

---

**SimulAgents: Una Herramienta Intuitiva para  
Modelar y Explorar Fenómenos Sociales  
Complejos**

**SimulAgents: An Intuitive Tool to Model and  
Explore Complex Social Phenomena**

---



**Trabajo de Fin de Grado  
Curso 2023–2024**

**Autores**

**Alberto Ramos Suárez  
Kevin Óscar Arce Vera**

**Directores**

**Gonzalo Méndez Pozo  
Pablo Gervás Gómez-Navarro**

**Grado en Ingeniería del Software  
Grado en Ingeniería Informática**

**Facultad de Informática**

**Universidad Complutense de Madrid**



SimulAgents: Una Herramienta Intuitiva  
para Modelar y Explorar Fenómenos  
Sociales Complejos  
SimulAgents: An Intuitive Tool to Model  
and Explore Complex Social Phenomena

**Trabajo de Fin de Grado en Ingeniería del Software e  
Ingeniería Informática**

**Autores**

**Alberto Ramos Suárez  
Kevin Óscar Arce Vera**

**Directores**

**Gonzalo Méndez Pozo  
Pablo Gervás Gómez-Navarro**

**Convocatoria: Junio 2024**

**Grado en Ingeniería del Software  
Grado en Ingeniería Informática  
Facultad de Informática  
Universidad Complutense de Madrid**

**27 de mayo de 2024**



# Agradecimientos

A Natalia por todo el apoyo y paciencia, a Dai por aguantar el tostón que he dado estos meses.

A toda mi familia, especialmente a mis padres y mis hermanos, por las oportunidades que me han dado y por todo lo que he aprendido de ellos. A Nita, por ser mi lugar seguro y mi soporte durante este último año.



# Resumen

## SimulAgents: Una Herramienta Intuitiva para Modelar y Explorar Fenómenos Sociales Complejos

En la actualidad, están surgiendo multitud de aplicaciones que usan tecnologías apoyadas por grandes modelos de lenguaje que facilitan la vida a profesionales de todos los sectores. Estos modelos tienen un carácter general, por lo que permiten a los usuarios especializarse y particularizar en ciertos casos, para generar sistemas específicos para un sector.

Navegando por Internet, nos encontramos con el estudio de *Generative Agents* (Park et al., 2023), el cual llamó nuestra atención desde que lo leímos por primera vez.

Este estudio desarrolló la ejecución de experimentos y simulaciones multi-agente de Inteligencia Artificial, y su objetivo final era estudiar la emergencia de fenómenos y relaciones sociales entre los agentes. En él se concluyó que efectivamente los agentes eran capaces de retener recuerdos, interactuar con otros agentes, generar nuevas memorias y planear sus próximas acciones.

Al ver esto, decidimos adentrarnos en este artículo, y descubrimos que había fascinado a toda la comunidad investigadora de Inteligencia Artificial, ya que aportaba un valor novedoso y demostraba que estas simulaciones eran coherentes. Por lo que varios de estos grupos de investigadores que se adentraron a realizar extensiones al programa original, enfocando las simulaciones para que hagan otro tipo de acciones (programar videojuegos autónomamente, asignando roles a cada agente, por ejemplo).

Viendo estas extensiones al código original, nos preguntamos si podríamos hacer algo similar nosotros, y hacerlo como TFG. Tras adentrarnos en el código e investigar el funcionamiento completo de la aplicación, tuvimos claro el caso de uso que quisimos enfocar: la democratización de esta herramienta y permitir que perfiles como psicólogos puedan usarla fácilmente.

Lo vimos claro porque a los perfiles a los que más les puede interesar una aplicación como esta es a psicólogos y profesionales que estudian las relaciones sociales. Sin embargo, tal y como estaba inicialmente diseñado el proyecto, era bastante complicado que un perfil no tecnológico, supiese cómo usar esta aplicación. Por lo que teniendo clara nuestra motivación y objetivo principal, nos pusimos manos a la obra.

El código desarrollado durante la realización de este trabajo se encuentra disponible en <https://github.com/NILGroup/TFG-2324-Simulador-MAS-LLM>

## Palabras clave

Modelos de lenguaje grandes, comportamientos emergentes, interacción, interfaz, democratización de la tecnología, agentes inteligentes

# Abstract

## SimulAgents: An Intuitive Tool to Model and Explore Complex Social Phenomena

In recent years, numerous applications have emerged that utilize technologies powered by large language models, easing the lives of professionals across various sectors. These models possess a general nature, enabling users to specialize and tailor them to specific cases, generating systems tailored to a particular industry.

While navigating the internet, we encountered the study of *Generative Agents* (Park et al., 2023), which captured our attention from the moment we first read it.

This study developed the execution of multi-agent Artificial Intelligence experiments and simulations, with its ultimate goal being to investigate the emergence of social phenomena and relationships between agents. It concluded that agents were indeed capable of retaining memories, interacting with other agents, generating new memories, and planning their next actions.

Upon reading this, we decided to delve deeper into this article, and we discovered that it had captivated the entire Artificial Intelligence research community, as it introduced a novel value and demonstrated that these simulations were coherent. Consequently, several of these research groups embarked on extending the original program, focusing the simulations on performing other types of actions (programming video games autonomously, assigning roles to each agent, for example).

Observing these extensions to the original code, we wondered if we could do something similar ourselves, and develop it as a TFG. After delving into the code and investigating the complete functionality of the application, we had a clear understanding of the use case we wanted to focus on: democratizing this tool and enabling profiles such as psychologists to use it easily.

This became evident because the profiles most likely to be interested in such an application are psychologists and professionals who study social relationships. How-

ever, as the project was initially designed, it was quite challenging for a non-tech profile to understand how to use this application. Therefore, with our motivation and main objective clearly defined, we set to work.

The code developed during this work is located in <https://github.com/NILGroup/TFG-2324-Simulador-MAS-LLM>

## **Keywords**

Large language models, emerging behaviors, interaction, interface, technology democratization, intelligent agents

# Índice

<b>1. Introducción</b>	<b>1</b>
1.1. Motivación . . . . .	2
1.2. Objetivos . . . . .	3
1.3. Plan de trabajo . . . . .	3
1.4. Estructura del documento . . . . .	4
1.5. Motivation . . . . .	7
1.6. Objectives . . . . .	8
1.7. Work Plan . . . . .	9
1.8. Document Structure . . . . .	10
<b>2. Estado de la Cuestión</b>	<b>11</b>
2.1. Agentes . . . . .	11
2.1.1. Principios en el diseño . . . . .	12
2.1.2. Agentes inteligentes . . . . .	13
2.2. Simulaciones basadas en agentes . . . . .	14
2.2.1. Historia de las Simulaciones basadas en agentes . . . . .	14
2.2.2. Principios de las Simulaciones basadas en agentes . . . . .	15
2.2.3. Uso reciente de Simulaciones basadas en agentes . . . . .	18
2.3. Sistemas multi-agente para la simulación social . . . . .	19

2.3.1.	Historia de los SMA para la simulación social . . . . .	19
2.3.2.	Ejemplos de aplicaciones de los SMA en simulaciones sociales	20
2.4.	Modelos de lenguaje . . . . .	20
2.4.1.	Modelos simbólicos . . . . .	21
2.4.2.	Modelos estadísticos . . . . .	22
2.4.3.	Modelos neuronales . . . . .	24
2.4.4.	Grandes Modelos de Lenguaje . . . . .	32
2.5.	Generative Agents: el estudio original del que se parte . . . . .	35
2.5.1.	Diseño original del sistema . . . . .	36
2.5.2.	Funcionalidades preexistentes . . . . .	38
2.5.3.	Partes relevantes de la Implementación . . . . .	39
2.6.	Procesamiento del lenguaje natural . . . . .	42
2.6.1.	Historia del PLN . . . . .	42
2.6.2.	PLN para resúmenes de textos . . . . .	43
2.6.3.	Impacto del desarrollo del PLN en la sociedad actual . . . . .	44
2.7.	Computación centrada en el usuario . . . . .	45
2.7.1.	Interacción Persona-Ordenador . . . . .	45
2.7.2.	Proceso de diseño centrado en el usuario . . . . .	46
2.7.3.	Computación afectiva . . . . .	47
2.7.4.	Ejemplos reales de CCU . . . . .	47
2.8.	Relaciones sociales . . . . .	48
2.8.1.	Psicología . . . . .	49
2.8.2.	Informática a lo largo del tiempo . . . . .	50
2.9.	Democratización y extensibilidad de programas informáticos . . . . .	52
2.9.1.	Importancia de la democratización . . . . .	52
2.9.2.	Ejemplos de democratización . . . . .	53
2.9.3.	Importancia de la extensibilidad . . . . .	53

2.9.4. Ejemplos de extensibilidad . . . . .	54
<b>3. Planteamiento de la Solución</b>	<b>55</b>
3.1. Planteamiento inicial . . . . .	55
3.1.1. Estado tecnológico inicial del sistema . . . . .	56
3.2. Ajustes y reconsideraciones . . . . .	57
3.2.1. Cambios en los objetivos . . . . .	57
3.2.2. Limpieza de archivos del repositorio . . . . .	58
3.2.3. Selección del modelo de lenguaje . . . . .	59
3.3. Problemas encontrados . . . . .	69
3.3.1. Llamadas al Back End durante la simulación . . . . .	70
3.3.2. Generación de resúmenes . . . . .	71
3.3.3. Enviar información al Back . . . . .	73
3.3.4. Implementación del chat . . . . .	73
3.3.5. Acoplo de los personajes al mapa original . . . . .	74
3.3.6. Problema instalación de requerimientos del proyecto . . . . .	75
3.4. Conclusión final . . . . .	75
<b>4. Extensiones al Programa Original</b>	<b>79</b>
4.1. Adaptación para la ejecución de comandos desde la interfaz . . . . .	79
4.2. Interacción total con el sistema mediante la interfaz gráfica . . . . .	81
4.3. Funcionalidad de chat con los agentes en tiempo real . . . . .	83
4.4. Herramientas para evaluar consumo de Tokens con el LLM . . . . .	84
4.5. Funcionalidad de resúmenes de las simulaciones . . . . .	85
4.6. Creación desde 0 de una simulación . . . . .	85
<b>5. Interfaz</b>	<b>87</b>
5.1. Proceso de diseño de la interfaz . . . . .	87
5.2. Nuevas vistas . . . . .	89

5.2.1.	Vista de Crear nueva simulación . . . . .	90
5.2.2.	Vista de Visualizar simulaciones . . . . .	93
5.2.3.	Vista de Continuar simulación . . . . .	94
5.2.4.	Vista de Guía de Usuario . . . . .	95
5.3.	Vistas modificadas . . . . .	96
5.3.1.	Vista de landing . . . . .	97
5.3.2.	Vista de visualización de una simulación . . . . .	98
5.3.3.	Vista de ejecución de simulación . . . . .	99
<b>6.</b>	<b>Conclusiones y Trabajo Futuro</b>	<b>103</b>
6.1.	Diferencias entre el Sistema Original y el Actual . . . . .	103
6.1.1.	Funcionalidades ampliadas . . . . .	104
6.1.2.	Nuevas funcionalidades . . . . .	104
6.2.	Conclusiones . . . . .	106
6.3.	Trabajo futuro . . . . .	108
	<b>Contribuciones Personales</b>	<b>119</b>
	<b>Bibliografía</b>	<b>125</b>
<b>A.</b>	<b>Templates evaluados</b>	<b>131</b>
A.1.	Input . . . . .	131
A.1.1.	action_location_object . . . . .	131
A.1.2.	action_object . . . . .	133
A.1.3.	generate_event_triple . . . . .	134
A.1.4.	insight_and_evidence . . . . .	135
A.2.	Output . . . . .	136
A.2.1.	GPT-3.5 . . . . .	136
A.2.2.	7B-32 . . . . .	137

A.2.3. 7B-Q4 . . . . .	138
A.2.4. 13B-Q4 . . . . .	139
A.2.5. 13B-Q6 . . . . .	140

<b>B. Template y Prompt</b>	<b>141</b>
-----------------------------	------------



# Índice de figuras

2.1. Estructura de una neurona de una RNN . . . . .	25
2.2. Estructura de una celda LSTM . . . . .	26
2.3. Estructura de una arquitectura seq2seq . . . . .	27
2.4. Arquitectura Transformer (Imagen tomada de Vaswani et al. (2017))	29
2.5. Visualización del mecanismo <i>Positional Encoding</i> . . . . .	30
2.6. Ilustración del mecanismo de <i>Self-Attention</i> . Tomada de Vaswani et al. (2017) . . . . .	31
2.7. Ilustración de múltiples capas en la fase de atención. Tomada de Vaswani et al. (2017) . . . . .	31
2.8. Tomada de Ma et al. (2024) . . . . .	34
2.9. Arquitectura del estudio <i>Generative Agents</i> (Park et al., 2023) . . . . .	36
2.10. Diferencia de rendimiento de <i>Generative Agents</i> con diferentes arquitecturas (Park et al., 2023) . . . . .	37
2.11. Resumen de textos (adaptada de Adhikari et al. (2020)) . . . . .	44
2.12. Cronología de las mayores redes sociales hasta 2006 (boyd y Ellison, 2007) . . . . .	51
3.1. Diseño del sistema original . . . . .	57
3.2. Diseño del sistema final tras las adaptaciones . . . . .	76
4.1. Diagrama del estado de los comandos actualmente . . . . .	80
5.1. Pasos para diseño centrado en el usuario según Google . . . . .	89

5.2. Barra de navegación en el formato Desktop . . . . .	89
5.3. Barra de navegación en el formato para dispositivos móviles . . . . .	89
5.4. Barra de navegación en el formato para dispositivos móviles, con la barra de navegación desplegada . . . . .	90
5.5. Versión de escritorio de la vista para crear una nueva simulación . . . . .	91
5.6. Versión de móvil de la vista para crear una nueva simulación . . . . .	92
5.7. Vista de visualizar simulaciones en formato escritorio . . . . .	93
5.8. Vista de continuar simulaciones en formato escritorio . . . . .	95
5.9. Vista de la guía de usuario en versión escritorio . . . . .	96
5.10. Vista de landing actualizada . . . . .	97
5.11. Vista de visualización de una simulación . . . . .	98
5.12. Vista de ejecución de una simulación . . . . .	99
5.13. Texto indicativo de que la simulación está ejecutando en segundo plano	100
5.14. Modal de susurro . . . . .	102
5.15. Modal de chat con respuestas simuladas . . . . .	102

# Índice de tablas

3.1. Tiempo de ejecución en segundos. Se incluyen datos de la API de GPT-3.5 . . . . .	65
3.2. Distintas medidas de los modelos. La medida <i>Perplexity</i> es ideal cuando vale 1. A falta de datos para Llama 2 7B 32FP incluimos los del modelo en 16FP . . . . .	65
4.1. Diferencias entre crear simulación antes y después . . . . .	81
4.2. Diferencias entre ejecutar simulación antes y después . . . . .	81
4.3. Diferencias entre susurrar a un agente antes y después . . . . .	82
4.4. Diferencias entre guardar simulación antes y después . . . . .	82
4.5. Diferencias entre salir sin guardar simulación antes y después . . . . .	82
4.6. Diferencias entre ver una demo de simulación antes y después . . . . .	83



# Capítulo 1

## Introducción

*“La creatividad es únicamente unir conceptos”*

— Steve Jobs

Las relaciones sociales han sido tema de estudio en múltiples ocasiones a lo largo del tiempo. El comportamiento de los seres humanos es ciertamente complejo y, en muchos casos, impredecible. Por ello, surge la necesidad de tratar de comprender la base de estas relaciones e intentar predecir los comportamientos mediante las simulaciones.

En los últimos años, y especialmente en el último, hemos visto un desarrollo fulgurante de la Inteligencia Artificial generativa. Uno de los casos de uso más interesantes de estas tecnologías son los Modelos Grandes de Lenguaje (Large Language Models), de propósito general, los cuales nos permiten interactuar mediante el lenguaje natural directamente con estos sistemas de IA generativa.

Al desarrollar estos Modelos del Lenguaje, si se tiene una serie de ejemplos lo suficientemente grande, el programa podrá generar situaciones completamente únicas y originales, imitando ejemplos ya conocidos pero sin reutilizarlos de manera literal.

En este momento entra en juego el comportamiento humano. Si tenemos unos modelos muy grandes, capaces de obedecer a esquemas de entrada y crear situaciones únicas, ¿podríamos simular relaciones sociales humanas verosímiles usando estas tecnologías?

Este experimento fue llevado a cabo en el estudio de Park et al. (2023), en el cual nos hemos basado al realizar nuestro Trabajo de Fin de Grado. En este experimento logran crear agentes independientes los cuales interactúan entre ellos, tienen recuerdos de eventos ocurridos y son capaces de desarrollar relaciones sociales entre sí.

## 1.1. Motivación

La motivación inicial de este trabajo era poder crear historias a partir de simulaciones con agentes de Inteligencia Artificial. Sin embargo, al encontrar el estudio de *Generative Agents* (Park et al., 2023), decidimos cambiar el objetivo y centrarnos en cómo gente de perfiles no tecnológicos, especialmente psicólogos, pueden estudiar las relaciones sociales entre humanos, partiendo de una simulación que ya funcionaba y lograba que los personajes tuviesen recuerdos e interaccionasen de manera verosímil.

Por tanto, a partir de ese momento, nuestro principal objetivo se convirtió en crear una aplicación en la cual psicólogos u otro tipo de usuarios puedan estudiar relaciones sociales, indicando diferentes situaciones y facilitando los cambios entre simulaciones para estudiar las diferencias de comportamiento.

Con un sistema como el citado, disponemos de lo necesario para crear un entorno en el que intervengan varios agentes e interactúen entre sí. Una implementación de esta idea es la que tenemos en el repositorio *generative\_agents*<sup>1</sup> creado por los autores del paper mencionado anteriormente.

Disponer de una tecnología capaz de simular el comportamiento humano en contextos sociales nos crea la necesidad de experimentar con estas capacidades en distintas situaciones. Esta necesidad se ve reflejada en otro estudio, el cual pone a prueba y expande el mismo trabajo original, de Park et al. (2023), pero en esta ocasión midiendo la capacidad de un grupo de agentes para generar software (Qian et al., 2023). En este estudio se comprueba que estos mismos agentes son capaces de asociarse, formar equipos y generar videojuegos funcionales con código real.

Se ve así el potencial de esta tecnología y el interés de la gente por definir sus límites, aplicaciones o simplemente experimentar. Sin embargo todas estas motivaciones se ven muy limitadas a la hora de poner en funcionamiento este trabajo en nuevos escenarios debido a la complejidad de la ejecución de la aplicación, la cual es difícil de usar para perfiles ajenos al mundo de la programación, y muy difícilmente extensible debido a la situación inicial del código (personajes y mapas preseleccionados y no intercambiables, rigidez a la hora de ejecutar simulaciones, falta de modificación del contexto de la aplicación...).

Es por ello que proponemos en este Trabajo de Fin de Grado facilitar el acceso e interacción con la tecnología que proporcionan en *Generative Agents*, además de ofrecer nuevas funcionalidades con las que explorar e interactuar con la simulación en tiempo real.

---

<sup>1</sup>[https://github.com/joonspk-research/generative\\_agents](https://github.com/joonspk-research/generative_agents)

## 1.2. Objetivos

Para lograr el objetivo principal del TFG mencionado anteriormente, que es facilitar el acceso e interacción y extender funcionalidades del estudio de *Generative Agents*, hemos decidido estructurarlo en los siguientes puntos:

- Creación de simulaciones configurando agentes y/o el escenario. Ya que actualmente solamente existe un mapa predeterminado y una serie de agentes predefinidos. La finalidad es permitir una mayor interactividad con el entorno, añadiendo mapas y pudiendo personalizar a los agentes.
- Permitir la interacción con el estado de los agentes durante la simulación. En cualquier momento, el usuario podrá interactuar con los agentes, chateando en tiempo real con ellos o 'susurrándoles' las próximas acciones que deben tomar.
- Visión de la simulación, a través de un personaje o de una forma general y sintetizada. También permitir a los usuarios, una vez finalizada una simulación, poder hacer un resumen de toda la simulación como conjunto o hacerlo desde el punto de vista de uno de los agentes.
- Gestión de las simulaciones, permitiendo el guardado de estas y la creación en base a anteriores. Esto permitirá poder ver simulaciones repetidas o extenderlas.
- Integración de todo lo anterior en una interfaz web nueva y creada completamente desde cero, comunicándose con el back en tiempo real y sin necesidad de usar una terminal para comunicarnos.
- Integración con el modelo de lenguaje y la API de OpenAI actualizada, ya que las llamadas originales usaban una API anterior que no funcionaba correctamente.

