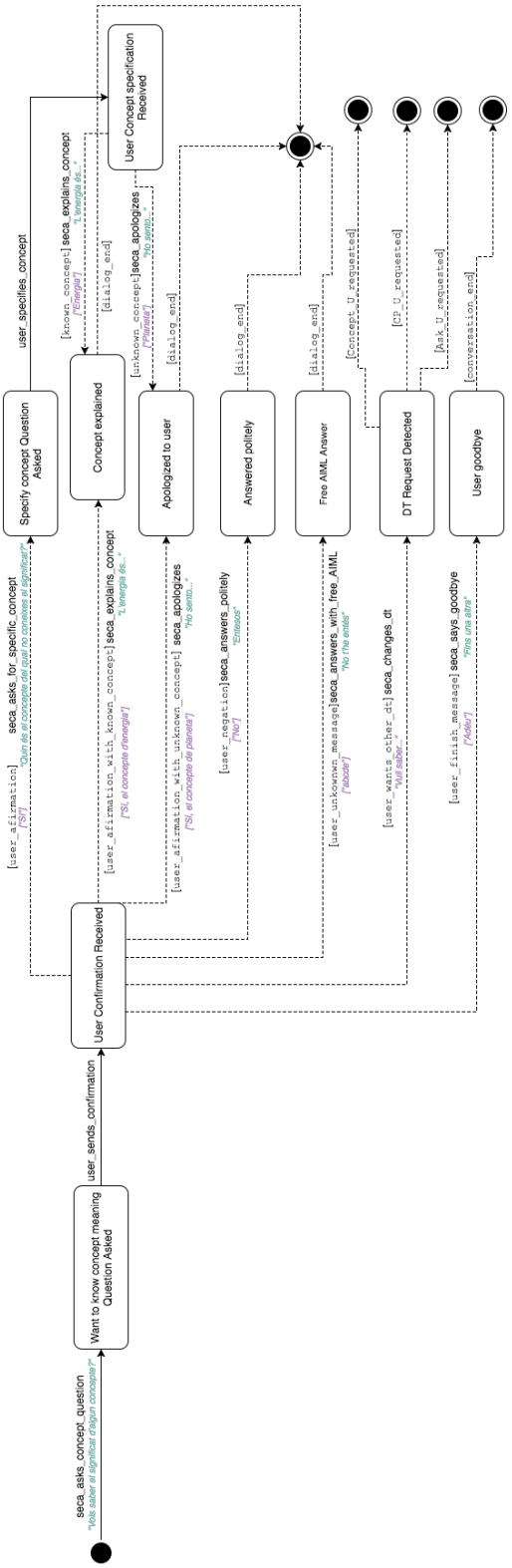

Figure D.9: Concept CA (DT_3) FSM Diagram.

D.10 CP CA (DT_4) FSM Diagram

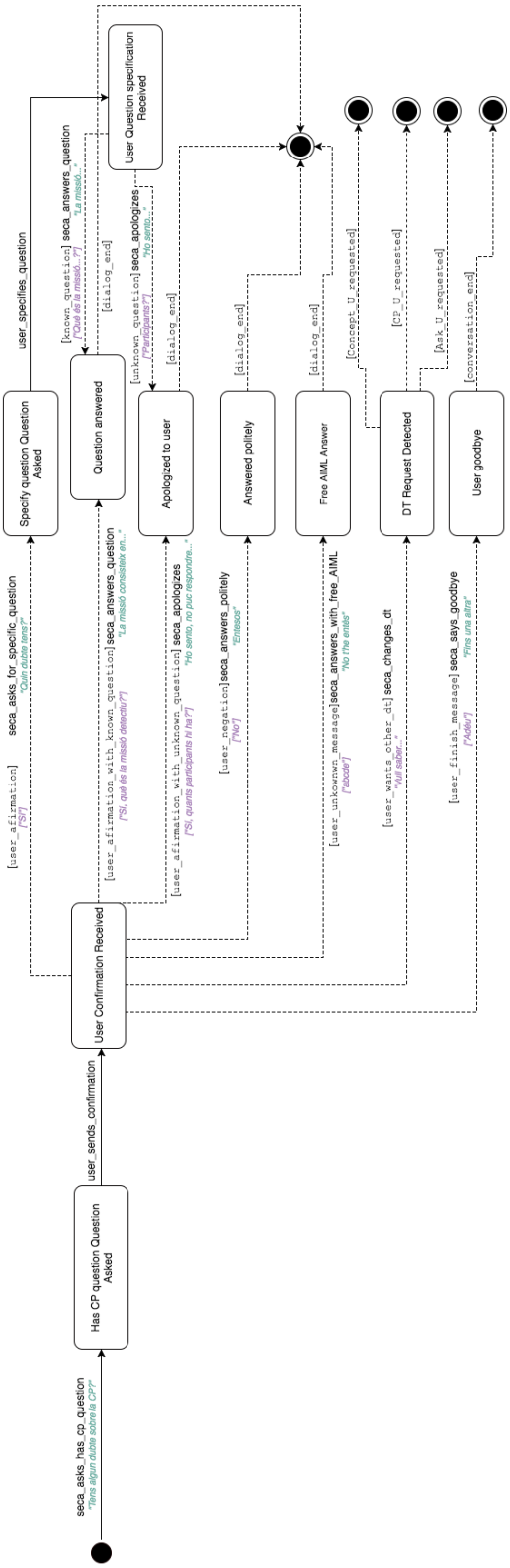

Figure D.10: CP CA (DT_4) FSM Diagram.

D.11 Free CA (DT_5) FSM Diagram

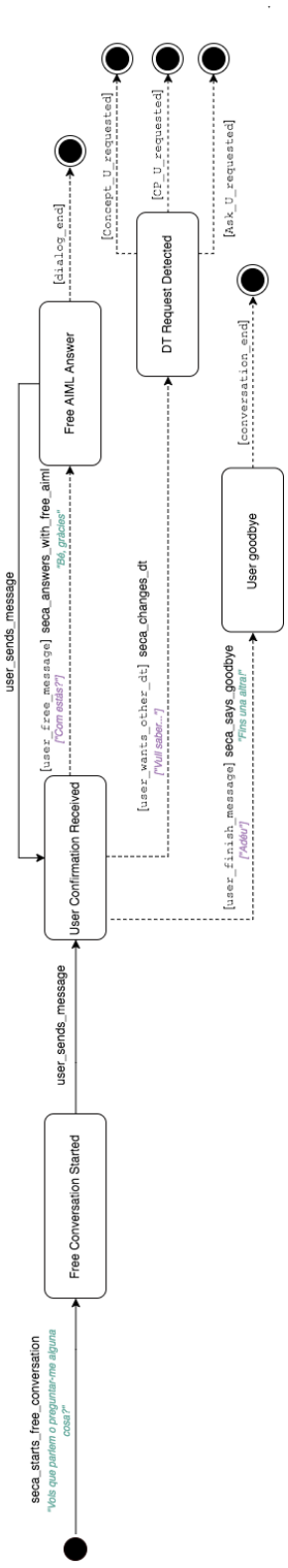

Figure D.11: Free CA (DT_5) FSM Diagram.

D.12 Concept U (DT_6) FSM Diagram

Figure D.12: Concept U (DT_6) FSM Diagram.

D.13 CP U (DT_7) FSM Diagram

Figure D.13: CP U (DT_7) FSM Diagram.