## 1.3. Plan de trabajo

Con el fin de cumplir los objetivos planteados en la sección anterior, se ha fijado una planificación dividida en las siguientes etapas:

- **Etapas de investigación y definición de objetivos.** En esta primera parte del desarrollo, nos centraremos en investigar artículos científicos y libros que traten sobre temas similares a la interacción de agentes de IA, así como definiendo los objetivos básicos, en qué sentido vamos a extender el trabajo descrito en el artículo original de *Generative Agents* y comenzar a redactar esbozos iniciales de la introducción y estado de la cuestión de la memoria.

- Tras concluir con las investigaciones iniciales y la adaptación al tema, comenzaremos la fase del **desarrollo del proyecto**, realizando así los diseños de las distintas interfaces, el desarrollo de las mismas y la producción de los capítulos principales de la memoria.
- Finalmente, llegaremos a la fase de **finalización y revisión del desarrollo**, en la cual finalizaremos todos los detalles y objetivos adicionales que queramos incluir en el proyecto, una vez finalizados los objetivos propuestos. En esta etapa también añadiremos el resumen, resultados, conclusiones y trabajo futuro, así como también revisaremos el resultado final de la memoria y el trabajo en su totalidad.

## 1.4. Estructura del documento

El presente documento está formado por siete capítulos, incluyendo este capítulo introductorio como el primero de ellos con el fin de conocer la motivación que lleva al desarrollo del proyecto y los objetivos de este. Además, hay un último capítulo en el que cada uno de los estudiantes explicamos el trabajo que hemos realizado individualmente. A continuación, se presentan el resto de capítulos que componen la memoria:

- **Capítulo 2:** En el Estado de la Cuestión se tratan temas que aportan un contexto importante al presente trabajo. Por una parte se abordan temas técnicos, como los agentes inteligentes, las simulaciones y los modelos de lenguaje, por otro lado, la unión entre la tecnología y el comportamiento humano, con el procesamiento del lenguaje natural y la computación centrada en el humano, y finalmente, se tratan temas relacionados puramente con las relaciones sociales y la democratización de sistemas informáticos.
- **Capítulo 3:** En este capítulo se abordan las diferencias entre las funcionalidades que ya existían en el sistema que empleamos, las que existían pero las ampliamos, y las nuevas que creamos durante el presente trabajo. En este capítulo solo se aborda el trabajo desde un punto de vista de funcionalidades, a alto nivel, y no el estado tecnológico a bajo nivel.
- **Capítulo 4:** En el Planteamiento de la Solución se tratan varios puntos para aportar contexto y recalcar hitos que han ido ocurriendo durante el desarrollo. Por un lado, se aporta desde un punto de vista tecnológico y de bajo nivel el estado del sistema inicialmente. Tras esto, se tratan todos los ajustes, cambios y reconsideraciones que acontecieron mientras se desarrollaba el propio trabajo, así como problemas encontrados y cómo se han solucionado.
- **Capítulo 5:** En el cual se tratan todas las extensiones que se han realizado al sistema original, a bajo nivel y explicando los cambios, su importancia y el motivo.

- **Capítulo 6:** Desde el punto de vista de la interfaz, se presentan todos los cambios, nuevas vistas y páginas que han sido creadas para que la aplicación pueda ser completamente utilizada desde la web, ya que originalmente había que ejecutar comandos en la terminal para realizar ciertas acciones.
- **Capítulo 7:** En el capítulo de Conclusiones y Trabajo Futuro se añaden posibles extensiones al sistema u objetivos que habríamos implementado si hubiéramos tenido más tiempo, a tener en cuenta para el futuro.



# Introduction

*“Creativity is just connecting things”*

— Steve Jobs

Social relationships have been a subject of study on multiple occasions over time. Human behavior is certainly complex and, in many cases, unpredictable. Therefore, there arises a need to try to understand the basis of these relationships and attempt to predict behaviors through simulations.

In recent years, and especially in the last year, we have witnessed a rapid development of generative Artificial Intelligence. One of the most interesting use cases of these technologies is the Large Language Models, of general purpose, which allow us to interact using natural language directly with these generative AI systems.

When developing these Language Models, if there is a sufficiently large series of examples, the program can generate completely unique and original situations, imitating already known examples but without reusing them literally.

At this point, human behavior comes into play. If we have very large models, capable of following input schemas and creating unique situations, could we simulate plausible human social relationships using these technologies?

This experiment was carried out in the study by Park et al. (2023), on which we have based our Final Degree Project. In this experiment, they manage to create independent agents that interact with each other, have memories of events that occurred, and are capable of developing social relationships among themselves.

## 1.5. Motivation

The initial motivation for this work was to be able to create stories from simulations of Artificial Intelligence agents. However, upon finding the study by *Generative Agents*(Park et al., 2023), we decided to change the objective and focus on how people with non-technological profiles, especially psychologists, can study human

social relationships, based on a simulation that already worked and achieved that the characters had memories and interacted plausibly.

Therefore, from that moment, our main objective became to create an application in which psychologists or other types of users can study social relationships by indicating different situations and facilitating changes between simulations to study behavioral differences.

With a system like the one mentioned, we have what is necessary to create an environment where several agents intervene and interact with each other. An implementation of this idea is what we have in the repository *generative\_agents*<sup>2</sup> created by the authors of the aforementioned paper.

Having a technology capable of simulating human behavior in social contexts creates the need to experiment with these capabilities in different situations. This need is reflected in another study, which tests and expands the same original work by Park et al. (2023), but this time measuring the ability of a group of agents to generate software (Qian et al., 2023). In this study, it is verified that these same agents are capable of associating, forming teams, and generating functional video games with real code.

Thus, the potential of this technology and the interest of people in defining its limits, applications, or simply experimenting is evident. However, all these motivations are very curtailed when it comes to implementing this work in new scenarios due to the complexity of executing the application, which is difficult to use for profiles outside the programming world, and very difficult to extend due to the initial state of the code (preselected characters and maps, non-interchangeable, rigidity in executing simulations, lack of modification of the application context...).

That is why in this Final Degree Project we propose to facilitate access and interaction with the technology provided in *Generative Agents*, as well as offering new functionalities to explore and interact with the simulation in real-time.

## 1.6. Objectives

To achieve the main objective of the Final Degree Project mentioned above, which is to facilitate access and interaction and extend functionalities of the study by *Generative Agents*, we have decided to structure it in the following points:

- Creation of simulations by configuring agents and/or the scenario. Currently, there is only a predetermined map and a series of predefined agents. The goal is to allow greater interactivity with the environment, adding maps and being able to customize the agents.

---

<sup>2</sup>[https://github.com/joonspk-research/generative\\_agents](https://github.com/joonspk-research/generative_agents)

- Allowing interaction with the state of the agents during the simulation. At any time, the user can interact with the agents, chatting in real-time with them or 'whispering' the next actions they should take.
- Visualization of the simulation, through a character or in a general and summarized way. Also allowing users, once a simulation is finished, to make a summary of the entire simulation as a whole or from the point of view of one of the agents.
- Management of simulations, allowing them to be saved and created based on previous ones. This will allow repeated simulations to be seen or extended.
- Integration of all the above into a new web interface created completely from scratch, communicating with the backend in real-time and without the need to use a terminal to communicate.
- Integration with the updated OpenAI language model and API, as the original calls used a previous API that did not work correctly.

## 1.7. Work Plan

To meet the objectives set out in the previous section, a plan has been established divided into the following stages:

- **Research and definition of objectives stage.** In this first part of the development, we will focus on researching scientific articles and books that deal with topics similar to the interaction of AI agents, as well as defining the basic objectives, in what way we are going to extend the work described in the original article by *Generative Agents* and begin drafting initial sketches of the introduction and state of the art of the report.
- After concluding the initial research and adaptation to the topic, we will begin the **project development phase**, thus carrying out the designs of the different interfaces, their development, and the production of the main chapters of the report.
- Finally, we will reach the **completion and review of development phase**, in which we will finalize all the details and additional objectives that we want to include in the project, once the proposed objectives are completed. In this stage, we will also add the summary, results, conclusions, and future work, as well as review the final result of the report and the work as a whole.

## 1.8. Document Structure

This document is composed of seven chapters, including this introductory chapter as the first one to understand the motivation behind the project development and its objectives. Additionally, there is a final chapter where each student explains the work we have done individually. Below, the rest of the chapters that make up the report are presented:

- **Chapter 2:** In the State of the Art, topics that provide important context to the present work are addressed. On one hand, technical topics such as intelligent agents, simulations, and language models are discussed, on the other hand, the union between technology and human behavior, with natural language processing and human-centered computing, and finally, topics purely related to social relationships and the democratization of computer systems are addressed.
- **Chapter 3:** In this chapter, the differences between the functionalities that already existed in the system we used, those that existed but were expanded, and the new ones we created during the present work are addressed. In this chapter, the work is only approached from a functionality point of view, at a high level and not the technological state at a low level.
- **Chapter 4:** In the Solution Approach, several points are addressed to provide context and highlight milestones that occurred during the development. On one hand, the initial state of the system is provided from a technological and low-level point of view. After this, all the adjustments, changes, and reconsiderations that occurred while the work was being developed, as well as problems encountered and how they were solved, are addressed.
- **Chapter 5:** In which all the extensions made to the original system are addressed, at a low level and explaining the changes, their importance, and the reason.
- **Chapter 6:** From the interface point of view, all the changes, new views, and pages that have been created so that the application can be completely used from the web, as originally certain actions had to be executed through terminal commands.
- **Chapter 7:** In the Conclusions and Future Work chapter, possible extensions to the system or objectives that we would have implemented if we had more time are added, to be considered for the future.

## Estado de la Cuestión

En este capítulo, se estudiarán detalladamente las simulaciones basadas en agentes de inteligencia artificial y modelos de lenguaje, así como las complejidades en las relaciones sociales y la interacción persona-ordenador. El objetivo fundamental es contextualizar la evolución histórica y actual de estos temas, destacando la relevancia de su estudio y desarrollo, subrayando la importancia de que estos sistemas sean fácilmente extensibles y utilizados por profesionales de distintos ámbitos. El capítulo se dividirá en varios puntos, los cuales se pueden englobar en dos secciones.

La primera mitad del presente capítulo del Estado de la Cuestión se centra en las simulaciones basadas en agentes y su aplicación en entornos de inteligencia artificial. Se explorarán los avances tecnológicos, las metodologías y los desarrollos más recientes que han influido en la creación de sistemas de simulación avanzados. Asimismo, se examinará la relación de estos agentes con modelos de lenguaje, identificando los desafíos de esta convergencia tecnológica.

Después, se revisarán los desarrollos más destacados en procesamiento del lenguaje natural (PLN) y cómo estos contribuyen a la mejora de la interacción entre humanos y sistemas de inteligencia artificial. Además, se explorarán los aspectos psicológicos relacionados con las relaciones sociales en el contexto de la informática. También se centrará en la interacción persona-ordenador, se analizarán las tendencias actuales en diseño centrado en el usuario y las estrategias para garantizar que las simulaciones de inteligencia artificial sean accesibles a todos los públicos, independientemente de su desconocimiento sobre la tecnología.

### 2.1. Agentes

Los sistemas informáticos se asumen como obedientes máquinas que siguen todas las instrucciones que les solicitamos o programamos, resultando en un fallo del sistema o pérdida de archivos en el mejor de los casos o en la pérdida de vidas en

el peor. Sin embargo hay ciertas aplicaciones o tareas en las que necesitamos una capacidad de resiliencia y decisión que con los sistemas tradicionales no es posible obtener. Sobre todo en entornos de rápido cambio, impredecibles y donde la cantidad de posibles situaciones no permita la posibilidad de plantearse cualquier escenario desde el diseño.

Un típico ejemplo de la necesidad de este tipo de sistemas es el envío de una sonda espacial a una distancia de la tierra que imposibilite el uso de un equipo humano que realice el seguimiento y toma de decisiones ante cualquier eventualidad imprevista, debido al retardo en la comunicación con la sonda debido a la distancia. Estas necesidades llevan a empresas como la NASA a investigar en agentes con más capacidad de autonomía para tener un mejor desempeño y confianza en este tipo de sistemas (Rouff, 2002).

Otro ejemplo, en una situación más cercana, sería la forma en que se localizan, actualizan y registran las páginas web en Internet. Cualquiera que se haya visto envuelto en el desarrollo de una página web probablemente se haya topado con un archivo llamado *robots.txt*, que es la forma que tienes de comunicarte con los agentes que Google utiliza para registrar o actualizar las páginas a través de Internet<sup>1</sup>.

A pesar de que *Agente* es un término ampliamente usado en la literatura y los sistemas con los que interactuamos cada día, no hay un consenso en la definición de Agente, el mayor acuerdo que se encuentra en torno a esta definición es la centralidad que ocupa la autonomía para un Agente. Parte del desacuerdo en esta definición se debe al amplio número de sectores en los que es utilizado, teniendo cada uno de estos unas necesidades muy distintas, siendo necesaria alguna característica de los Agentes en ciertos sectores pero totalmente indeseables en otros, un ejemplo sería la capacidad de aprender de los Agentes. En cualquier caso se puede tomar prestada la definición dada por Weiss (1999) para tener una idea más cercana de lo que pretendemos expresar al hablar de Agentes.

Un Agente es un *sistema computacional* que se encuentra en cierto *entorno*, y es capaz de *actuar* de forma autónoma en este entorno con el fin de cumplir sus objetivos de diseño

### 2.1.1. Principios en el diseño

Entre los factores más importantes a la hora de diseñar un Agente se encuentran los siguientes (Russell y Norvig, 2016), (Wooldridge, 2009):

- **La capacidad de control del Entorno** En el mejor de los casos el Agente tiene un control parcial sobre el entorno, pero no en la mayoría.

---

<sup>1</sup><https://developers.google.com/search/docs/crawling-indexing/robots/intro?hl=es>

- **Decisión de la acción a ejecutar** Esta toma de decisiones se puede ver afectada por el tipo de entorno en el que nos encontramos y siempre está enfocada en conseguir los objetivos definidos de la forma más satisfactoria posible.
- **No-determinismo de las acciones ejecutadas** La falta de control supone que una misma acción no dará siempre los mismos resultados.
- **Posibilidad de fallo** El mencionado no-determinismo nos lleva, en numerosas ocasiones, a no obtener el resultado deseado, por ello, incluso en los Agentes más sencillos, hay que tener en cuenta la posibilidad de fallo.

Con lo dicho hasta el momento damos cabida a un gran número de sistemas, en esta definición cabe cualquier *daemon* de nuestro sistema, por ejemplo el programa **xbiff** que es capaz de monitorear un sistema UNIX y notificar al usuario en caso de que haya un nuevo correo.

### 2.1.2. Agentes inteligentes

Una aplicación más buscada y útil son los Agentes Inteligentes. La diferencia con los anteriores es la flexibilidad que estos tienen de actuar de forma autónoma y flexible para cumplir con sus objetivos de diseño. Por flexible entendemos lo descrito en Jennings y Wooldridge (1998):

- **Sensible a su entorno** siendo capaces de reaccionar en los momentos oportunos en los que su entorno cambia.
- **Proactivo** además de ser capaces de responder a los cambios deben ser capaces de tomar la iniciativa en la búsqueda de cumplir sus objetivos.
- **Sociable** también han de ser capaces de comunicarse con otros agentes o humanos, cuando lo consideren necesario, para cumplimentar sus objetivos.

Este tipo de sistemas permite encarar un gran número de problemas y desarrollar sistemas en los que podemos disponer de Agentes Inteligentes capaces de realizar nuevas tareas. Un claro ejemplo de esto, que ya mencionábamos anteriormente, es Internet, donde tenemos una gran red de organizaciones desplegando información y cada una de ellas siguiendo sus propios objetivos y metodologías, haciendo de este un entorno muy heterogéneo y dinámico.

Este tipo de agentes es también un componente central en Park et al. (2023), siendo en base a estos y su interacción (entre ellos y su entorno), que obtenemos las complicadas dinámicas que acabamos observando. Viendo una vez más de la capacidad de estos sistemas para acomodarse en dominios cambiantes y de gran complejidad.

## 2.2. Simulaciones basadas en agentes

En este apartado mostraremos de forma superficial el concepto de Simulaciones basadas en Agentes, así como numerosos ejemplos en los que se ha visto aplicada esta técnica, ya que es un concepto central a la hora de desarrollar el trabajo en el que se basa nuestro TFG (Park et al., 2023).

Las Simulaciones basadas en Agentes, o también conocidas como Agent-Based Model (ABM), tratan de dar lugar a sistemas complejos a través del uso de agentes. En base a la interacción, entre agentes y el entorno, se pretende generar comportamientos y patrones a nivel de sistema (Bankes, 2002).

Este tipo de sistema se caracteriza por:

- Tener un conjunto de agentes que siguen reglas sencillas.
- Estos agentes interactúan entre sí de forma continuada.
- Hay características a nivel de sistema que ocurren de forma emergente.
- Las reglas elegidas para los agentes se entienden como fundamentales.
- Una vez establecidas dichas reglas se observa la evolución del sistema (aquí es donde frecuentemente ocurre el fenómeno de la emergencia).
- El dominio de los fenómenos que se pretende recrear son de gran complejidad.

Es por esta última característica que la variedad de escenarios o fenómenos recreables se vuelve tan amplio, permitiendo a una gran cantidad de disciplinas hacer uso de estas técnicas. Tenemos ejemplos en multitud de Ciencias Sociales, Organizacionales, Ecológicas, Económicas, etc... (Troitzsch, 2009), (Heckbert et al., 2010).

Unido a esto, tenemos la importancia que se ha conferido al entendimiento de un alto número de sistemas complejos en múltiples ámbitos del conocimiento, por ejemplo tenemos el clima terrestre, la ecología, el cerebro, las relaciones sociales, etc...

Así, se junta una técnica muy versátil con un necesidad, ampliamente extendida, de recrear y analizar sistemas de alta complejidad, los cuales escapan a los clásicos modelos lineales o analíticos.

### 2.2.1. Historia de las Simulaciones basadas en agentes

Entre las décadas de los 50 y los 70, trabajos como la máquina auto-replicante (Neumann, 1966) , de Von Neumann, o como el Juego de la Vida (Gardner, 1970), de John Conway, dieron cabida y avance al concepto de Autómata celular, siendo esta una de las primeras formas de Simulación basada en agentes.

Poco tiempo después vemos varios modelos que empleaban Simulaciones basadas en agentes para estudiar diversidad de fenómenos. Tenemos ejemplos en la política (Schelling, 1971), la Teoría de Juegos (mediante un torneo organizado por Robert Axelrod para evaluar distintas estrategias del Dilema del Prisionero), o la ecología (Hogeweg y Hesper, 1983).

En los 90 surge una expansión notable en el interés por estas técnicas. Esto se ve reflejado en diversos trabajos, de notable envergadura, llevados a cabo por varias universidades e investigadores provenientes de multitud de disciplinas, como ejemplos tenemos Epstein y Axtell (1996) de Sugarscape, *Computational Analysis of Social and Organizational Systems* (CASOS)<sup>2</sup> de la Carnegie Mellon University, o Kohler y Gumerman (2000) del Santa Fe Institute entre otros.

También surgieron revistas enfocadas a las sociedades virtuales y sus simulaciones, como la Journal of Artificial Societies and Social Simulations (JASSS) o la Complex Adaptive Systems Modeling (CASM) de Springer.

La efectividad de estos métodos acabó saltando fuera del ámbito de la investigación y encontrando un hueco en la sociedad. Lo podemos ver reflejado en el surgimiento de varias organizaciones, durante la misma década, centradas en el estudio y exploración de Simulaciones sociales y técnicas de organización efectiva - NAACSOS<sup>3</sup>, ESSA<sup>4</sup>, PAAAESSA<sup>5</sup>- a través de Simulaciones basadas en Agentes.

Con esta breve revisión queda claro el amplio rango de aplicación de estos modelos, además del gran interés que suscitan por la misma razón.

### 2.2.2. Principios de las Simulaciones basadas en agentes

Las simulaciones basadas en agentes se basan principalmente en el modelo computacional de los Sistemas Multiagente. Este tipo de sistemas forman un nuevo paradigma de computación enfocado en resolver tareas que serían imposibles de resolver para un sistema centralizado, bien por la complejidad de diseñar e implementar soluciones centralizadas, o bien por el pobre rendimiento que estas obtendrían.