D.14 Combination CA (DT_9) FSM Diagram

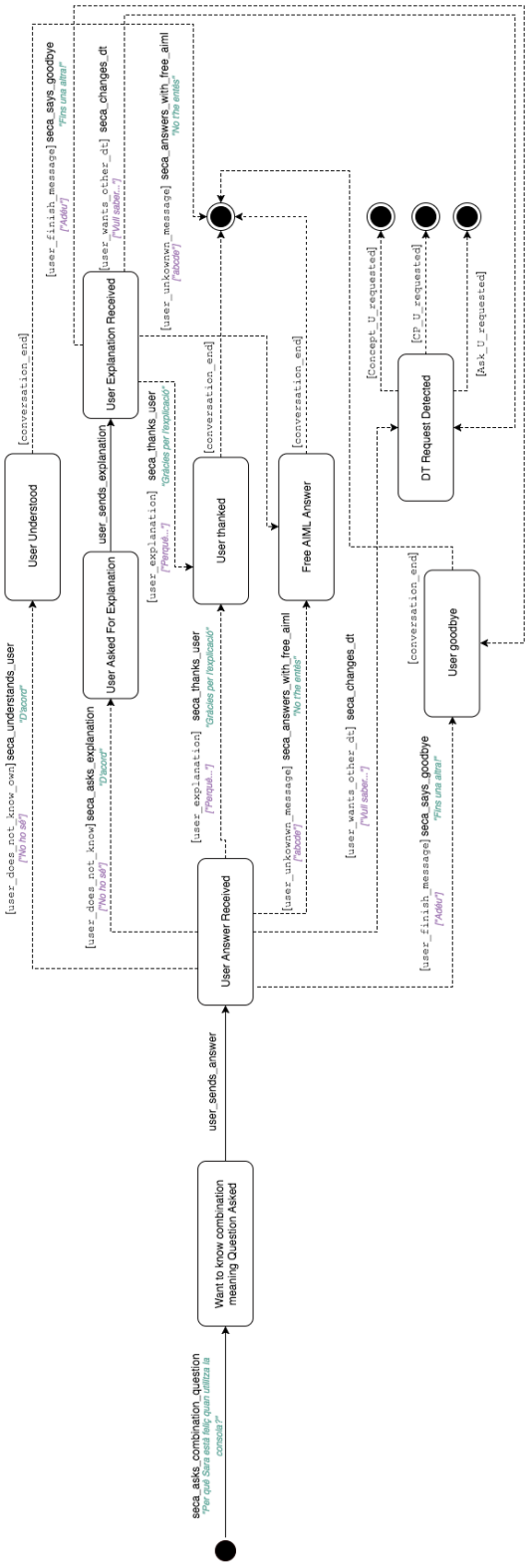

Figure D.14: Combination CA (DT_9) FSM Diagram.