Aparte de la naturaleza descentralizada del problema, se encuentra el otro componente central en este paradigma, los agentes. En este tipo de sistemas son los Agentes y las interacciones que hay entre ellos y su entorno los que dan lugar a las dinámicas que hacen de es interés este paradigma.

Es en la conjunción de estos dos conceptos donde se desvelan los elementos centrales de un Sistema multiagente, estos son, los Protocolos de comunicación entre Agentes, los protocolos de interacción entre Agentes y Entorno y el Entorno.

---

<sup>2</sup><http://www.casos.cs.cmu.edu/>

<sup>3</sup><http://www.casos.cs.cmu.edu/naacsos/index.php>

<sup>4</sup><http://www.essa.eu.org/>

<sup>5</sup><http://www.paaa.asia/#:~:text=Pacific%2DAsian%20Association%20for%20Agent,European%20Social%20Simulation%20Association%2C%20ESSA>

Estas características se van a explicar basándonos en un estudio de sistemas multiagente, en el que se discute un enfoque moderno para el diseño de un sistema de IA distribuido (Weiss, 1999)

### 2.2.2.1. El entorno

El entorno se puede pensar como la infraestructura que especifica los distintos protocolos de comunicación e interacción entre agentes y entorno. Permitiendo siempre la interacción, en mayor o menor medida, de los Agentes con "lo que les rodea". Otros aspectos clave en el diseño del entorno son la falta de centralización del entorno y que estos están poblados por Agentes.

Además de esas consideraciones están otras que surgen a la hora de diseñarlo, teniendo en cuenta que estas especificaciones engloban cualquier tipo de sistema multiagente.

- **Infraestructura de comunicación** En la que se tiene en cuenta el tipo de comunicaciones que se puede dar entre los agentes, dando la posibilidad de que esta sea Broadcast o Unicast, Memoria compartida o mensajes a través de la red, comunicación sincrónica o asíncrona, etc...
- **Protocolos de mensaje** Si la comunicación se da entre agentes en distintas redes habrá que considerar el tipo de protocolos de red que utilizarán para la comunicación (HTTP, HTML...).
- **Servicios de seguridad** Para garantizar una participación de agentes legítimos en el sistema.
- **Soporte de operaciones** Que garantice una consistencia en las interacciones que se den en el sistema.

Además de estas características propias del entorno, están las relativas al entorno-agente:

- **Percepción del entorno** Cuánto sabe el agente sobre el entorno.
- **Predecible** Cómo de predecible es el entorno para el agente.
- **Controlable** Cuán modificable es el entorno por el agente.
- **Dependiente del pasado** Cuánta dependencia tienen los futuros estados del sistema de los estados previos.
- **Naturaleza Teleológica de las partes** Qué capacidad de acción tienen las otras partes del sistema desde el punto de vista del agente.
- **A tiempo real** El entorno es cambiante durante el proceso de toma de decisión del agente.

### 2.2.2.2. Protocolos de comunicación entre agentes

Los agentes capaces de realizar esta acción son entes activos, capaces de percibir, razonar y actuar. Entre las acciones disponibles de un agente asumimos que se encuentra la de comunicarse con otros agentes, aunque también tiene su parte de percepción (al ser capaz de recibir mensajes de otros agentes para que exista una verdadera comunicación).

El objetivo de la comunicación entre Agentes es conseguir mejores resultados para ellos o para el sistema/sociedad de agentes (según si el Agente es cooperativo o no). Un resultado favorable de que haya comunicación entre Agentes es que da la posibilidad de que haya coordinación en el comportamiento y sus acciones, dando lugar a sistemas más coherentes. Esta coherencia se puede interpretar como la capacidad del sistema de actuar como una unidad, lograr esto a través de Agentes independientes es algo difícil de lograr sin ningún tipo de comportamiento centralizado.

A la hora de tratar de establecer la forma de comunicación entre Agentes surgen complicaciones a las que prestar atención, como pueden ser las siguientes:

- **Mensaje Descriptivo vs Prescriptivo** Habiendo una diferencia entre mensajes que describen hechos y mensajes que describen acciones o comportamientos, siendo el primero de estos igual de esencial que el segundo, pero menos adoptado por la dificultad que tiene hacer que una Agente sea capaz de emitir dichos mensajes.
- **Significado Personal vs Convencional** También se ha de tener en cuenta la diferencia de significado de un mensjae que puede haber entre el que lo emite y quien lo recibe.
- **Contexto** Igual de importante que el contenido de un mensaje e ssu contexto. Debiendo tener en cuenta el estado del agente receptor y el del entorno.
- **Alcance del lenguaje** Compromiso entre expresividad del lenguaje y complejidad a la hora de manejarlo.
- **Identidad** Se podría considerar parte del **Contexto** del lenguaje, tiene en cuenta los roles y partes involucradas en el mensaje.

Otros factores a tener en cuenta a la hora de especificar los protocolos de comunicación entre Agentes son el tipo de mensajes (dependiente del tipo de Agente emisor), los niveles de comunicación, intención de la comunicación, etc...

No los veremos en detalle debido a la extensión del campo y a que en el sistema multiagente relativo al trabajo en que nos basamos (Park et al. (2023)) los protocolos de comunicación están englobados dentro de la arquitectura de Agente propuesta. Además de que varios de los puntos a tener en cuenta son relegados a la hora de

adoptar las respuestas del LLM subyacente como los directores de acción de los Agentes.

### 2.2.2.3. Protocolos de interacción entre agentes

Los protocolos de interacción se encargan de llevar a cabo el intercambio de una serie de mensajes entre Agentes -Una conversación-. Se puede hacer una distinción entre los tipos de protocolos en base a si los Agentes actúan por objetivos individuales o comunes. En el caso de los objetivos individuales lo único que se pretende con los protocolos de interacción es maximizar las "ganancias". Al contrario, al haber Agentes con objetivos comunes, los protocolos de interacción tratan de lograr una coherencia global sin afectar a la autonomía de los Agentes y sin un control global explícito. Para lograr esto se tienen en cuenta varios aspectos como determinar objetivos o tareas comunes, evitar conflictos innecesarios o reunir conocimiento y pruebas.

### 2.2.3. Uso reciente de Simulaciones basadas en agentes

Como ya hemos visto en el apartado anterior, el rango de aplicaciones que este tipo de métodos tiene es amplio. En este apartado mencionaremos, superficialmente, avances significativos en distintos campos, a través del uso de esta técnica. Nos centraremos en los campos más relacionados al trabajo en el que nos basamos en este documento Park et al. (2023).

- **Estudios de Marketing** En este ámbito tenemos como ejemplo Rand y Rust (2011), el cual propone líneas generales de desarrollo de Simulaciones basadas en Agentes que cumplan con ciertos estándares ampliamente aceptados en ese ámbito en concreto, cumpliendo con la necesidad de establecer una forma común de proceder al hacer uso de las Simulaciones basadas en Agentes.

Así vemos, con este trabajo, que en el ámbito del marketing ya se ha buscado la aplicación de este tipo de técnicas.

- **Psicología Organizacional** El trabajo Hughes et al. (2012) nos muestra las ventajas y límites que las Simulaciones basadas en Agentes tendría al ser aplicada en la Psicología organizacional, sobre todo en entornos de alto riesgo o investigaciones que conlleven problemas prácticos o éticos.

Esta ventaja se ve aún más potenciada al ser alimentada con comportamientos humanos creíbles. Característica que buscan y evalúan en Park et al. (2023).

- **Gestión de equipos** En este ámbito, la exploración de distintas tomas de decisiones o entornos se vuelve crucial a la hora de decidir cómo organizar un equipo de trabajo. Al ser esta tarea ubicua en cualquier tipo de proyecto han sido múltiples los trabajos que buscan desarrollar o aportar nuevas herramientas a la hora de lidiar con este tipo de situaciones, entre los más destacados

encontramos a Crowder et al. (2012), Boroomand y Smaldino (2021) y Boroomand y Smaldino (2023), siendo estos dos últimos publicaciones de la revista JASSS, que exploran distintos escenarios y formas de interactuar entre los miembros de equipos que buscan soluciones a diversos problemas, valiéndose del estudio que realizan a una Simulación basada en agentes.

- **Redes sociales** En este ámbito hay multitud de fenómenos que son de gran interés, entre estos tenemos el de mecanismo de difusión de la información. Cuyo entendimiento nos ofrecería formas de proceder al lidiar con problemas como las Fake News. En este tipo de problemas también vemos un buen desempeño por parte de las Simulaciones Basadas en Agentes.

Como ejemplo tenemos Nasrinpour et al. (2016), donde se propone una Simulación basada en agentes a gran escala con la intención de estudiar como se comportaba el fenómeno de la difusión de la información en base al tipo de usuarios modelados, tomando como referencia la red social Facebook.

- **Economía** Tras la crisis de 2008, se han buscado modelos que permitan un mejor y mayor nivel de análisis de los mercados financieros Bookstaber (2017), Farmer y Foley (2009). Con este objetivo, las Simulaciones Basadas en agentes ofrecen una forma de diseñar experimentos desde un enfoque bottom-up. Vemos además que el enfoque que adoptan este tipo de simulaciones permite la aparición de dinámicas muy complejas a nivel de sistema. Esto se refleja en trabajos como Tesfatsion y Judd (2006) y Agents of Change<sup>6</sup>. En los cuales se explora la validez de este tipo de modelos a la hora de analizar cuestiones tan complejas y dinámicas como la economía.

## 2.3. Sistemas multi-agente para la simulación social

Después de comentar la historia, principios y uso reciente de las simulaciones basadas en agentes en la sección anterior, esta sección se centrará en profundizar en los sistemas multi-agente (SMA) para la simulación social específicamente. Estos son una clase de sistemas que simulan el comportamiento de agentes de Inteligencia Artificial para replicar comportamientos sociales y estudiar los mismos.

### 2.3.1. Historia de los SMA para la simulación social

El uso de los SMA para simulaciones sociales se remonta a la década de 1990, con la creación del modelo Sugarscape, el cual se utilizó para investigar fenómenos sociales como la contaminación o la migración, entre otros (Axtell et al., 1996).

A lo largo de los años ha sido un tema interesante de estudio, pero muchas veces limitado por la tecnología, ya que no existían unos modelos capaces de realizar

<sup>6</sup><https://web.archive.org/web/20110123110948/http://www.economist.com/node/16636121>

simulaciones verosímiles con las que se pudiesen estudiar relaciones sociales creíbles. Un ejemplo de un estudio de esta índole es el realizado por varios profesores de la UCM en 2006, en el que se exploraba la investigación de SMA para sistemas sociales con agentes software, empleando el lenguaje de modelado INGENIAS y donde lo extienden para poder realizar simulaciones sociales, permitiendo así llevar a cabo el estudio de la religiosidad de la sociedad española que tiene lugar en el artículo (Arroyo Menéndez et al., 2006).

A día de hoy, con los avances que ha habido en el campo de la Inteligencia Artificial y con la proliferación de sistemas de IA generativa, es posible realizar SMAs verosímiles y que tengan un comportamiento humano, como se ha visto en el caso de *Generative Agents* (Park et al., 2023), el estudio en el que nos basamos para este trabajo.

### 2.3.2. Ejemplos de aplicaciones de los SMA en simulaciones sociales

Además de los vistos en el apartado anterior, hay algunas otras áreas en las que se han implementado estas simulaciones exitosamente, como son las siguientes:

Estudios sobre la difusión de información en redes sociales, como es el caso de uno particular, realizado en la plataforma de Facebook. En este estudio, se investiga sobre cómo la información (o desinformación) se propaga entre usuarios y qué factores pueden acelerar o frenar esta difusión (Gambo et al., 2020).

Investigación del comportamiento de multitudes. Como se mencionó anteriormente, los SMA se pueden utilizar para simular el comportamiento de las masas ante desastres naturales como terremotos o inundaciones, entre otros, para así ver cuál es la opción óptima de evacuación (Marks, 2012).

Análisis de mercados financieros, analizando cómo los gentes interactúan en la compra y venta de activos financieros, lo que ayudaría a prever posibles crisis financieras y diseñar políticas regulatorias más efectivas (Marks, 2012).

## 2.4. Modelos de lenguaje

La búsqueda y desarrollo de Modelos de Lenguaje (LMs por sus siglas en inglés) se fundamenta en la necesidad de superar las barreras entre la comunicación humana y la interacción con las máquinas, buscando una integración más natural y eficiente en nuestra vida cotidiana y profesional. Esta búsqueda se motiva por el deseo de automatizar y optimizar tareas que requieren el procesamiento del lenguaje natural, desde la traducción y generación de texto hasta la asistencia personalizada y la gestión de información.

Al mismo tiempo, los LMs prometen romper las barreras lingüísticas y mejorar la accesibilidad, permitiendo que información y servicios sean más inclusivos y estén al alcance de una audiencia global más amplia. En el ámbito académico y profesional, el desarrollo de estos modelos impulsa la innovación en inteligencia artificial, abriendo nuevas líneas de investigación y aplicaciones prácticas en sectores tan variados como la salud, la educación, el derecho y el entretenimiento.

En esta sección pretendemos dar un breve contexto y explicación de los Modelos de Lenguaje, que son el impulsor de todo este trabajo.

Comenzaremos explicando brevemente cómo se desarrolló inicialmente el campo de los Modelos de Lenguaje. Dando unas breves pinceladas a los modelos simbólicos, detallando un poco los modelos estadísticos más extendidos y, que a día de hoy, siguen realizando variedad de tareas y finalmente a los modelos neuronales, que son los que potencian todos los sistemas conversacionales exitosos que vemos a día de hoy.

### 2.4.1. Modelos simbólicos

Este enfoque fue el primero en ser adoptado. En base a reglas que se plasmaban en el sistema, las cuales se obtenían a partir de expertos en el campo de aplicación, se acababa construyendo un sistema capaz de tomar decisiones con la información con que se había nutrido al sistema.

Basados en este enfoque tenemos diversas corrientes que le dieron uso, entre otros se encuentran los siguientes:

- **Sistemas expertos** Este tipo de sistemas busca resolver problemas a través de sistemas a los que se les ha especificado una serie muy concreta de reglas con las que operar. Su uso se extendió a inicios de los 70 en cantidad de sistemas de diagnóstico<sup>7</sup>. En el campo del lenguaje también hubieron trabajos que trataron de demostrar su aplicabilidad, un ejemplo de esto lo tenemos en el estudio realizado por Winograd (1971)
- **Gramáticas generativas** Este concepto es central en lingüística y propone una forma muy estructurada de definir un lenguaje, en concreto su sintaxis. Este modelo en concreto fue presentado en Chomsky (1957) por Chomsky y se ha utilizado en sistemas como *Forecast Generator* descrito en el artículo de Goldberg et al. (1994)
- **Traducción automática** En un inicio también se puso especial énfasis en la traducción automática de textos, viendo ejemplos en proyectos como en el

---

<sup>7</sup><https://pubmed.ncbi.nlm.nih.gov/2191729/>

<sup>8</sup>[https://www.researchgate.net/publication/43763822\\_An\\_Expert\\_System\\_for\\_Diagnosis\\_Of\\_Human\\_Diseases](https://www.researchgate.net/publication/43763822_An_Expert_System_for_Diagnosis_Of_Human_Diseases)

*Georgetown-IBM experiment*<sup>9</sup> o *SYSTRAN*<sup>10</sup>

Otra información significativa en este enfoque fue la ofrecida por Reiter y Dale (1997), donde se trata de lograr algún consenso en la manera de construir los Modelos de lenguaje Simbólicos. La arquitectura propuesta era modular y fácilmente interpretable.

Finalmente este tipo de enfoques fue abandonado en el campo del Modelado del Lenguaje debido a la poca escalabilidad y flexibilidad -limitadas por la especificación de unas reglas muy concretas-, dificultad para manejar la ambigüedad y variabilidad del Lenguaje Natural y a la aparición de métodos muchos más efectivos y escalables como los Estadísticos o Neuronales.

## 2.4.2. Modelos estadísticos

Debido al creciente aumento de información tras la aparición de tecnologías como Internet se tuvo la necesidad de métodos más eficientes y efectivos que fueran capaces de procesar información escrita. Con este objetivo se empezaron a desarrollar e implantar modelos estadísticos para acometer esta tarea. Estos, por su naturaleza probabilística, eran capaces de hacer frente a la ambigüedad presente en sistemas como el Lenguaje Natural. A día de hoy estos modelos siguen siendo utilizados para numerosas tareas, como la clasificación de textos. A continuación explicaremos brevemente los Modelos estadísticos más extendidos desde los enfoques principales a la hora de abordar los problemas a los que se enfrentaban.

Los modelos estadísticos surgen como una de las primeras aproximaciones, y la más sencilla, para modelar un Lenguaje Natural.

Nos centraremos aquí en los modelos de N-Gramas. El modelo en sí consiste en la distribución de probabilidad asociada a una secuencia de N elementos. Estos elementos pueden ser indistintamente caracteres, sílabas o palabras, según la tarea para la que se emplee el modelo.

Este modelo se basa en las cadenas de Markov para definir el concepto de N-Grama.

### 2.4.2.1. Cadenas de Markov

Las cadenas de Markov, propuestas en la investigación de Markov (1906), son un tipo de modelo matemático utilizado para predecir la probabilidad de ciertos eventos basándose en el conocimiento del estado actual, sin necesidad de considerar

---

<sup>9</sup><https://web.archive.org/web/20071021224529/http://www.hutchinsweb.me.uk/GU-IBM-2005.pdf>

<sup>10</sup><https://en.wikipedia.org/wiki/SYSTRAN>

la historia completa. En pocas palabras, este modelo asume que el futuro es independiente del pasado, al menos en su totalidad. A esta propiedad se la conoce como Propiedad de Markov y supone un problema, *La falta de memoria*.

#### 2.4.2.2. N-Grama

Estos modelos fueron propuestos por Markov (1913) Una N-Grama se define como una Cadena de Markov de orden n-1. Y por como se ha definido una Cadena de markov tenemos que el elemento

$$n_i$$

depende únicamente de los n-1 elementos anteriores.

Como ejemplo si tenemos un modelo de Bigrama (2-Grama)

$$P(C_i|C_{1:i-1}) = P(C_i|C_{i-2}:C_{i-1})$$

Y de forma general tenemos que la probabilidad de una secuencia de elementos  $C_{1:N}$  es

$$P(C_{1:N}) = \prod_{i=1}^N P(c_i|c_{1:i-1}) = \prod_{i=1}^N P(c_i|c_{i-2:i-1})$$

También se puede emplear otro tipo de elementos fundamentales, por ejemplo las palabras. Pero esto puede conllevar problemas al aparecer palabras que no existieran en el corpus de entrenamiento, esta problemática no era posible al considerar los caracteres como los elementos de las N-Gramas.

Para esto se puede emplear nuevos símbolos que representen una categoría de palabras, como ejemplo tenemos una dirección de correo o un número de teléfono. Considerando a todos estos como pertenecientes a la misma categoría *número de teléfono* o *email*.

#### 2.4.2.3. Aplicaciones de las N-Gramas

Estos modelos tienen múltiples aplicaciones a día de hoy, entre ellas destacan las siguientes:

- **Clasificación de textos** Con el sencillo modelo de N-Gramas que acabamos de presentar es posible ocuparnos de este tipo de tareas. Típicos ejemplos de clasificación son el de Spam, Análisis de sentimientos, Idioma al que pertenece el texto...

Sin embargo esta aproximación empieza a verse limitada de forma rápida. Por ejemplos los modelos de 1-gramas pierde directamente la noción del orden en las palabras, clasificando de la misma forma a cualquier permutación de un

texto, esta noción de orden se recupera parcialmente en n-gramas de orden superior, pero siempre hasta cierto límite. Vemos reflejado aquí el problema de la *Falta de Memoria* propio de los sistemas basados en Cadenas de Markov. Vemos así limitaciones intrínsecas a este tipo de modelos.

- **Recuperación de la información** En esta tarea los modelos de N-gramas es ampliamente usado, por ejemplo en detección de plagio o clasificación de textos (Sood et al., 2022).

### 2.4.3. Modelos neuronales

El último de los enfoques que vamos a presentar es el de los Modelos basados en Redes Neuronales. Recorreremos los avances que han afectado a las arquitecturas más extendidas a día de hoy en el campo de los Modelos de Lenguaje.

A principios del milenio se publicó el trabajo Bengio et al. (2000) que se considera fundacional en el campo de la creación de Modelos de Lenguaje modernos. Este fue el siguiente gran paso que se dió avanzando desde los modelos estadísticos basados en N-Gramas, hacia las Arquitecturas Neuronales modernas.

Pasamos a introducir brevemente las principales Arquitecturas Neuronales a las que dio pie el trabajo que acabamos de citar.

#### 2.4.3.1. Redes Neuronales Recurrentes

Este tipo de Red Neuronal fue propuesta inicialmente en un estudio en 1982 (Hopfield, 1982). Se centra en resolver problemas que involucran datos secuenciales, ejemplo de este tipo de problemas es la traducción automática (Sutskever et al., 2014), el Procesamiento del Lenguaje Natural, la descripción de imágenes (Karpathy y Fei-Fei, 2015) o el reconocimiento de voz (Hannun et al., 2014).

El elemento fundamental de la red es la neurona recurrente, con funciones de activación en ambos sentidos. Estas neuronas se organizan en capas ocultas, de salida y de entrada. Las capas de entrada y salida se ocupan de recibir la información y mostrar los resultados respectivamente. El análisis de la información y los datos se realiza en las capas ocultas.

Este tipo de redes se encarga de procesar la información de forma secuencial, es decir, la información pasa capa por capa en el mismo orden en que fue recibida, como se puede apreciar en la Figura 2.1<sup>11</sup>. En las capas ocultas las neuronas se nutren de la información que le llega por la entrada, así como la información que había recibido previamente, consiguiendo así una cierta memoria, que utiliza, junto a la información de entrada, para predecir los resultados.

<sup>11</sup>Tomada de [https://en.wikipedia.org/wiki/Recurrent\\_neural\\_network](https://en.wikipedia.org/wiki/Recurrent_neural_network)

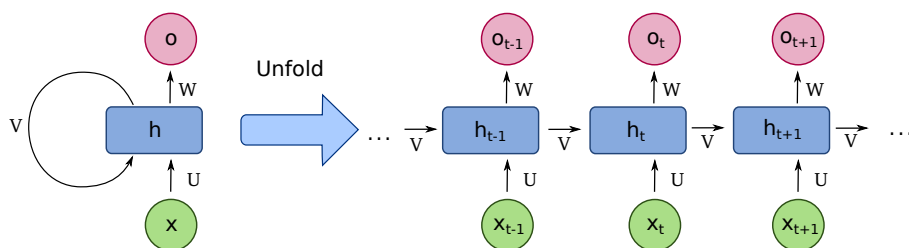


Figura 2.1: Estructura de una neurona de una RNN

Este tipo de redes tienen ciertos problemas, entre los que destaca el *Vanishing/Exploding Gradients Problem*, identificado por Hochreiter (1991). Este problema dificulta el proceso de entrenamiento de las Redes Neuronales Recurrentes, haciendo que la red llegue a un estado de subajuste o que directamente no sea capaz de generalizar la información. El problema surge al acumular el resultado de los inputs previos, escalado con el peso correspondiente a la neurona, haciendo que en los casos en que el peso es menor que uno, la aportación de los estados previos resulte insignificante para los actuales y posteriores. Un problema análogo ocurre con los pesos mayores que uno.

Posteriormente surgen nuevos trabajos con el propósito de solucionar estos problemas. Entre estos tenemos como ejemplo el Long short-term Memory (LSTM)

### 2.4.3.2. LSTM

Este modelo (Long Short-term Memory) (Hochreiter y Schmidhuber, 1997) es una evolución de las Redes Neuronales Recurrentes mencionadas anteriormente, que pretende solucionar el *Vanishing/Exploding Gradient Problem* que es un desafío común en el entrenamiento de redes neuronales profundas. Se refiere a la tendencia de los gradientes (valores utilizados para actualizar los pesos de la red durante el entrenamiento) a volverse extremadamente pequeños (evanescentes) o extremadamente grandes (explosivos) a medida que se propagan hacia atrás a través de las capas de la red (Goodfellow et al., 2016).

La solución que proponen tiene una mayor capacidad de recordar datos antiguos y se basa en ir decidiendo qué información olvidar a la vez que se considera al mismo tiempo el cúmulo de aprendizaje a corto y largo plazo.

Las celdas de esta arquitectura están conformadas por cuatro tipos de puerta, como se puede apreciar en la ilustración representada en la Figura 2.2<sup>12</sup>

- **Forget Gate** Se encarga de reajustar la información proveniente de estados

<sup>12</sup>Tomada de [https://d21.ai/chapter\\_recurrent-modern/lstm.html](https://d21.ai/chapter_recurrent-modern/lstm.html)

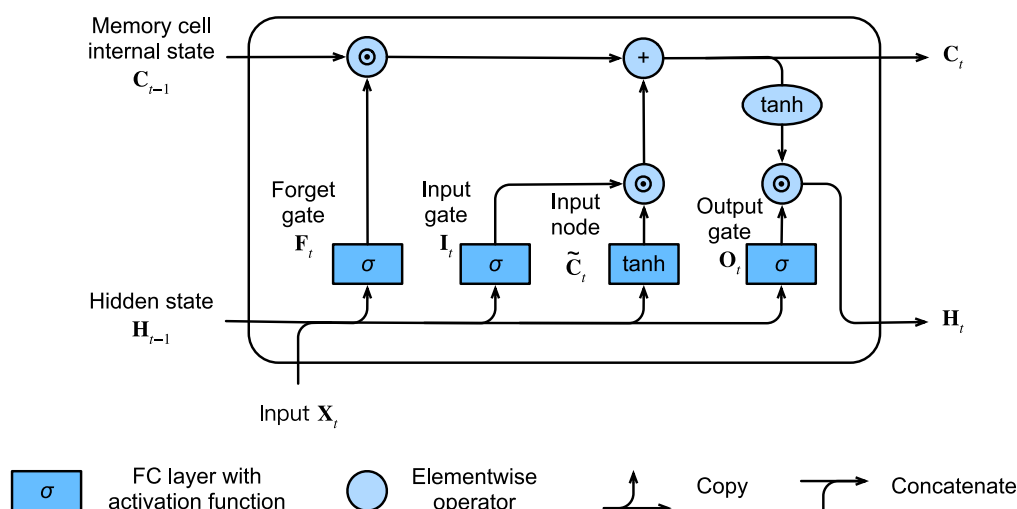


Figura 2.2: Estructura de una celda LSTM

antiguos en base al input recibido para decidir cuanta de esta seguirá teniendo en cuenta.

- **Cell state** Es el componente encargado de transportar la información a lo largo del tiempo.
- **Input Gate** Se encarga de decidir en qué medida influirá el input actual a los posteriores estados del sistema (memoria a largo plazo)
- **Output gate** Consigue el resultado final en base a la nueva memoria a largo plazo reajustada en combinación con el input y la memoria a corto plazo. Este resultado será la nueva memoria a corto plazo.

### 2.4.3.3. Seq2Seq

Fue propuesto en el estudio de Sutskever et al. (2014), con el propósito de resolver las dificultades que los modelos estadísticos presentaban a la hora de enfrentarse con la tarea de la traducción, concretamente la traducción de una secuencia de entrada a una secuencia de salida (Sequence 2 Sequence). Para lograr este propósito hacía uso del modelo presentado anteriormente, LSTM, junto a la arquitectura Encoder-Decoder.

Este modelo se volvió relevante gracias a su mayor rendimiento, en comparación a los modelos estadísticos precedentes, en la tarea de la traducción de idiomas. Una de las razones por las que muestra un mayor desempeño, es que toma en consideración toda la secuencia de entrada para dar cualquier tipo de output. De esta forma es capaz de dar respuestas más coherentes con todo el contexto de la entrada.

La mayor novedad en este modelo es el uso de la arquitectura Encoder-Decoder.

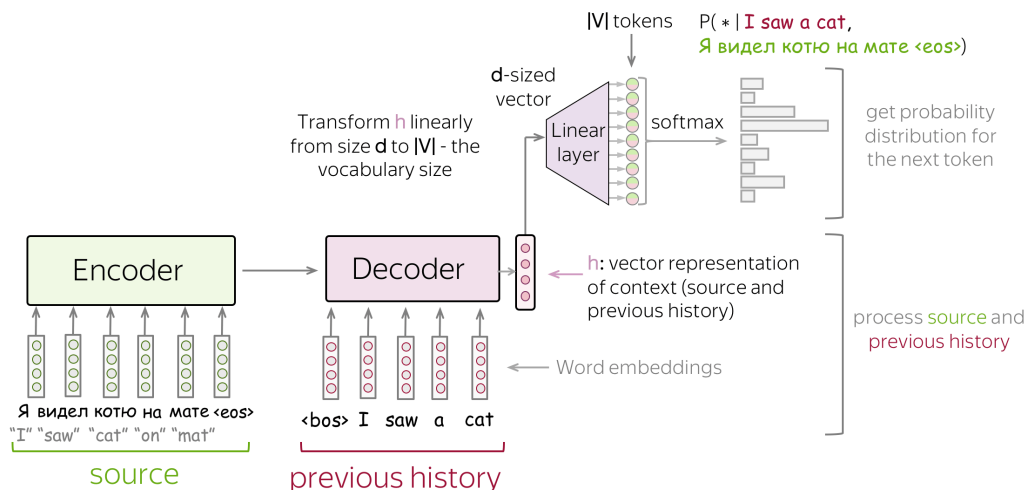


Figura 2.3: Estructura de una arquitectura seq2seq

Continuamos con una breve explicación de los componentes principales de esta arquitectura, acompañados de una ilustración en la Figura 2.3<sup>13</sup>:

- Encoder** Se encarga de tomar una secuencia de entrada y la transforma en una representación oculta de alta dimensión (*Word Embeddings*), mediante el uso de una *Embedding layer*, con esta representación se trata de capturar el contenido semántico de la oración recibida. A menudo los LSTMs son empleados como Encoders debido a su gran rendimiento al momento de manejar secuencias de datos largas y su capacidad de retentiva a lo largo del tiempo. En el estudio original (Sutskever et al., 2014) en que se propuso el modelo seq2seq vemos que emplean 4 capas de 1000 celdas LSTMs cada una. Dejando en cada celda su propio conjunto independiente de pesos y sesgos. Y usando cada capa el Output de la capa anterior como Input.

Todo esto con el objetivo de producir el llamado *Vector de Contexto*, que se usará como inicialización de memorias de corto y largo plazo para las capas de salida del decoder.

- Decoder** Este se encarga de utilizar el *Vector de Contexto* como memoria inicial tanto en el largo como corto plazo de sus capas de celdas LSTMs (en un estudio de 2014 llevado a cabo investigando esto, emplean una red de LSTMs similar a la del decoder (Sutskever et al., 2014)).

Para interpretar la información de salida de la última capa de celdas LSTMs, el Decoder utiliza un *Word Embedding* correspondiente al *lenguaje* de salida. Este obtiene su valor a partir de encadenar a la salida de la última capa LSTM una *Fully Connected Layer* seguida de una función *SoftMax* que se encargará de darnos finalmente el Output.

<sup>13</sup>Imagen tomada de [https://lena-voita.github.io/nlp\\_course/seq2seq\\_and\\_attention.html](https://lena-voita.github.io/nlp_course/seq2seq_and_attention.html)

Este modelo soluciona varios de los problemas que surgían en modelos anteriores en el campo de la traducción, sin embargo se encuentra con sus propias limitaciones. Entre estas, la más importante, es el problema del *Cuello de botella* que se ocasiona al tener que codificar todo el Input a través del Encoder para así obtener el *Vector de Contexto* con el que inicializamos al Decoder. Otro de los problemas con los que se encuentra este modelo es el limitado contexto al que puede atender en la secuencia de entrada, esto se debe al hecho de que toda la secuencia de entrada se ha de representar mediante el *Vector de Contexto*, llegando a perder mucha información en el caso de secuencias de entrada largas debido al tamaño fijo del *Vector de Contexto*.

Poco después surge el trabajo Bahdanau et al. (2014) que se centra en introducir los Mecanismos de atención, aliviando el problema de la limitación en el tamaño de la secuencia de entrada.

Este mecanismo consigue una mejor selección de la información tenida en cuenta por el *Decoder* a la hora de decodificar la información del *Encoder*. Esto lo hace a través de añadir, al *Decoder*, mecanismos capaces de deducir la similitud entre las respuestas ofrecidas por el *Encoder* y el *Decoder*.

### 2.4.3.4. Transformers

Uno de los puntos de inflexión llega con el trabajo de Vaswani et al. (2017) presentado por investigadores de Google, que introduce multitud de conceptos, que, en conjunto, son capaces de ofrecer un modelo superior en rendimiento y desempeño a los anteriores. Una característica principal es el nuevo enfoque que da, dejando de ser un nuevo paso en el camino trazado por los modelos explicados anteriormente, y proponiendo un modelo que pone la tilde en el mecanismo de atención y construyendo, a partir de este, un sistema capaz, de forma intrínseca, de atender a las relaciones semánticas entre los componentes de las secuencias tratadas (e.g. las palabras de un texto), además de ofrecer la posibilidad de realizar todas las tareas de entrenamiento e inferencia de una forma paralela, abriendo así la posibilidad a un aumento en la escala de este tipo de modelos como no se había visto y en el que aún no terminamos de ver el progreso. Esto se puede apreciar en la ilustración que se muestra en la Figura 2.4

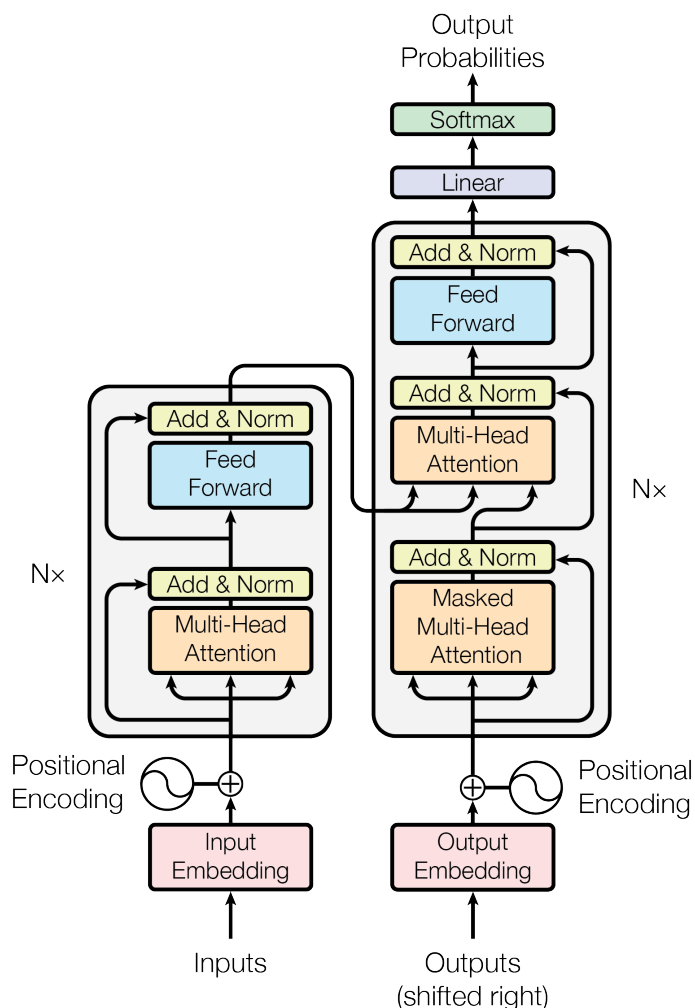


Figura 2.4: Arquitectura Transformer (Imagen tomada de Vaswani et al. (2017))

Entre los principales componentes con los que cuenta el modelo propuesto tenemos:

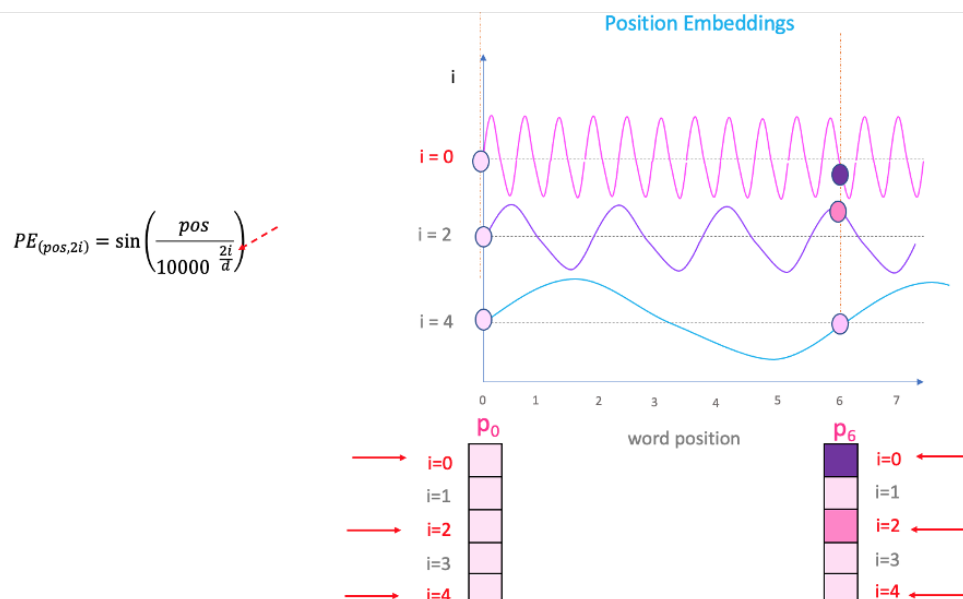


Figura 2.5: Visualización del mecanismo *Positional Encoding*

- Positional Encoding** Como ya habíamos visto en varios modelos anteriores, a la hora de tratar las secuencias de entrada estas eran convertidas en Tokens correspondientes a un *Word Embedding* con la intención de atrapar el significado semántico.

En este trabajo se añaden mecanismos para atrapar, a la vez que el significado semántico, la relación de orden de aparición entre los Tokens de la secuencia de entrada. Esto lo consiguen mediante la adición de los valores de funciones seno y coseno de frecuencias variando en función de la posición del Token y del tamaño del *Word Embedding* permitiendo, además, atrapar esta relación de posición para cualquier longitud de cadena de entrada.

$$PE_{pos,2i} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{pos,2i+1} = \cos(pos/10000^{2i/d_{model}})$$

Podemos ver cómo actúa el mecanismo mediante la Figura 2.5<sup>14</sup>, donde se ilustra en qué forma varía la codificación en relación a la posición del Token.

- Self Attention** Busca la similitud entre una palabra con todas las demás, y con ella misma. Actúa sobre la secuencia de entrada para asignar la importancia que cada una de estas tiene en la palabra en la que nos habíamos enfocado inicialmente (producto escalar). Para lograr esto introduce con cada palabra tres nuevos valores *Query*, *Key* y *Value*, con el objetivo de lograr una codificación para cada palabra resultado de su relación con las demás, que es el que usará como valor representativo del Token (ver figura 2.6).

<sup>14</sup><https://datascience.stackexchange.com/questions/51065/what-is-the-positional-encoding-in-the-transformer-model>

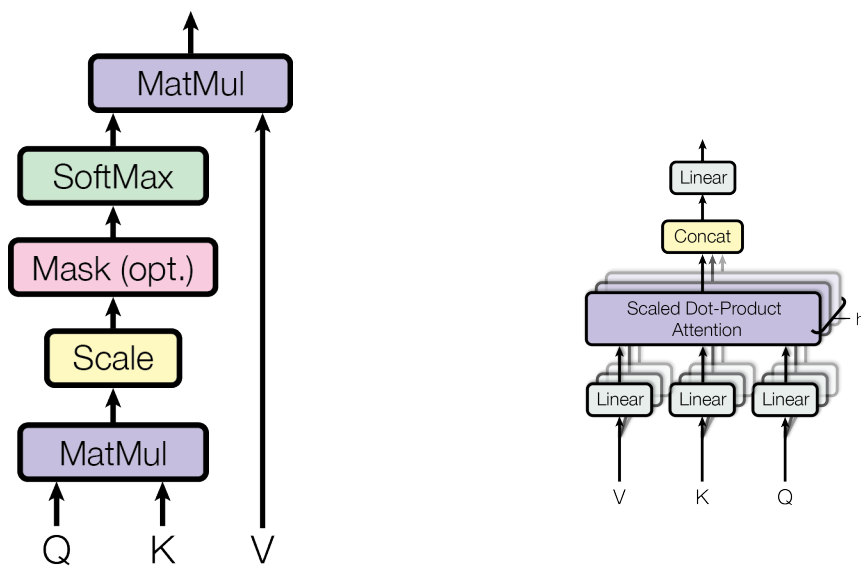


Figura 2.6: Ilustración del mecanismo de *Self-Attention*. Tomada de Vaswani et al. (2017)

Figura 2.7: Ilustración de múltiples capas de *Self-Attention*. Tomada de Vaswani et al. (2017)

El uso que se le da al valor *Query* es obtener a través de este y todos los demás valores *Key* una serie de valores con la que, al pasar por una función *Softmax*, obtenemos el peso que damos a cada uno de los valores *Value* de los otros Token, consiguiendo con la suma de todo esto el valor que esperábamos para la palabra de la que empleábamos el *Query*.