D.15 Behaviour CA (DT_{10}) FSM Diagram

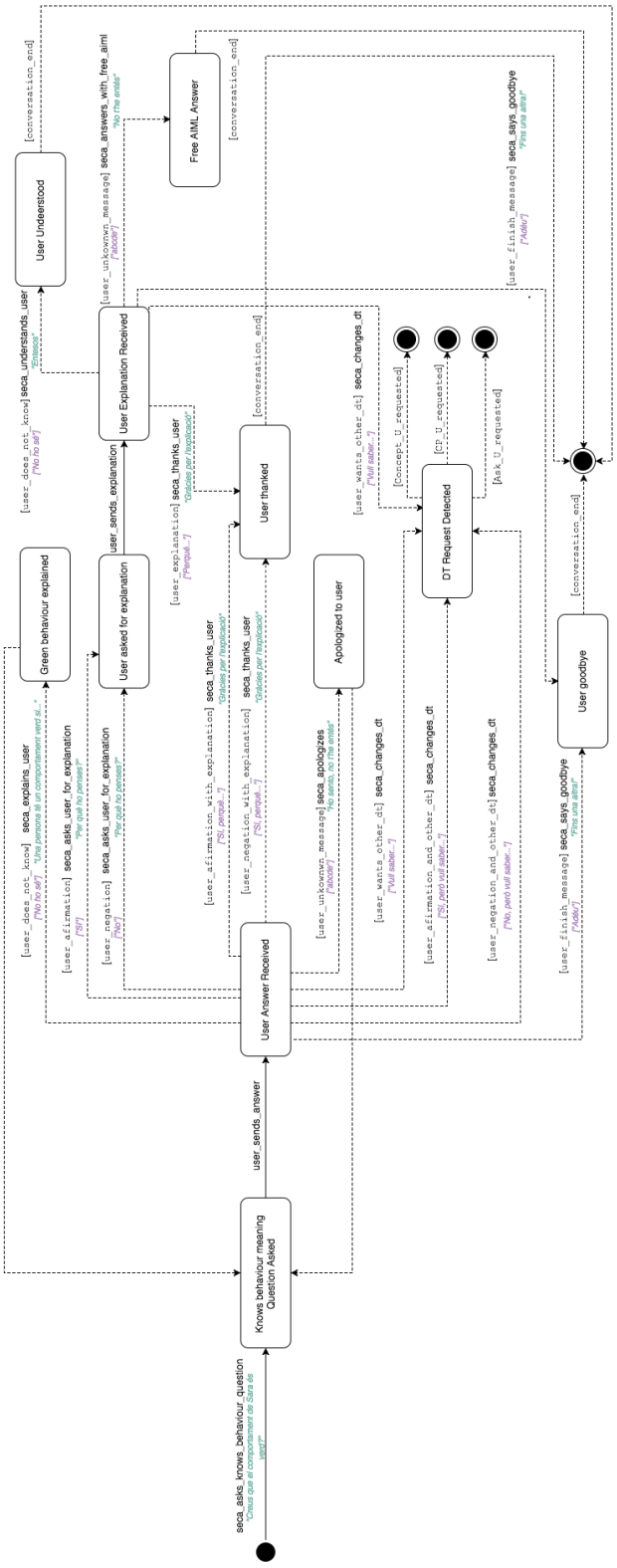

Figure D.15: Behaviour CA (DT_{10}) FSM Diagram.

D.16 Room CA (DT_{11}) FSM Diagram

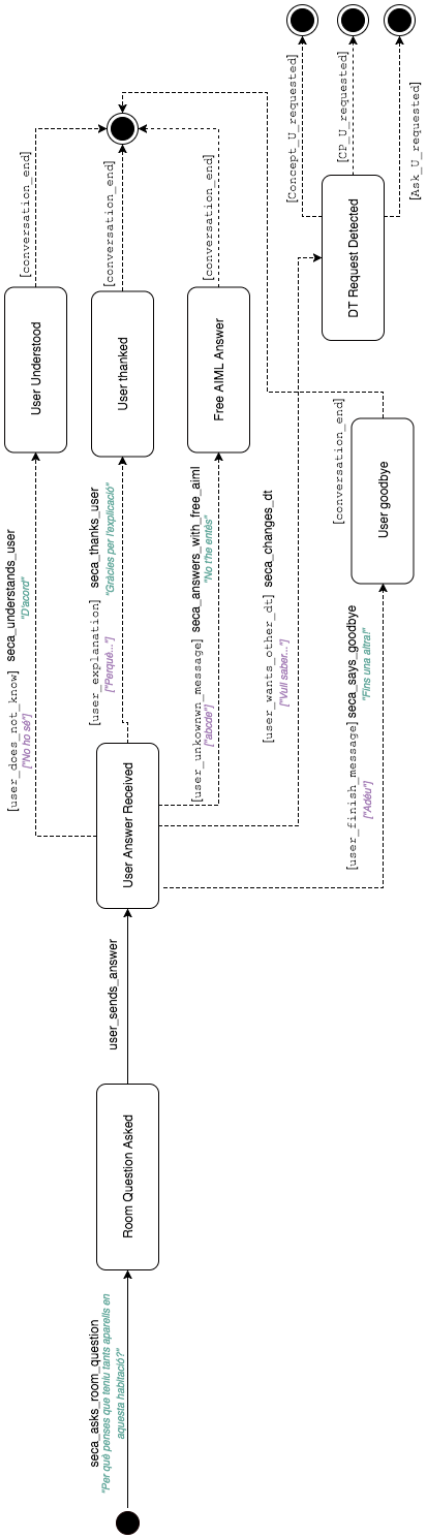

Figure D.16: Room CA (DT_{11}) FSM Diagram.