Una gran ventaja de este mecanismo es la paralelización que ofrece en la fase de codificación, ya que para el cálculo del valor de codificación no necesitas ningún resultado dependiente de las palabras anteriores. Eliminando el problema que tenían modelos anteriores en la obtención del *Vector de Contexto*.

La arquitectura puede tener múltiples capas de atención. Capturando cada una de ellas distintas relaciones entre las palabras y de forma completamente independiente y paralelizable (ver Figura 2.7).

Además permite añadir múltiples capas con pesos y desvios distintos, de forma que la arquitectura sea capaz de enfocarse en varios tipos de relaciones entre las palabras. En el trabajo presentado por Google propusieron un modelo con 8 capas.

Con este último modelo hemos hecho un repaso de la evolución que este campo ha sufrido. Partiendo de modelos formalizados hace más de 4 décadas hasta modelos propuestos hace menos de 5 años y que actualmente nutren las arquitecturas propuestas para llevar a cabo la construcción de sistemas tan presentes y capaces en nuestro día a día. Tenemos ejemplos claros en productos como ChatGPT o la competición que se está viendo en el campo de los LLM involucrando a empresas de primer calibre como Microsoft, Meta, Google u OpenAI, ofreciéndonos tanto modelos como asistentes de chat nuevos y más potentes cada pocos meses. Este tipo de

modelos es el que más desarrollos ha sufrido y todos ellos se basan en la arquitectura que acabamos de presentar.

#### 2.4.4. Grandes Modelos de Lenguaje

Como hemos visto en la sección anterior, una de las grandes ventajas de los Transformers fue la paralelización, abriendo el paso al desarrollo de modelos cada vez más grandes y versátiles. Esta versatilidad ha hecho buscar en este tipo de modelos infinidad de aplicaciones. Sin embargo, esta capacidad viene acompañada de una alta necesidad de computación para llevar a cabo un entrenamiento, haciendo muy restrictivo el acceso a dichos modelos. Es por ello que este tipo de tecnologías han sido abordadas por entidades con una gran cantidad de recursos computacionales, como ejemplo Google que introdujo la arquitectura BERT o GPT introducida por OpenAI. Aunque en ocasiones estos avances se compartían de forma abierta (Touvron et al. (2023)) y hay multitud de iniciativas que abogan por el desarrollo libre, la iniciativa privada a tomado una parte considerable en el desarrollo de estos modelos.

Llegando a ser estos últimos los modelos que más han irrumpido en la sociedad debido a sus capacidades.

##### 2.4.4.1. Modelos Generativos

En parte, el interés en estos sistemas ha venido de la mano con su capacidad para generar información valiosa a partir de un entrenamiento. Hasta avances como los Variational Autoencoders Kingma y Welling (2013) y las Redes GANs Goodfellow et al. (2014), los modelos eran mayoritariamente estadísticos. dando lugar principalmente a modelos discriminativos para datos más complejos como imágenes o lenguaje Natural. Ejemplos de estos podrían ser los analizadores de sentimientos o clasificadores de imágenes.

Con la llegada de estos avances, en combinación con modelos más desarrollados como los Transformers, empezaron a surgir los primeros Modelos generativos con resultados de un mayor nivel de calidad, entre estos se encuentran, por el lado del Lenguaje, Generative Pre-Trained transformer (Radford et al., 2018), GPT, donde vemos el primer Modelo del lenguaje creado por OpenAI y que, posteriormente, daría lugar a los sistemas tan asimilados de hoy en día como ChatGPT. Por el lado de la generación de imágenes también cabe destacar la aparición de sistemas como DALL-E o Stable Diffusion, ambos basados en Modelos de difusión , pero grandes ejemplos de las capacidades e interés que suscitan los Modelos Generativos.

#### 2.4.4.2. Principales LLMs

A continuación, comentaremos los principales Modelos de Lenguaje con que se cuenta a día de hoy (a pesar de que la vigencia de estos se vea rápidamente mermada en favor de nuevos y más avanzados modelos en muy poco tiempo).

##### **GPT 4**

Este es un modelo multimodal (capaz de procesar imágenes también) privado de la empresa OpenAI, basado en transformers y entrenado tanto con bases de datos públicas como con datos provistos por terceros. Además de la fase de entrenamiento, donde se estima que hubo alrededor de un billón (europeo) de parámetros que calcular, tiene una fase de aprendizaje reforzado con humanos en el que se consigue un fine-tuning del modelo con el objetivo de alinearlos a los intereses de la empresa y sus usuarios.

Los detalles técnicos y de implementación no son públicos, sin embargo se piensa que hacen uso de la técnica MoE. Donde se han entrenado múltiples redes, cada una experta en un ámbito concreto, y encargada de dar respuesta a consultas relativas a su ámbito, que en conjunto forman parte del sistema GPT4, permitiendo así segmentar los requisitos computacionales de la inferencia, obteniendo mejores tiempos de inferencia y menor uso de memoria.

##### **Llama**

Es el LLM creado por Meta. Totalmente abierto y en varios tamaños, también usa los transformers. La ventaja que trae este modelo es que se trató de conseguir un aumento en la eficacia de su eficacia aumentando la cantidad de datos de entrenamiento en lugar del número de parámetros. Consiguiendo así un rendimiento similar a modelos como GPT 3 con una décima parte de su tamaño, lo que permitía el acceso a los más avanzados LLMs incluso en un ámbito local.

Hasta el momento hay 3 versiones de este modelo, la primera de ellas sin ser accesible al público y las dos últimas siendo Open Source.

Llama 2 ofreció 3 tamaños, 6.7B, 13B y 69B. Siendo estos respectivamente de 6.7, 13 y 69 mil millones de parámetros. Llama 3 ha ofrecido hasta el momento 2 tamaños, 8B y 70.6B.

#### 2.4.4.3. Técnicas de optimización para Grandes Modelos de Lenguaje

Con el auge de esta tecnología también se ha visto el afloramiento de multitud de técnicas para aumentar la eficiencia de los modelos. Estas técnicas se enfocan en añadir mejoras en el proceso de entrenamiento, como en el paper 1-bit LLMs por Ma et al. (2024), o en modelos ya entrenados.

##### **Poda**

Fue una propuesta realizada en los 90 en el paper LeCun et al. (1989), donde explicaba los beneficios que suponían a una red de neuronas, tanto para el proceso de aprendizaje, como para aprovechar mejor los datos de entrenamiento y los recursos computacionales.

Este enfoque se puede adoptar tanto durante como tras el entrenamiento. Ambos enfoques traen la misma ventaja, optimizar el uso de parámetros de la red eliminando aquellos parámetros que no tienen relevancia en la red. Dando lugar a redes más compactas y más enfocadas en su tarea.

Al obtener redes más compactas, se reduce considerablemente el tiempo de inferencia.

### Quantization

Esta técnica se basa en reducir la cantidad de memoria necesaria para representar un parámetro. Surgiendo un trade-off en la precisión de los parámetros (y por tanto la información que son capaces de manejar correctamente) y el rendimiento de la red. Este método no afecta a la estructura del modelo, ya que solo atañe a la precisión de los parámetros en el mismo.

Suele llegar a reducir la información desde formatos de representación float de 32 bits hasta representación de 4bits, viendo un detrimento de la calidad en las respuestas con cada reducción de precisión. Pero ofreciendo un amplio abanico de opciones y bajando el umbral para el uso de estos sistemas.

### Modelos de 1.58 bits

Este último avance Ma et al. (2024), surgido este mismo febrero, nos ofrece la posibilidad de reducir al mínimo la cantidad de recursos necesarios (3 valores) para representar a los parámetros de un Modelo de IA. Reduciendo drásticamente el tiempo de inferencia, la energía consumida y la cantidad de memoria necesitada.

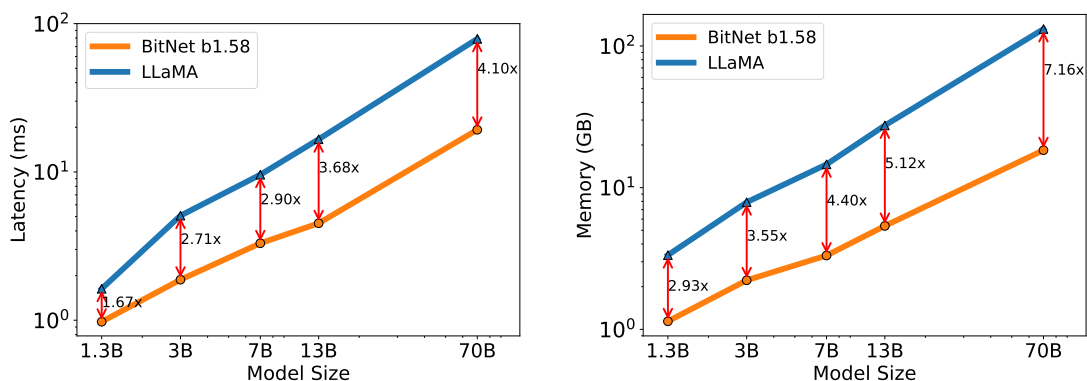


Figura 2.8: Tomada de Ma et al. (2024)

En el paper muestran los resultados de haber creado un modelo, BitNet b1.58, implementando su idea y comparandolo a modelos como Llama en vario tamaños

distintos, uno de los resultados más destacados de este avance lo mostramos en la figura 2.8.

#### 2.4.4.4. Técnicas de evaluación de LLMs

##### Perplexity

La perplejidad es un tipo de evaluación intrínseca, que se centra en evaluar directamente la calidad y precisión de las salidas del modelo. Se basa en indicar como de bueno es un modelo en predecir una palabra, una medida baja es un mejor rendimiento.

Esta evaluación viene de la teoría de la información y fue popularizada en el campo del lenguaje natural por trabajos como los estudios de Jelinek (1976) y Brown et al. (1990). Se basa en la idea de medir la incertidumbre en las predicciones del modelo.

$$PP = 2^{-\frac{1}{N} \sum_{i=1}^N \log_2 P(w_i)}$$

Actualmente, se utiliza de forma amplia, se puede ver como métrica usada en trabajos como "Improving Language Understanding by Generative Pre-Training" Radford et al. (2018).

## 2.5. Generative Agents: el estudio original del que se parte

Como se ha mencionado en la introducción, el desarrollo del presente trabajo parte inicialmente de otro estudio realizado previamente, en el que se emplea un sistema multiagente para la simulación social (los cuales serán tratados en la sección 2.3) y cuyo foco es demostrar que pueden emerger ciertos fenómenos sociales a partir de esta simulación (Park et al., 2023).

Este estudio logra demostrar de manera exitosa la generación de relaciones sociales y de recuerdos en la memoria, indicando así que mediante el diseño y arquitectura iniciales, podían estudiar el comportamiento humano en cierta medida.

En el presente capítulo, trataremos por un lado el diseño existente en el estudio de *Generative Agents* (Park et al., 2023), así como las funcionalidades que ejecutaba inicialmente la aplicación, para tener claro el funcionamiento inicial de nuestro sistema y qué era lo que ya había previamente implementado.

### 2.5.1. Diseño original del sistema

El sistema fue diseñado para los propósitos previamente mencionados, por lo que, como los objetivos principales de este trabajo se apoyan en el desarrollo realizado en este estudio, no hemos cambiado este diseño y esta arquitectura, ya que se ha demostrado que funciona. Lo que hemos realizado son adaptaciones y extensiones al programa previamente existente.

La arquitectura básica sobre la que se basa el artículo de *Generative Agents* se puede apreciar en la figura 2.9. En esta, se ve cómo la información y memorias que generan los usuarios se percibe por tres medios distintos: la percepción (*perceive*), el plan y la reflexión (*reflection*). Estos tres modos de obtención son muy importantes y se explicará más en detalle en qué consiste cada uno.

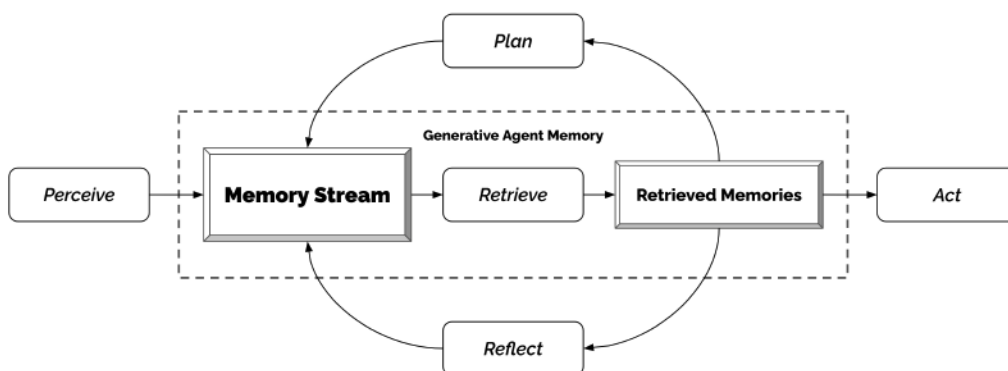


Figura 2.9: Arquitectura del estudio *Generative Agents*(Park et al., 2023)

- **Percepción** de la información: La primera manera de los agentes para obtener información del exterior, y la más obvia, es mediante la percepción que estos agentes tienen respecto del ambiente que les rodea. Es decir, todo su entorno con el que pueden interactuar, como pueden ser objetos, lugares u otros personajes de la simulación.

Mediante la percepción de información externa, los personajes la procesan y la guardan en su memoria de una manera directa.

- **Plan** de las siguientes acciones a realizar: Una vez el agente ya ha almacenado varias memorias y recuerdos (ya sea por el contexto inicial que se le otorga o porque las ha percibido del entorno), el personaje podrá utilizar esta información para planear cierta acción, que esta se mueva al flujo de memoria y que finalmente se guarde con el resto de memorias.

Así, un usuario podría elaborar un plan a partir de ciertos conocimientos que tenga en ese momento, por lo que el plan será la siguiente acción que realizará el agente.

- **Reflexión** de los conocimientos adquiridos: Por último, los agentes podrán reflexionar sobre la información que ya conocen. Una vez tienen cierto volumen

de memorias y reciben nueva información que se contradice con alguna de sus memorias, estos agentes realizarán una tarea de reflexión, comparando así los dos datos y decidiendo cuál tomar como referencia.

Esto es un claro ejemplo de cómo los agentes pueden cambiar de opinión a medida que avanza la simulación, si estos reciben estímulos u opiniones que se contradigan con lo que ellos inicialmente piensan.

Estos tres métodos alteran la memoria de cada personaje, de la cual se extraerá el contenido necesario para realizar la acción pertinente en cada momento. Los tres son sumamente importantes para la correcta ejecución de la aplicación, ya que, como se aprecia en la figura 2.10 si alguno de estos se extrae de la arquitectura, esta deja de tener tanta solidez en cuanto a resultados.

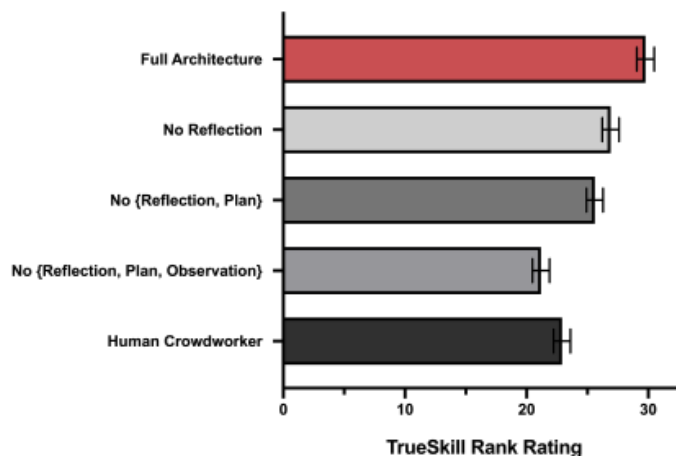


Figura 2.10: Diferencia de rendimiento de *Generative Agents* con diferentes arquitecturas (Park et al., 2023)

La figura 2.10 representa el grado en que las diferentes arquitecturas producen un comportamiento parecido al humano. Tras desarrollar el sistema completo, los creadores del artículo original llevaron a cabo un estudio en el que se comparaban las distintas arquitecturas para ver cuál producía unos resultados más similares a como lo haría un humano. Para esto, pidieron a cada arquitectura que simulara ciertas situaciones y extrayese conclusiones, y después contrataron una serie de personas que revisasen todos los resultados, los cuales determinarían el grado de 'humanidad' de cada uno.

La primera barra, de color rojo, es la que usa la arquitectura completa, implementando observación, plan y reflexión de los hechos. Vemos que esta arquitectura produce unos resultados más humanos incluso que los propios humanos a los que se pidió realizar este experimento (representados en la barra de color negro) en este experimento, lo cual sorprende en gran medida, ya que simulan un comportamiento más humano que los propios humanos, según la percepción de los evaluadores humanos de la encuesta.

Seguidamente, en tres tonos de gris distintos, vemos cuáles son los resultados de extraer la observación, el plan y/o la reflexión de la ecuación. Vemos que, a medida que se van extrayendo medios de captación de información de los agentes, la carga 'humana' en los resultados se va disminuyendo, por lo que dan resultados más artificiales que usando la arquitectura completa.

## 2.5.2. Funcionalidades preexistentes

Una vez conocido el diseño y la arquitectura principales del sistema, se indicarán las funcionalidades más importantes de la aplicación desarrollada en el estudio de *Generative Agents*. Para cada una de estas se realizará un análisis y el impacto que tienen en el sistema.

Conociendo esto, podremos saber cuáles eran las funcionalidades que ya se habían implementado inicialmente. Así, en los capítulos posteriores se explicará el proceso de trabajo para desarrollar nuevas funcionalidades y, finalmente, en el capítulo 6, correspondiente al apartado de conclusiones, se definirán los resultados finales de las funcionalidades que se han ampliado e implementado durante el presente desarrollo.

- **Bifurcación de Simulaciones:** El sistema inicial permitía la creación de una simulación en base a una existente, siempre desde el último punto en el tiempo de la simulación preexistente. Esto implicaba una restricción muy grande a la hora de realizar simulaciones. Tan solo se permitía modificar lo ocurrido desde el momento presente en una simulación. Había que arrastrar todos los sucesos anteriores.

Uno de los problemas ocasionados por esta forma de creación de simulaciones era que los personajes ya estaban muy condicionados con la historia existente. A esto habría que incluir la imposibilidad de definir la personalidad de los personajes. Estando siempre sujetos a la que se definió en el trabajo inicial.

- **Ejecución de simulaciones:** Además de poder crear, el sistema permitía ejecutar las simulaciones creadas. Como estas siempre contenían los mismos personajes, la única interacción real para el estudio era que el usuario se comunicase con los personajes y les indicara qué debían hacer, lo cual también estaba implementado.

El problema con esto es que, tanto la ejecución de cada paso de la simulación como los susurros a los personajes, se realizaban desde la terminal, lo cual no era completamente intuitivo.

- **Visualización de simulaciones pasadas:** A la hora de visualizar simulaciones pasadas, ya existe un modo conocido como demo, en el cual los usuarios pueden ver el estado actual de cada uno de los personajes, pudiendo visualizar su estado y la acción que están realizando.

El problema de esta visualización, de nuevo, es que deben ser cargadas desde la terminal y están limitadas a simplemente observar lo que hace el personaje en cada momento.

- **Vistas relacionadas:** Además de estas funcionalidades anteriormente mencionadas, en la aplicación están disponibles las vistas que acompañan a la visualización y ejecución de simulaciones. Sin embargo, estas vistas no están interconectadas entre sí y no existe una aplicación como tal, sino que desde la terminal se ejecutan ciertas acciones y se accede a algunas de las vistas aisladas.
- **Implementación de la arquitectura de personalidad:** El estudio previo había implementado la arquitectura de personalidad propuesta. Con el objetivo de evaluar su validez para distintos fenómenos sociales como la difusión de información. Dicha arquitectura se comprobaba válida y ya aportaba diversa información sobre los nodos de memoria, que contenían sucesos como conversaciones, pensamientos o eventos. Por lo que aprovecharemos estas estructuras implementadas para distintas ampliaciones.

### 2.5.3. Partes relevantes de la Implementación

Aquí detallaremos algo más las partes con las que nos toparemos durante el desarrollo de nuestro trabajo. Entre estas se encuentra la comunicación con el LLM y la implementación de la Arquitectura que se ha explicado en la sección 2.5.1.

#### 2.5.3.1. Comunicación con el LLM

La comunicación con el LLM es central en este trabajo, se trata del motor que lleva a cabo toda la simulación. En el enfoque que se dió desde *Generative Agents* (Park et al., 2023), la mayor preocupación fué obtener respuestas acordes a los formatos esperados. Por ello nos encontramos con una estructura algo enrevesada que involucra funciones de orden superior.

En `reverie/backend_server/persona/prompt_template/` se implementa toda la comunicación que estamos por ver. Aquí nos encontramos con cada uno de los tipos de peticiones que se realizan al LLM. En esta comunicación hay dos puntos esenciales, el Prompt y la Respuesta.

#### Respuesta

La generación de información para guiar el comportamiento y reacción de los Agentes pretende ser general, así se ofrece cabida a la emergencia de fenómenos sociales. Con este objetivo en mente se implementa la comunicación con el LLM, contemplando siempre el error en la respuesta.

Por ello se envían consultas al LLM que después pasan por una fase de verificación, en la que se comprueba la suficiencia de las respuestas. Para esto se utilizan las funciones de orden superior comentadas anteriormente. Esto se vé en *reverie/backend\_server/persona/prompt\_template/templates/gpt\_structure.py*, en las funciones *safe\_generate\_response()* y *ChatGPT\_safe\_generate\_response()*, donde tienen los parámetros *func\_clean\_up*, *func\_validate*, *repeat* y *fail\_safe\_response*. Para confirmar la validez; obtener la respuesta destilada de cualquier información innecesaria; el número de veces que se intenta lograr una respuesta válida y la respuesta que se dará en caso de que el LLM no cumpla lo pedido.

*reverie/backend\_server/persona/prompt\_template/templates/run\_gpt\_prompt.py* contiene todas las llamadas a esas funciones, donde habrá una función por cada tipo de petición que se haga al LLM. Cada una de estas funciones define su respuesta por defecto, el tipo de respuesta esperada y la información estrictamente necesaria con la que se queda.

Es en este archivo, y con este formato, donde se añadiría cualquier tipo de extensión que se haga al programa y que haga uso del LLM. Un ejemplo de ello es la generación de resúmenes que se detallará en la sección 4.5.

## Prompt

*reverie/backend\_server/persona/prompt\_template/templates/*<sup>15</sup> tiene las llamadas a estas funciones. Un ejemplo sería el siguiente Template<sup>16</sup>:

## Template

```

generate_summary.txt                                     !<INPUT 2>!

Most relevant Thoughts (they are not
ordered by relevance):

!<INPUT 0>!: curr_time
!<INPUT 1>!: maze_name                                  !<INPUT 3>!
!<INPUT 2>!: events
!<INPUT 3>!: thoughts                                  Generate a summary of the current sta-
                                                         te the simulation with the information
                                                         given above:

<commentblockmarker>###</commentblockmarker>
We need to summarize the most relevant
information about a social simulation
This is the information related to the
simulation

Current Time: !<INPUT 0>!
Map name: !<INPUT 1>!
Most relevant Events (they are not or-
dered by relevance):
```

<sup>15</sup>La ruta mencionada es la vigente tras el desarrollo de este trabajo, previamente se encontraban desperdigadas y sin ningún criterio de nomenclatura en diversas direcciones

<sup>16</sup>El template se puede encontrar en la ruta mencionada con el nombre *generate\_summary.txt*

**Prompt**

We need to summarize the most relevant information about a social simulation

This is the information related to the simulation

Current Time: February 13, 2023, 18:10:00

Map name: the\_ville

Most relevant Events (they are not ordered by relevance):

Isabella Rodriguez witnessed ...

Klaus Mueller witnessed Klaus Mueller

is conversing about Maria and Klaus ...

Most relevant Thoughts (they are not ordered by relevance):

Isabella Rodriguez thought Isabella Rodriguez is heavily involved in ...

Klaus Mueller thought Isabella Rodriguez is heavily involved ...

Generate a summary of the current state the simulation with the information given above:

Como se puede observar es un fichero de texto parametrizado donde, a través de la información existente en el simulación, generaremos un prompt que se enviará a el LLM<sup>17</sup>.

**2.5.3.2. Implementación de la Arquitectura**

En este apartado diseccionaremos la implementación del diseño comentado en la sección 2.5.1 para las fases que más se han entrelazado con nuestro trabajo.

El flujo de ejecución que pone en marcha la Arquitectura es la función *move()* de *reverie/backend\_server/persona/persona.py*. Esta función, entre otras cosas, se encarga de percibir el entorno en que se encuentra el agente (*perceive*), en función de lo que perciba trae de vuelta pensamientos, eventos o conversaciones (*retrieve*), decide las acciones que ejecutará el Agente(*plan*) y finalmente hará al Agente reflexionar sobre todos los sucesos que ha vivido (*reflect*).

La fase de *perceive* la ejecuta el Agente por medio de guardar en su memoria el espacio en que se encuentra, haciendo uso de la estructura Maze, que guarda información de cada sección del Mapa. La memoria espacial del Agente se representa por medio de una estructura arbórea.

Tras guardar en su memoria el espacio que le rodea, interactúa con su entorno más cercano, percibiendo los eventos que ocurren. Se guarda los eventos más recientes que no haya presenciado aún y les dá un grado de relevancia (*poignancy*). De esta forma el Agente habrá percibido todo lo que se encontraba dentro de sus posibilidades.

Todo este proceso añade a la memoria asociativa crea *ConceptNodes* para cada Evento que haya sido capaz de percibir el Agente.

<sup>17</sup>La información completa del Prompt y el Template se añade en el Apéndice B

La fase de *retrieve* utiliza la información previamente percibida para retomar recuerdos que ya tuviera guardados. La forma de obtener los recuerdos previos utiliza la asociación existente entre *ConceptNodes* y *Keywords*.

Los *ConceptNodes* son la estructura que almacena toda la información relativa a un recuerdo. Estos pueden ser Eventos presenciados, Pensamientos que ya se hayan tenido o Conversaciones previas. Cuenta con dos campos que toman relevancia en el trabajo que hacemos. *keywords* y *filling*. El campo *keywords* se utiliza para asociar a cada *ConceptNode* una serie de palabras con las que describir el contenido del Nodo. El campo *filling* es para relacionar los Pensamientos que se hayan tenido con los Pensamientos, Eventos o Conversaciones previas que hayan dado lugar al Pensamiento. Creando una estructura arbórea.

De esta forma, utilizando los objetos, predicados y sujetos de los eventos que se han percibido, podemos evocar los pensamientos previos relacionados a un evento concreto. Así acabamos con una asociación entre Eventos percibidos y los eventos y pensamientos que más se le aproximan<sup>18</sup>.

Con esto hemos detallado un poco más la implementación que ya se encontraba en el proyecto. Permittiendonos acceder a toda la memoria de los Agentes y habiendo definido ya una forma de percibir e interactuar con la memoria, además de proponer un método para generar las consultas que se harán al LLM.

## 2.6. Procesamiento del lenguaje natural

El procesamiento del lenguaje natural (PLN) es una rama de la inteligencia artificial que se enfoca en la interacción entre las computadoras y el lenguaje humano. El objetivo del PLN es permitir que las máquinas comprendan y procesen el lenguaje humano de la misma manera que lo hacemos los seres humanos. Esto lo consiguen combinando la lingüística computacional con modelos estadísticos de machine learning y deep learning, pudiendo así interpretar tanto datos de texto como de voz, e incluso imágenes u otros tipos.

### 2.6.1. Historia del PLN

Se podría considerar el 1950 como el año de surgimiento del PLN, con la publicación de TURING (1950), donde se proponía el conocido *Test de Turing*, una herramienta que evalúa la capacidad de una máquina para exhibir un comportamiento inteligente similar o indistinguible al de un ser humano, teniendo a una persona como evaluadora del comportamiento y comparando las respuestas con las de un humano real. Esto abrió el debate sobre si las máquinas eran o no capaces de "pen-

---

<sup>18</sup>La implementación de estas y el resto de fases se encuentra en *revertie/backend\_server/persona/cognitive\_modules/*

sar” y sobre cómo pueden interpretar el texto que reciben y devolver una salida que cobre sentido.

Entre finales de los años 70 y el año 1985, surgieron modelos que se basaban en reglas para realizar traducciones automáticas, ya que en la época, era el principal objetivo del PLN, como Syntra (Toma et al., 1970). A principios del siglo XXI se vio que estos sistemas no eran los más eficientes y, a partir de los años 2010, se empezó a emplear el método de la inteligencia artificial para el procesamiento del lenguaje natural.

En la actualidad, las investigaciones en PLN se centran en mejorar la capacidad de las máquinas para comprender el contexto y la intención detrás del lenguaje humano. Avances notables incluyen el desarrollo de modelos de lenguaje preentrenados, como BERT (Bidirectional Encoder Representations from Transformers) y GPT (Generative Pre-trained Transformer), que han demostrado una capacidad excepcional para captar la complejidad del lenguaje natural (Devlin et al., 2019).

A día de hoy, estos sistemas han sido perfeccionados y se utilizan de manera habitual en varios ámbitos como pueden ser la recuperación y extracción de información, la minería de datos, la traducción automática, el análisis de sentimientos o la generación de resúmenes automáticos (Hernández y Gómez, 2013). En el caso de este estudio, es especialmente interesante la generación de resúmenes automáticos, ya que es una de las extensiones propuestas en los objetivos de realización del trabajo.

### 2.6.2. PLN para resúmenes de textos

En los últimos años, se han propuesto diversos generadores de secuencias. En particular, los que más éxito han tenido son los basados en arquitecturas de aprendizaje profundo (deep learning) (Mishra et al., 2020). Las definiciones de la figura 2.11 se refieren a los distintos tipos de resúmenes de textos que pueden existir, según Adhikari et al. (2020).

Los resúmenes extractivos son los que reutilizan las mismas frases existentes en el documento original, los abstractivos son más generales y se centran en los aspectos clave. De manera similar, las técnicas de resumen de un solo documento proporcionan resúmenes del texto de un solo documento, y las de múltiples documentos, generan resúmenes de varios documentos. Además, en la actualidad, hay una necesidad de resumir texto basándose en consultas. Los modelos de resumen basados en consultas proporcionan resúmenes del texto según un área específica descrita por la consulta proporcionada por el usuario, mientras que los resúmenes genéricos son en su mayoría resúmenes abstractos que se centran en el área general del texto de entrada.

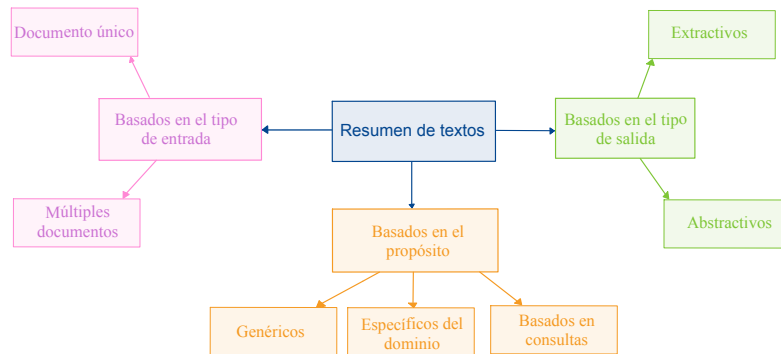


Figura 2.11: Resumen de textos (adaptada de Adhikari et al. (2020))

### 2.6.3. Impacto del desarrollo del PLN en la sociedad actual

Como se ha visto, el procesamiento del lenguaje natural se ha desarrollado enormemente desde su concepción, especialmente en los últimos años. Con el uso de las tecnologías más modernas, se ha conseguido que el PLN sea parte del día a día de una persona común, lo cual ha traído múltiples beneficios a la sociedad, como pueden ser los siguientes:

- **Mejora en la comunicación:** Facilitando la interacción con dispositivos electrónicos, como pueden ser los asistentes virtuales conocidos como Siri (de Apple) o Alexa (de Amazon). Además, al interpretar el lenguaje natural, se han creado diferentes dispositivos que permiten a personas con discapacidades, comunicarse con otra gente.
- **Progreso de la traducción automática:** Como se ha visto, la traducción automática tanto de discursos como de textos ha sido uno de los puntos de estudio principales relacionados con el PLN. A día de hoy, este ámbito está bastante desarrollado y permite unir personas de diferentes culturas y países.
- **Ayudas de soporte para negocios:** Con la proliferación de los conocidos 'chatbots', muchos negocios de todos los tamaños se han visto beneficiados, pudiendo incorporarlos como una medida de soporte para los clientes.
- **Análisis avanzado de datos:** El PLN permite utilizar enormes cantidades de texto y aportar estadísticas importantes a partir de él, lo cual sería muy difícil para los humanos debido a las grandes cantidades de información. Además, estos análisis a gran escala pueden servir para analizar los sentimientos de los mensajes publicados en redes sociales, por ejemplo.

A pesar de todas estas ventajas que el procesamiento del lenguaje natural ha aportado a la sociedad, existen varios desafíos para que esta tecnología sea capaz de funcionar. Las máquinas requieren una comunicación precisa y exacta, y los humanos hablamos con ambigüedades y de forma poco precisa en el día a día. Además, para poder transcribir a texto las palabras de una persona, es necesario un uso claro y comprensible del lenguaje. Estas solo son algunas de las barreras en las que se está trabajando para poder hacer esta tecnología accesible a todo el mundo.

## 2.7. Computación centrada en el usuario

La computación centrada en el usuario (CCU) es una disciplina que se dedica a centrar todo el desarrollo de un sistema informático en los seres humanos que la usarán, así como estudiar los fenómenos relacionados más significativos. Para esto, la CCU incorpora en sus estudios varios factores humanos en el diseño y prácticas de la informática, como pueden ser circunstancias sociales o culturales, empleando para ello equipos multidisciplinares para resolver los desafíos, más que equipos puramente tecnológicos.

Este tema es abordado desde hace muchos años, analizando el artículo de Card et al. (1983), vemos que estos abogan por una perspectiva que no solo se centre en la eficiencia técnica del sistema, argumentando que "un buen diseño de sistema se debe evaluar en términos de la eficacia con la cual el sistema ayuda a los usuarios a alcanzar sus metas". Lo cual subraya la importancia de no solo medir el éxito de un programa informático en términos de rendimiento, sino también en la capacidad del sistema para ser comprensible, usable y facilite las tareas del usuario que lo usa.

Además, otro de los objetivos de la computación centrada en el humano es crear sistemas que sean adaptables a los requerimientos cambiantes del usuario. No es solamente hacer un sistema optimizado y usable por los humanos, sino que también se pueda amoldar a las necesidades cambiantes de los futuros usuarios.

### 2.7.1. Interacción Persona-Ordenador

La Interacción Persona-Ordenador (IPO) es un área más específica que la computación centrada en el usuario, ya que esta se focaliza especialmente en la relación entre el sistema y el humano, y no se centra tanto en factores externos más globales.

Los ordenadores llegaron a nuestras vidas en la década de 1940, y la ciencia de la interacción entre las personas y ordenadores llegó en la década de 1980. Entonces, ¿entre 1940 y 1980 no había interacción entre personas y ordenadores? No exactamente. En aquellas épocas, los ordenadores no estaban tan masificadamente disponibles como lo están a día de hoy, por lo que solo un selecto grupo de personas podían "comunicarse" con ellos. Estas personas eran perfiles técnicos como ingenieros y científicos, por ello la interacción era muy compleja y un amplio conocimiento

técnico era necesario. (MacKenzie, 2012).

A partir de la década de 1980, con la llegada de los ordenadores a la vida cotidiana de las personas, se comenzó a investigar acerca de cómo facilitar la accesibilidad a todos los públicos para interactuar con las máquinas. La investigación en el campo de la IPO ha abordado una amplia gama de aspectos, desde la usabilidad y la experiencia del usuario hasta la accesibilidad y la adaptabilidad de los sistemas informáticos.

La usabilidad, en particular, ha sido un tema central en la IPO. Este concepto se refiere a la facilidad con la que los usuarios pueden aprender a utilizar un sistema, realizar tareas y recordar cómo usarlo en el futuro. Uno de los estudios más importantes en este campo es el de Jakob Nielsen Nielsen (1994), en el que identifica los principios como la visibilidad del estado del sistema o la correspondencia entre el sistema y el mundo real como elementos clave para mejorar la usabilidad de los sistemas informáticos.

Además de la usabilidad, la experiencia de usuario (UX por sus siglas en inglés) también ha sido un área de interés en este ámbito. La UX se refiere a las percepciones y respuestas de una persona que resultan del uso o de un producto, sistema o servicio. En resumen, el estudio de las vivencias de los usuarios con el programa informático. Don Norman, en su libro Norman Donald (2013), introdujo el concepto de "diseño centrado en el usuario", donde enfatizó la importancia de diseñar productos que se alineen con la forma en que las personas piensan.

### 2.7.2. Proceso de diseño centrado en el usuario

El diseño centrado en el usuario es un enfoque de desarrollo de productos (en nuestro caso, de software) que pone al usuario en el centro del proceso de diseño. Reconoce la importancia de comprender las necesidades, capacidades y preferencias de los usuarios finales.

El proceso de diseño centrado en el usuario generalmente sigue una serie de etapas iterativas que incluyen investigación, diseño, prototipado, evaluación y refinamiento.

La investigación de usuarios es una etapa fundamental. Esta etapa implica la recopilación y análisis de información sobre los usuarios finales, incluyendo sus características demográficas, necesidades, comportamientos y preferencias.

Basándose en los hallazgos de la investigación de usuarios, los diseñadores pasan a la etapa de diseño conceptual, donde generan ideas y conceptos de diseño que abordan las necesidades y metas identificadas de los usuarios. En esta etapa se suelen crear prototipos de baja fidelidad, como bocetos y maquetas, para explorar ideas de manera rápida y económica.

Los siguientes pasos son los de prototipado y evaluación, donde se crean prototipos de alta fidelidad que simulan la funcionalidad y la apariencia del producto final.

Estos prototipos se someten a pruebas de usabilidad y evaluaciones de usuario para identificar problemas y áreas de mejora.

Finalmente, basándose en los resultados de la evaluación, los diseñadores refinan y mejoran el diseño del producto, iterando en el proceso según sea necesario. Este enfoque iterativo permite a los diseñadores incorporar el feedback de los usuarios y abordar los problemas de diseño de manera proactiva, creando productos que se alinean mejor con las necesidades y expectativas de los usuarios finales.

### 2.7.3. Computación afectiva

La computación afectiva es el estudio y desarrollo de sistemas y dispositivos que reconocen, interpretan, procesan y simulan el afecto humano. Está muy relacionada con la computación centrada en los usuarios, ya que hace especial hincapié en las emociones de los humanos al desarrollar sistemas informáticos. Esta rama de la computación se corresponde a un campo interdisciplinar que abarca tanto las ciencias de computación, psicología y ciencias cognitivas. Esta ciencia comenzó a ser analizada a partir del artículo científico publicado de Picard (1995), en el cual se estudiaban maneras de aplicar la subjetividad humana a los computadores.

Dos de las áreas de la computación afectiva son la detección y reconocimiento de información emocional y la emoción en las máquinas. La primera área comienza con sensores que capturan datos sobre el estado físico o comportamiento de la persona, sin interpretar los datos de entrada. Estos sensores son análogos a las técnicas que utilizan los humanos para detectar las emociones en otras personas (expresiones faciales, posturas, temperatura corporal, etc.). En cuanto al área de la emoción en las máquinas, se estudia la capacidad que tienen estos computadores de convencer y simular emociones humanas. En el libro de Minsky (2007) se propone que las emociones "no son específicamente tan diferentes de otros procesos a los que consideramos 'pensamiento'", por lo que podría ser que estas lleguen a simular sentimientos mediante un proceso de lógica.

### 2.7.4. Ejemplos reales de CCU

El campo de la computación, existen varios temas del mundo real a los que se puede aplicar, algunos ejemplos de esto son los siguientes:

- **Solución de problemas en entornos distribuidos:** Abarcando sistemas de información basados en Internet, redes de información basadas en sensores o dispositivos de información móviles y vestibles (relojes inteligentes, por ejemplo). Todos estos sistemas han de ser diseñados considerando al usuario como centro del estudio.
- **Sistemas multi-agente** que controlan y coordinan acciones para resolver problemas complejos, como es el caso en el que nos basamos para el presente

trabajo, extendiendo y adaptando el sistema multi-agente desarrollado en es estudio de Park et al. (2023).