D.17 Time CA (DT_{12}) FSM Diagram

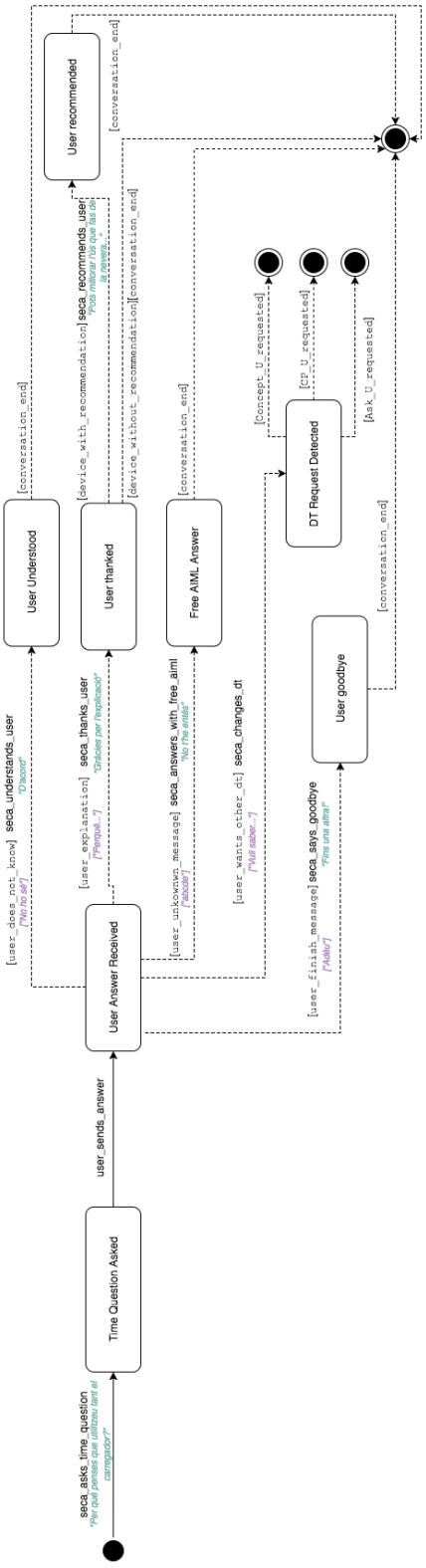

Figure D.17: Time CA (DT_{12}) FSM Diagram.

D.18 Use CA (DT_{13}) FSM Diagram

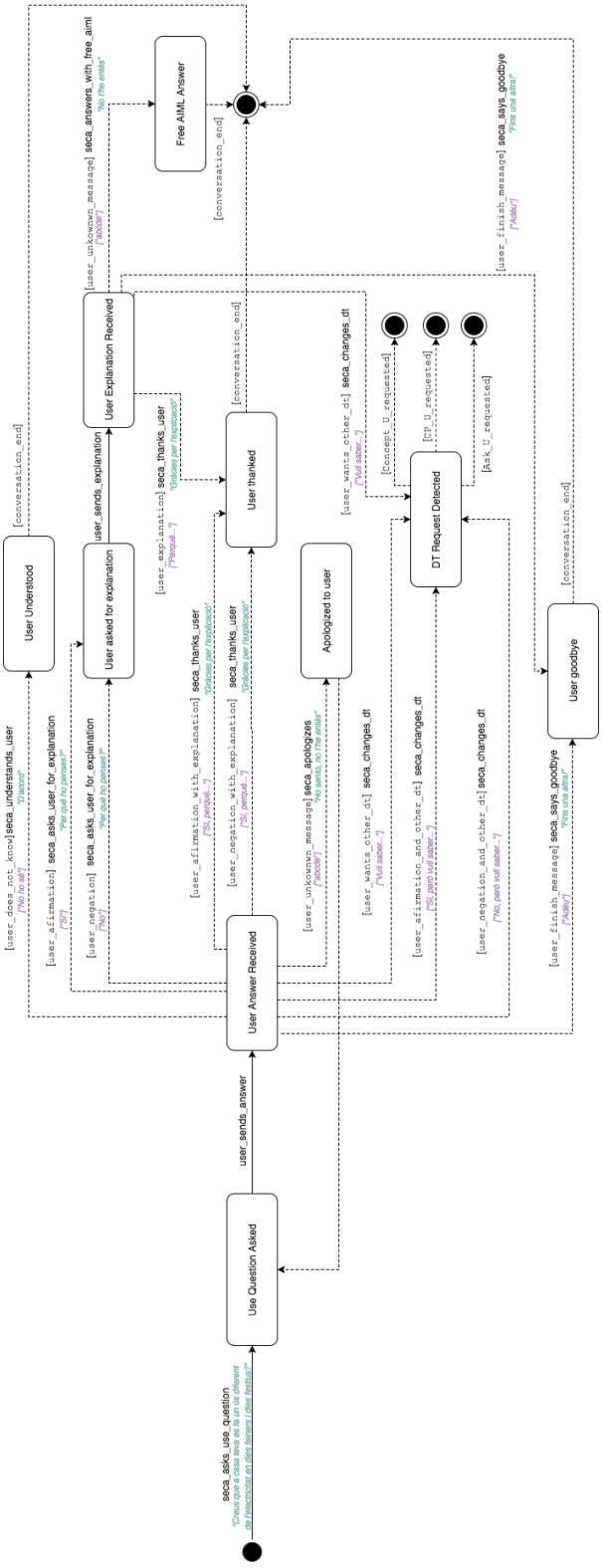

Figure D.18: Use CA (DT_{13}) FSM Diagram.