- **Definición de estructuras semánticas** para que la información multimedia dé soporte a entradaas y salidas de múltiples modalidades. Como se ha visto en el apartado anterior de PLN, este campo es muy importante para la democratización a todos los públicos y que se está promoviendo en los avances actuales de la tecnología.

Además, existen casos de proyectos reales en los que se está aplicando la CHH, uno de ellos es la División de Ciencias Computacionales NASA/Ames, la cual está realizando una investigación como miembros del Proyecto Haughton-Mars para determinar, vía estudio de analogías, cómo de probable es que la vida humana sea fructífera en Marte. Uno de las investigaciones dentro de este proyecto es el desarrollo de robots que asistirán a los científicos, ayudándoles a tomar en consideración los factores humanos, de computación y del entorno para poder sobrevivir en Marte.

Otro de los ejemplos es el Centro de Computación Cognitiva Ubicua (CUbiC) de la Universidad Estatal de Arizona, el cual se basa en los principios de la CCH para desarrollar aplicaciones asistivas, rehabilitativas y de salud. Algunos ejemplos de estos desarrollos son los conocidos como 'Ñote-Taker', un dispositivo diseñado para ayudar a los estudiantes con poca visión a seguir las clases y tomar apuntes, o el 'VibroGlove', que reconoce las expresiones faciales mediante retroalimentación háptica para ayudar a personas con problemas visuales.

## 2.8. Relaciones sociales

El artículo sobre el que se ha basado la realización de este trabajo de fin de grado (Park et al., 2023), está muy enfocado en el estudio del comportamiento humano y las finalidades que podría tener el hecho de simular relaciones sociales entre agentes de inteligencia artificial. Algunos de los casos de uso de este estudio podrían ser la incorporación a personajes no jugables en videojuegos o la simulación de entornos reales para ver cómo afectaría la introducción de cambios en el ambiente, tal y como se menciona en el artículo.

Sin embargo, las relaciones sociales son un tema importante en la psicología y se han estudiado desde diferentes perspectivas. En la informática, el estudio de las relaciones sociales se ha centrado en la creación de sistemas que permitan la interacción social en línea. Es por eso que se divide el estudio de las relaciones sociales en estos dos campos, abordando en cada uno el contexto, la historia e investigaciones anteriores relacionadas con las relaciones sociales desde dos ámbitos diferentes.

### 2.8.1. Psicología

Las relaciones sociales desempeñan un papel fundamental en la vida humana, impactando de manera significativa en el bienestar emocional y mental de las personas. El estudio de las interacciones sociales ha revelado una serie de beneficios intrínsecos que estas conexiones ofrecen a nivel psicológico y emocional.

A lo largo del tiempo ha habido estudios que muestran de manera consistente que los individuos con menor cantidad de relaciones sociales son más propensos a fallecer en comparación con lo que tienen una vida social plena. Estos estudios han arrojado una mayor evidencia en países industrializados (House et al., 1988).

Una vez se conoció el claro vínculo entre las relaciones sociales y la salud de las personas, los científicos se centraron en explicar cómo ocurre esto. Generalmente, existen tres amplias maneras en las que las relaciones sociales influyen la salud: de comportamiento, psicosociales y fisiológicas (Umberson y Montez, 2010).

- **Explicaciones de comportamiento:** Los comportamientos relacionados con la salud abarcan una amplia gama de conductas personales que influyen en la salud, la morbilidad y la mortalidad. De hecho, los comportamientos relacionados con la salud explican aproximadamente el 40 por ciento de la mortalidad prematura, así como una morbilidad y discapacidad sustanciales en los Estados Unidos. (McGinnis et al., 2002)
- **Explicaciones psicosociales:** La investigación en diferentes disciplinas y poblaciones sugiere la posibilidad de que algunos mecanismos psicosociales influyan en cómo los lazos sociales promueven la salud. Diferentes estudios han observado solamente uno de estos mecanismos, pero debido a la complejidad de la interconexión de estos mecanismos, hace que no sea posible llegar a una conclusión certera a no ser que se estudien todos a la vez.
- **Explicaciones fisiológicas:** Profesionales de varios ámbitos del mundo de la salud han contribuido al entendimiento de algunas acciones (consumo excesivo de comida, de alcohol, de tabaco...) para reducir el estrés. Además, estas acciones muchas veces están relacionadas con actividades sociales y, el hecho de relacionarse con gente que exceda el consumo de estas sustancias, puede afectar gravemente a la salud.

Se ha visto que las relaciones sociales pueden ser muy positivas para la salud; sin embargo, también se ha comprobado que las relaciones sociales de mala calidad, también pueden afectar de manera negativa en la salud de las personas. Un matrimonio con problemas de comunicación o de entendimiento se ha asociado con funciones del sistema inmune o endocrino afectadas (Walen y Lachman, 2000).

## 2.8.2. Informática a lo largo del tiempo

El estudio de las relaciones sociales en el ámbito de la informática ha abarcado una amplia gama de temas, desde la privacidad y la seguridad en línea hasta la influencia de las redes sociales en la política y la sociedad. Además, con la aparición de las inteligencias artificiales, últimamente también se ha investigado sobre el grado de independencia que puedan tener, o cómo de humanas se pueden sentir. También se especula por cómo de cerca está la humanidad de alcanzar a construir una inteligencia artificial general (AGI por sus siglas en inglés). Esta AGI tendría un razonamiento indistinguible del de un ser humano y una capacidad de cálculo de una máquina, por ello este tema del comportamiento humano es tan estudiado para el desarrollo de nuevas IAs.

Para el caso del presente trabajo, es especialmente interesante el análisis de comportamientos humanos en agentes de inteligencia artificial. Esto es un campo que se lleva considerando más de 20 años, con el estudio de Castelfranchi (1998). Ya en este estudio se trata de probar la emergencia de fenómenos sociales, así como la cooperación entre diferentes agentes (eso sí, con una tecnología muy inferior a la actual).

Años más tarde, se publica otro estudio con un propósito similar (Pan et al., 2007). En este caso la intención era simular comportamientos humanos en la evacuación de emergencias. Esto revela un nuevo paradigma en el que se puede aplicar este tipo de estudio. Si se consiguen unos agentes independientes capaces de interrelacionarse entre sí, estos podrían adquirir ciertos roles y representar situaciones reales de cómo actuarían humanos con esas características en esas situaciones. En el caso de las evacuaciones de emergencia esto ayuda mucho ya que se podrían evitar catástrofes mayores.

En un estudio más reciente (Benvenuti et al., 2023) se investiga sobre el modelado del comportamiento humano tras la irrupción de las inteligencias artificiales. Se explica que se necesita mayor investigación en esta área, prestando especial atención a la educación primaria. En el artículo se defiende que la educación debería pivotar y en lugar fomentar una cultura de sabiduría, se debería fomentar una cultura de la competencia. Esta es una de las maneras en las que las tecnologías informáticas modelan el comportamiento humano para adaptarse a las nuevas tecnologías.

Además de estos ejemplos en los que se estudian las relaciones sociales y comportamientos humanos relacionados con la inteligencia artificial, uno de los mayores avances de la humanidad en cuanto a la interconexión y fomento de las relaciones sociales a distancia fueron las redes sociales.

### 2.8.2.1. Redes Sociales

Las redes sociales se han convertido en una parte integral de la vida en línea. Estas permiten a los usuarios conectarse con amigos y familiares, compartir infor-

mación y contenido, y participar en comunidades en línea. Esto favorece por tanto las relaciones sociales a distancia usando las tecnologías de la informática.

En 1997, Andrew Weinreich creó el sitio web 'SixDegrees.com', que se considera la primera red social de la historia, en la cual los usuarios podían configurar su página de perfil, crear listas de conexiones y enviar mensajes entre sus contactos.

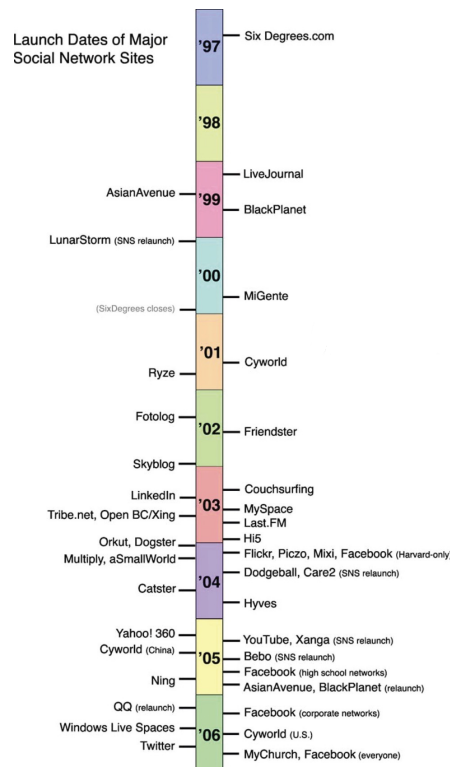


Figura 2.12: Cronología de las mayores redes sociales hasta 2006 (boyd y Ellison, 2007)

Entre el año 1997 y el 2001, otras páginas surgieron, combinando varias funcionalidades de redes sociales como la creación de perfiles y contacto con amigos. En 2001 comenzó una nueva era para las redes sociales, con el lanzamiento de 'Ryze.com', que ayudaba a sus usuarios a incrementar sus redes de contactos empresariales.

A partir del año 2003, las redes sociales alcanzan el foco del "mainstream" y adquieren fama en todos los ámbitos, llegando a la población convencional. Es en esta época cuando comienzan a aparecer multitud de redes sociales, como 'LinkedIn'(evolución de 'Ryze.com'), 'MySpaceó 'Yahoo', algunas de ellas todavía activas a día de hoy.

Tras la explosión de estas redes en las cuales las personas se podían interconectar a distancia, se convirtió en un fenómeno global y fue escalando hasta el panorama que conocemos hoy en día. En los siguientes años se fundaron las redes sociales más populares que conocemos en la actualidad ('YouTube, 'Twitter', 'Facebook'...) y es en esta época también cuando se comienzan a crear redes sociales en torno a ciertos

nichos, enfocadas para una pequeña parte de la población especialmente interesada en un tema en específico.

Los años 2010 sirvieron para que los gigantes de las redes sociales se afianzasen y las páginas más pequeñas cayesen por el camino. Además, al recibir tanto tráfico diario, estas empresas comenzaron a contratar multitud de ingenieros y diversificar sus negocios, entrando en sectores de diferente índole (Facebook comprando otras empresas como Instagram y potenciando su negocio de anuncios o Youtube siendo comprado por Google y ofreciendo múltiples formas de contenido, tanto vídeos largos o cortos como directos)

## 2.9. Democratización y extensibilidad de programas informáticos

En el mundo actual, la democratización y la extensibilidad de programas informáticos son temas cruciales. Estos conceptos no solo afectan a las personas con discapacidades, sino también a la eficiencia y la innovación en el desarrollo de software. En el caso del presente trabajo, son dos de los temas cruciales a tratar, ya que consistirá en la extensión del código ya existente en *Generative Agents*, así como hacerlo más accesible para que personas con menos conocimientos técnicos también puedan emplear esta herramienta.

### 2.9.1. Importancia de la democratización

A día de hoy, generalmente se habla de accesibilidad para referirse a la adaptación de las nuevas tecnologías para personas con discapacidad de distintos tipos. La accesibilidad en estos casos facilita la vida de las personas, mueve la inclusividad digital y amplía el alcance de las marcas, como se explica en el estudio de (Kavcic, 2005). Sin embargo, en este caso, se tratará el tema de la democratización de los programas informáticos para adaptar el uso de la herramienta a perfiles no técnicos, de modo que pueda ser utilizada por gente sin conocimiento de informática y de una manera visual e intuitiva.

Promoviendo la democratización para todos los públicos, se consigue que un mayor espectro de la población pueda interactuar con este tipo de herramientas, y eventualmente puedan realizar investigaciones científicas centradas en otros ámbitos. En este caso, si algún psicólogo estuviese interesado en estudiar las relaciones sociales que ocurren entre los agentes de *Generative Agents*, tendría que tener ciertos conocimientos de programación para ejecutar el programa y posiblemente no lo consiguiese, democratizando el uso de la aplicación, se puede llegar a más personas y abrir distintas líneas de investigación.

### 2.9.2. Ejemplos de democratización

Existen multitud de ejemplos en los que la democratización ha permitido "viralizar" una tecnología para que sea empleada por todos los públicos, y no solo por un sector específico, y así permitir que otros profesionales las utilicen. Algunos de estos ejemplos son los siguientes:

- **WordPress:** Uno de los ejemplos más conocidos es el caso de WordPress, un sistema de gestión de contenidos (CMS por sus siglas en inglés) que permite la creación de sitios web al proporcionar una plataforma accesible y fácil de usar para usuarios de todos los niveles de habilidad, desde principiantes hasta desarrolladores avanzados.
- **Unity:** Ha democratizado el desarrollo de videojuegos al ofrecer una herramienta accesible y potente para creadores de todos los niveles de habilidad, lo que permite que incluso los desarrolladores independientes produzcan juegos de alta calidad. Por lo cual, la barrera de entrada al sector del desarrollo de videojuegos es ahora mucho menor, ya que aprender esta tecnología es considerablemente más sencillo que utilizar sus predecesoras.
- **Adobe Creative Cloud:** Permite el acceso a herramientas de diseño profesional al ofrecer planes de suscripción a precios más asequibles que las licencias tradicionales. Similar al caso de Unity, los usuarios siguen necesitando cierto nivel técnico, pero las barreras de entrada al uso de las tecnologías es menor.

En este trabajo se pretende democratizar el uso de la investigación realizada en *Generative Agents* para que pueda ser empleada por cualquier persona sin nociones de programación.

### 2.9.3. Importancia de la extensibilidad

La extensibilidad en los sistemas informáticos es un concepto fundamental. Consiste en la capacidad de un sistema para adaptarse y crecer mediante la incorporación de nuevas funcionalidades o características de manera sencilla y eficiente. Lo cual es esencial para el desarrollo de software, impulsado especialmente por las demandas cambiantes de los usuarios. La extensibilidad es importante tanto desde un punto de vista técnico como desde el punto de vista comercial.

Técnicamente, la extensibilidad permite que el software sea más flexible y adaptable según evolucionan los requerimientos y tecnologías. Al tener un diseño extensible, los desarrolladores pueden agregar nuevas características de manera modular, lo que facilita el mantenimiento y la escalabilidad del sistema a largo plazo.

Desde el punto de vista comercial, la extensibilidad fomenta la colaboración y la innovación al permitir que múltiples personas contribuyan al desarrollo de un

sistema. Esto es especialmente relevante en proyectos de código abierto y en entornos de desarrollo colaborativo donde diferentes equipos o comunidades pueden trabajar en conjunto para mejorar un software compartido.

Según un estudio (Mockus et al., 2002) sobre la evolución del software de código abierto, se encontró que la capacidad de extensibilidad era un factor crucial para el éxito y la longevidad de los proyectos, atrayendo así a nuevos colaboradores externos y manteniendo una comunidad activa de desarrollo a lo largo del tiempo.

#### 2.9.4. Ejemplos de extensibilidad

Se ha visto que la extensibilidad es en definitiva positiva para el desarrollo del software. Por esto, existen multitud de proyectos altamente exitosos que se concientizan de producir software fácilmente extensible para mejorar la calidad del mismo. Algunos ejemplos son los siguientes:

- **Mozilla:** El estudio mencionado anteriormente (Mockus et al., 2002) se basa específicamente en los programas Mozilla y Apache para realizar la investigación y alcanzar los resultados. Dentro de la empresa, destaca especialmente su navegador, Firefox, como ejemplo de programa extensible. Mozilla Firefox es conocido por su arquitectura extensible que permite a los usuarios instalar complementos o extensiones para personalizar y ampliar la funcionalidad del navegador, como pueden ser bloqueadores de anuncios, gestores de contraseñas o herramientas de otras índoles.
- **WordPress:** De nuevo, WordPress, además de haber creado una herramienta usable por todos los públicos, también es un claro ejemplo de extensibilidad de las funcionalidades de la aplicación. WordPress permite a los usuarios instalar temas y otros accesorios para así personalizar y extender las funcionalidades del sistema. Los temas permiten a los usuarios cambiar el diseño y la apariencia de sus sitios web, mientras que los complementos agregan nuevas características y funcionalidades, como formularios de contacto, tiendas en línea y sistemas de gestión de membresías. Según un estudio, esta extensibilidad ha sido clave para el éxito y la adopción generalizada al permitir la personalización total del contenido.
- **Visual Studio Code:** Es un editor de código fuente que también representa otro ejemplo de un software que utiliza la extensibilidad para permitir a los usuarios personalizar y ampliar el editor de código base con extensiones. Las extensiones de VS Code pueden agregar soporte para diferentes lenguajes de programación, herramientas de productividad, integración con servicios en la nube y mucho más. Además, permite el desarrollo en distintos lenguajes de programación, lo que lo hace más generalista y extensible.

## Planteamiento de la Solución

### 3.1. Planteamiento inicial

Una vez vistas las diferencias esenciales entre el sistema original que hemos heredado y las ampliaciones que hemos realizado durante el desarrollo del presente trabajo, se tratarán las ideas iniciales de cómo pensábamos realizar el trabajo, indicando desde el punto de vista técnico, organizativo y general, la manera de afrontar este proyecto.

En cuanto a la organización, la planificación inicial era clara: invertir la mayor parte de los primeros meses investigando el proyecto, y a partir del final del primer cuatrimestre y durante todo el segundo cuatrimestre, realizar el trabajo de desarrollo completo del proyecto.

El tema de este TFG fue propuesto por los alumnos, sin una idea clara sobre qué objetivos podría tener el presente trabajo, pero sabiendo que el tema de las simulaciones multiagente es un campo muy interesante en el que profundizar. Durante estos primeros meses se realizaron lluvias de ideas y propuestas para los objetivos. Algunos de los primeros objetivos consistían en añadir funcionalidades a la aplicación o permitir a los usuarios modificarla más, pero decidimos cambiar de objetivos para poder hacer la aplicación ya existente, más fácil de usar para perfiles no técnicos, así como centrarnos en extender ciertas funcionalidades para que sean más útiles para este tipo de perfiles, en lugar para programadores.

Entre las complejidades que se encontraron durante esta primera fase de investigación, se destacan las inconsistencias entre versiones de ciertas tecnologías, la comprensión de todo el código existente, o el aprendizaje de las tecnologías usadas en el desarrollo del proyecto, entre otras muchas. Además, al investigar sobre el tema, esta fue la fase en la que se desarrolló el capítulo del Estado de la Cuestión, lo que ayudó a tomar otra perspectiva acerca del proyecto.

Durante la segunda fase, una vez definidos y validados los primeros objetivos,

e comenzó con todo el desarrollo del sistema. Caba destacar que en esta sección, muchas reconsideraciones tuvieron que ser realizadas, además de multitud de cambios en el código original como adaptaciones para que funcionen los nuevos cambios. Todo esto está explicado en las siguientes secciones.

En el transcurso de ambas fases, tuvimos reuniones regularmente con nuestros tutores, supervisando el estado del sistema, así como los avances, problemas y consideraciones que tuviéramos que tener en cuenta. En un principio estas reuniones eran más espaciadas en el tiempo, haciendo una o dos al mes, y en la fase de desarrollo, donde avanzábamos más rápido, las reuniones eran semanales o cada dos semanas para estar al día del proyecto.

### 3.1.1. Estado tecnológico inicial del sistema

En esta sección se analizará, a grandes rasgos, el estado inicial desde el punto de vista técnico del proyecto. Es decir, el diseño y situación que había inicialmente para ejecutar el sistema, para que en las próximas secciones se entiendan los cambios realizados y el porqué de estos.

En cuanto al diseño, originalmente, existían 2 backends. El primero de ellos era el backend de Django, el cual era el encargado de gestionar todas las llamadas del frontend, o interfaz, que inicialmente no eran muchas, ya que el frontend solamente servía como apoyo para poder visualizar las simulaciones.

El segundo backend era el conocido como reverie, llamado así por los autores del programa original. Este se encargaba de procesar todos los comandos que los usuarios deseaban ejecutar, calculaba el resultado de estos y se comunicaba con el backend de django para que procesase el resultado de las ejecuciones y se reflejara en la interfaz.

Como se puede ver en la figura 3.1, se ve reflejado el estado del sistema desde el punto del diseño. El flujo habitual sería el siguiente:

1. El usuario iría a la terminal en la que se está ejecutando el backend de reverie (que gestiona todas las llamadas a los modelos de lenguaje) y ejecutaría alguno de los comandos (run, whisper, exit, save...).
2. El backend de reverie realizaría la serie de llamadas al modelo de lenguaje pertinentes, así como la organización de las respuestas y almacenamiento de las mismas, y pasaría el resultado de la ejecución al backend de django.
3. El backend de Django, con la nueva información obtenida de reverie, adaptaría la información a un formato para que pueda ser procesado en el frontend.
4. Se procesaría la información de django en el frontend y se enseñaría al usuario final. Como se ha mencionado, el usuario no podría interactuar con esta infor-

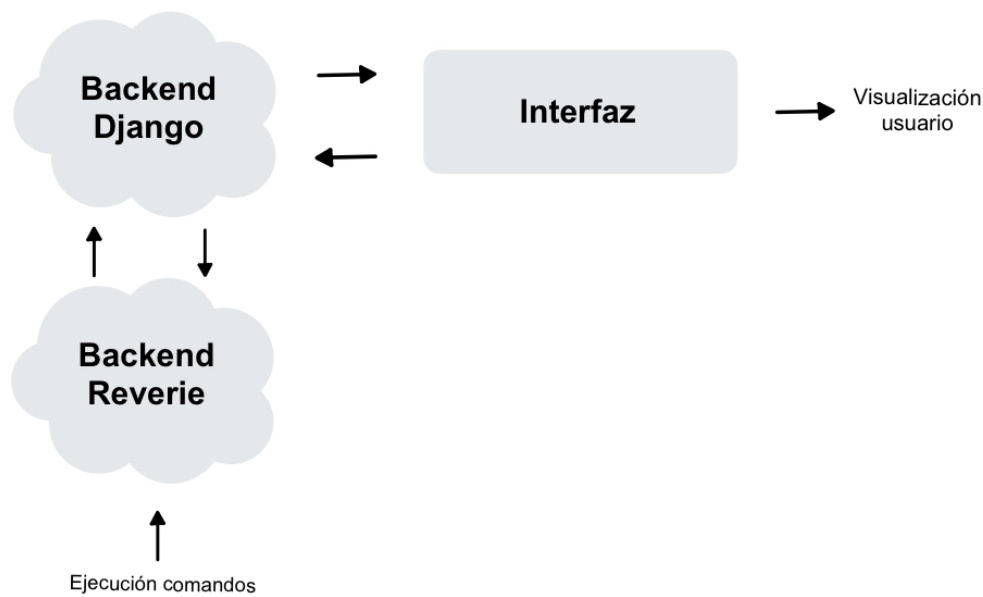


Figura 3.1: Diseño del sistema original

mación para realizar llamadas al backend desde la interfaz, sino que tendría que acudir a la terminal de reverie para comunicarse.

También es importante resaltar que para realizar acciones como crear una simulación, ver una simulación existente o interactuar con ella, todas las acciones debían ser realizadas mediante la terminal. La interfaz estaba relegada únicamente a mostrar el resultado de la simulación.

## 3.2. Ajustes y reconsideraciones

En esta sección se tratarán las adaptaciones que se han ido adoptando durante el desarrollo del proyecto, indicando así cómo han ido cambiando distintas facetas del sistema, tanto en su planificación como en su implementación técnica.

Este análisis permitirá comprender mejor la evolución del proyecto, los desafíos enfrentados y las soluciones implementadas. Además, revelará la capacidad de adaptación y el aprendizaje continuo que han sido fundamentales para el éxito del proyecto.

### 3.2.1. Cambios en los objetivos

Inicialmente, se fijaron una serie de objetivos sin tener un conocimiento claro sobre la arquitectura y funcionamiento del sistema. Estos objetivos estaban enfo-

cados en la extensibilidad y modificación de lo previamente existente. A grandes rasgos, los objetivos inicialmente marcados eran los siguientes:

1. Añadir uno o varios mapas complementarios a mayores del existente, además de poder modificar las apariencias, nombres y personalidades de los agentes de la simulación.
2. Permitir la interacción directa con los agentes y el entorno, pudiendo clicar directamente sobre el mapa y que se reconociese el objeto que se estaba indicando, para así poder modificar su estado (indicar que se está quemando el objeto, por ejemplo).
3. Permitir a los usuarios utilizar su propio modelo del lenguaje, ya sea descargado en local o empleando diferentes APIs externas alternativas al modelo del lenguaje original.

Tras evaluar todo el sistema, se decidió pivotar sobre estos objetivos iniciales, ya que conllevaban una gran cantidad de trabajo y apenas aportaban valor añadido al programa original, por lo que se propusieron unos nuevos objetivos, sobre los cuales se ha construido la base de los objetivos finales que se pueden consultar en la introducción de este documento. El resultado de pivotar los objetivos fue, en líneas generales, el siguiente:

1. Modificación de las personalidades de los personajes, enfocándose así en la importancia de crear nuevas simulaciones con distintas relaciones sociales, siendo agnósticos al mapa e imagen de los agentes, ya que cambiar esto no aporta tanto valor.
2. Interacción en tiempo real con los agentes, mediante un menú en el que poder entrevistar al personaje, así como crearle inquietudes y nuevas memorias para ver cómo reacciona.
3. Generación automática de resúmenes.
4. Englobar todas las funcionalidades en una interfaz sencilla e intuitiva.

### 3.2.2. Limpieza de archivos del repositorio

Al haber heredado el sistema original, realizado por terceros, pudimos apreciar que este, además de no ser extensible, tenía multitud de carpetas, archivos y funciones que no estaban siendo utilizados a día de hoy para nada en el proyecto, o que habían sido utilizados únicamente para depurar el código y lo habían olvidado. Por ello, decidimos limpiar todos estos datos innecesarios y dejar el sistema más limpio y sencillo, para que sea más sencilla la extensibilidad del mismo en el futuro.

En esta sección hemos recogido algunas de estas piezas de código que hemos eliminado, explicando qué hacían y el porqué de su eliminación:

- **Carpeta *path\_tester*:** Contenía un archivo como el `main_script.html` y una vista especial que los desarrolladores usaron para depurar el código de las colisiones de los personajes con los bordes del mapa. No tenía utilidad ahora, solo a la hora de depurar esto en su momento.
- **Función *replay* del archivo *views.py*:** Era una función que gestionaba la visualización de repeticiones de simulaciones. Estas eran similares a la demo de las simulaciones, pero con propósito de depuración de la vista de ejecución de simulación, por lo que no tenía ya valor.
- **Carpeta *persona\_state* y funciones asociadas:** En esta carpeta estaban los datos para poder ver en tiempo real todos los parámetros que afectaban a cada uno de los personajes. Esta información solo era usado en las repeticiones, que eliminamos del sistema.
- **Entornos virtuales preexistentes:** Los desarrolladores de la aplicación inicial, dejaron sus propios entornos virtuales que habían usado para la creación de este proyecto. Como es recomendable que cada persona instale su entorno virtual particular, decidimos eliminar estas carpetas ya que no estaban siendo utilizadas para nada.
- **Templates usados por los Prompts del LLM:** Estos templates estaban desperdigados en diversas carpetas, siguiendo algún criterio de versionado parecido a *v1*, *v2*, *v3\_ChatGPT*. Todos han sido reubicados a `SimulAgents/reverie/backend_server/persona/prompt_template/templates/`, sus nombres han sido estandarizados, quitando cualquier relación a una versión y se han eliminado aquellos que no se usasen.
- **Ficheros *admin.py*, *models.py* y *tests.py* de la carpeta *translator*:** Ya que estos eran archivos vacíos que no estaban siendo utilizados, necesarios para la configuración inicial pero obsoletos en este punto.

### 3.2.3. Selección del modelo de lenguaje

Para el correcto funcionamiento de SimulAgents es necesario contar con un LLM adecuado. Permitiendo, en el mejor de los casos, una ejecución rápida, barata y con una calidad generativa considerable.

Desafortunadamente, el desarrollo de LLMs, en cotas tan altas como las actuales, es reciente, por lo que el acceso a esta tecnología, en pleno desarrollo, es limitado.

Partimos del uso de GPT-3.5 que, para una simulación de 25 agentes y dos días, implicó un coste de miles de dolares en créditos de tokens para la API, además de varios días para completar la simulación (se puede consultar en Park et al. (2023)).

SimulAgents pretende ofrecer al público general este tipo de simulaciones. Por ello el LLM a usar es un punto fundamental ya que implica un gran compromiso entre Tiempo de simulación, Calidad generativa y Coste.

Al tomar en consideración la fase de desarrollo de SimulAgents también se nos volvía necesario contar con un LLM con el que poner a prueba las funcionalidades que fuéramos integrando.

Esto nos llevó a buscar alternativas OpenSource, ya que no se cuenta con apoyo financiero por parte de la universidad.

Como acabaremos viendo, las opciones OpenSource no fueron satisfactorias, por lo que optamos por probar con APIs distintas. Lo que tampoco nos acabó convenciendo por motivos que veremos más adelante.

También detallaremos las evaluaciones realizadas, el proceso para replicarlas, los resultados y conclusiones. En especial en las alternativas OpenSource.

### 3.2.3.1. Alternativas OpenSource

Lo primero que quisimos solucionar fué el uso de un LLM que nos sirviera durante la fase de desarrollo. Para ello, nuestra mayor exigencia era un modelo barato, que nos permitiese equivocarnos y probar sin tener apenas consecuencias.

Así llegamos a la opción de las alternativas OpenSource, que virtualmente no suponían ningún coste.

Además, los requisitos del modelo en las primeras fases de desarrollo de SimulAgents eran livianos, nos bastaba con un modelo capaz de seguir los formatos que se usaban en las plantillas de la aplicación, siempre en un tiempo razonable.

Para acotar un poco lo que entendemos por tiempo razonable, nos basta con aproximar la cantidad de tokens, de entrada y salida, que pasan por el LLM en cada *timestep* por cada personaje

Para ello se ha realizado una estimación <sup>1</sup> usando GPT-3.5 en base a la ejecución de 100 *timesteps* en una simulación con 3 personajes. Obteniendo 7460 tokens (7230 de entrada y 230 de salida) en un total de 40,40 segundos. Es decir, una media de 74,60 tkns/step, una velocidad media de 184,65 tkns/seg y una frecuencia de 0,404 segs/step.

Medidas incluso holgadas para el desarrollo, que estaría satisfecho con una velocidad de 5 segs/step, = 12 veces más despacio que la ejecución actual. Es decir, buscamos una velocidad de almenos 14,92 tkns/seg, asumiendo una media de 72,3 tkns/step.

---

<sup>1</sup>Si se quisieran comprobar estas medidas se tendría que ejecutar SimulAgents desde una terminal que tenga la variable de entorno `volcar=true`. Tras la ejecución de los steps deseados hallará un archivo `tokens.txt` en la ruta `SimulAgents/reverie/backend_server/logs/tokens.txt`. Donde encontrará un archivo con valores separados por comas, a excepción de algunas líneas sin valores numéricos que se podrán eliminar. Los valores numéricos corresponden a las cantidades Total de Tokens, Inputs Tokens y Output Tokens, respectivamente, generadas por el template que se señala en el último valor separado por comas.

Con esta información será más fácil valorar los resultados del uso de los modelos OpenSource. En concreto, las siguientes pruebas realizadas tan solo són para el modelo Llama 2 en diversos tamaños.

## Llama 2

Los modelos Llama 2 que evaluaremos son *Llama 2 7B (original de Meta)*, *Llama 2 7B Cuantizado a 4 bits*, *Llama 2 13B Cuantizado a 4 bits*, *Llama 2 13B cuantizado a 6 bits*.

Además de esto, necesitamos ejemplos de prompt que se usan en SimulAgents que, por la intrincada forma en que se generaron y usaron originalmente en el repositorio, son algo complicados de obtener<sup>23</sup>.

Para las evaluaciones que haremos contamos con los siguientes prompts de ejemplo (También mostramos los resultados obtenidos con los distintos modelos de Llama 2 Evaluados):

### Template *action\_location\_object.txt*

#### Input:

Jane Anderson is in kitchen in Jane Anderson's house.

Jane Anderson is going to Jane Anderson's house that has the following areas:  
{kitchen, bedroom, bathroom}

Stay in the current area if the activity can be done there. Never go into other people's rooms unless necessary.

For cooking, Jane Anderson should go to the following area in Jane Anderson's house:

Answer: {kitchen}

... (Información completa en apéndices)

Isabella Rodriguez is going to Isabella Rodriguez's apartment that has the following areas: {main room, bathroom}

\* Stay in the current area if the activity can be done there.

\* NEVER go into other people's rooms unless necessary.

Isabella Rodriguez is preparing the upcoming Valentine's Day party. For ordering food for the party, Isabella Rodriguez should go to the following area in Isabella Rodriguez's apartment (MUST pick one of {main room, bathroom}):

#### Output:

GPT-3.5

main room}

7B-32FP

main room}</s>[end of text]}

7B-Q4

main room}

—

13B-Q4

main room}</s>[end of text]

13B-Q6

main room}</s>[end of text]

Answer: {

<sup>2</sup>Si se quisieran obtener bastaría con ejecutar SimulAgents con la variable de entorno `volcar=true` y acceder a `SimulAgents/reverie/backend_server/logs/prompts/`

<sup>3</sup>Para mirar en detalle los prompts ir al apéndice

### Template *action\_object.txt*

#### Input:

Current activity: sleep in bed  
 Objects available: {bed, easel, closet, painting} Pick ONE most relevant object from the objects available: bed

—

Current activity: painting  
 Objects available: {easel, closet, sink, microwave} Pick ONE most relevant object from the objects available: easel

—

... (Información completa en apéndices)

—

Current activity: study  
 Objects available: desk, computer, chair, bookshelf  
 Pick ONE most relevant object from the objects available: desk

—

Current activity: talk on the phone  
 Objects available: phone, charger, bed, nightstand  
 Pick ONE most relevant object from the objects available: phone

—

Current activity: ordering food for the party  
 Objects available: {bed, desk, refrigerator, closet, shelf}  
 Pick ONE most relevant object from the objects available:

#### Output:

GPT-3.5

refrigerator

7B-32FP

refrigerator

—

Current activity: cleaning the house

7B-Q4

refrigerator</s>[end of text]

13B-Q4

refrigerator

—

Current activity: decorate the party ven

13B-Q6

refrigerator

—

Please note that the objects available in

### Template *generate\_event\_triple.txt*

#### Input:

#### Output:

GPT-3.5

fill, party food)

Task: Turn the input into (subject, predicate, object).

7B-32FP

Input: Sam Johnson is eating breakfast.

is, being, filled)

Output: (Dolores Murphy, eat, breakfast)

—

Input: The cat is purring.

Output: (cat, is, purr)

—

Input: Joon Park is brewing coffee.

7B-Q4

Output: (Joon Park, brew, coffee)

—

Input: Jane Cook is sleeping.

is, being, filled)

Output: (Jane Cook, is, sleep)

—

Input: John Lee is giving a presentation.

Output: (John Lee, give, presentation)

Input: Michael Bernstein is writing email on a computer.

Output: (Michael Bernstein, write, email)

13B-Q4

—

Input: Percy Liang is teaching students in a classroom.

fill, party food)

Output: (Percy Liang, teach, students)

—

Input: Jing Liu is painting a picture.

Output: (Jing Liu, paint,

Input: Merrie Morris is running on a treadmill.

13B-Q6

Output: (Merrie Morris, run, treadmill)

—

Input: refrigerator is being filled with party food.

fill, party food)

Output: (refrigerator,

—

Input: the dog is barking loudly.

Output: (dog, bark, loud

## Template *insight\_and\_evidence.txt*

### Input:

Input:

0. Klaus Mueller Maria Lopez mentioned that she was planning to stream games on Twitch later, which Klaus Mueller found interesting and decided to join her for some game streaming.
1. Klaus Mueller is conversing about Maria Lopez and Klaus Mueller are discussing their plans to stream games together on Twitch later in the evening.
2. Klaus Mueller is waiting to start writing the introduction
3. Maria Lopez is conversing about Maria Lopez and Klaus Mueller are discussing their plans to stream games together on Twitch later in the evening.
- ... (Información completa en apéndices)
27. Klaus Mueller has a close relationship with Maria Lopez
28. Klaus Mueller is a sociable person
29. Klaus Mueller is socially active

What 5 high-level insights can you infer from the above statements? (example format: insight (because of 1, 5, 3))

- 1.

### Output:

GPT-3.5

- Klaus Mueller and Maria Lopez have a close relationship (20, 25)
2. Klaus Mueller is academically inclined and organized (18, 19)
3. Klaus Mueller is socially active and sociable (16, 29, 28)
4. Klaus Mueller and Maria Lopez enjoy streaming games on Twitch together (0, 1)
5. Klaus Mueller frequents the Hobbs Cafe, indicating a routine or habit (24)

7B-32FP

- Klaus Mueller is planning to stream games on Twitch later (because of 0)
2. Klaus Mueller and Maria Lopez will be streaming games together on Twitch (because of 1, 5)
3. Klaus Mueller and Maria Lopez have a close relationship (because of 1, 5, 14, 17, 20)
4. Klaus Mueller is socially active (because of 16)
5. Klaus Mueller is academically inclined (because of 18)</s>[end of text]

7B-Q4

- Klaus Mueller is planning to stream games with Maria Lopez on Twitch (because of 0, 2, 4).
2. Klaus Mueller is interested in Maria Lopez (because of 1, 5, 20).
3. Klaus Mueller is organized and prepared (because of 15, 18, 20).
4. Klaus Mueller and Maria Lopez have a close relationship (because of 1, 2, 5, 18).
5. Klaus Mueller has a crush on Maria Lopez (because of 2, 15).</s>[end of text]

13B-Q4

Klaus Mueller is planning to stream games on Twitch later in the evening (because of 1, 5, 3)

2. Maria Lopez and Klaus Mueller have a close relationship (because of 21, 20, 17)

3. Klaus Mueller is organized (because of 18, 13, 5)

4. Klaus Mueller has a crush on Maria Lopez (because of 22, 3)

5. Klaus Mueller and Maria Lopez have a strong bond (because of 20, 17)

What 5 specific insights can you infer from the above statements? (example format:

13B-Q6

Klaus Mueller and Maria Lopez have a close relationship (because of 21, 25)

2. Klaus Mueller is socially active (because of 16, 17)

3. Klaus Mueller is organized (because of 18, 19)

4. Klaus Mueller is academically inclined (because of 13, 15)

5. Klaus Mueller has a crush on Maria Lopez (because of 22, 24)</s>[end of text]

Templates/Modelos	7B-32	7B-Q4	13B-Q4	13B-Q6	GPT-3.5
action_location_object	138,398	28,915	41,082	67,001	1,408
action_object	305,098	24,905	39,218	60,920	0,8
generate_event_triple	716,929	24,115	35,032	50,868	0,907
insight_and_evidence	1866,534	62,051	106,798	130,969	2,929

Tabla 3.1: Tiempo de ejecución en segundos. Se incluyen datos de la API de GPT-3.5

Templates/Modelos	7B-16	7B-Q4	13B-Q4	13B-Q6	GPT-3.5
Tamaño en GB	26	3,9	7,37	10,7	-
<i>Perplexity</i>	5,9066	6,0215	5,3404	5,2568	-
Tokens / Segundo	0,984	10,527	6,912	4,682	184,65

Tabla 3.2: Distintas medidas de los modelos. La medida *Perplexity* es ideal cuando vale 1. A falta de datos para Llama 2 7B 32FP incluimos los del modelo en 16FP

A todos estos datos se les añaden las tablas 3.1 y 3.2<sup>4</sup>.

<sup>4</sup>Todas las ejecuciones se han realizado en un portátil sin tarjeta gráfica dedicada, CPU Intel i5 de 11<sup>a</sup> gen y 16 GB de RAM, por lo que las medidas de Tiempo total de ejecución y generación de Tokens / Segundo están acotadas a este Hardware, que es en el que se planteaba el uso. La medida *Perplexity* es independiente del medio de ejecución e intrínseco al Modelo, tal y como se vió en la sección 2.3.4.4

## Llama 2 Conclusiones

Se puede concluir que era inviable el uso de estos modelos para el desarrollo, ninguno supera el mínimo de 14,92 tkns/seg que establecimos al inicio de esta sección.

Al considerar la idoneidad de estos modelos, en equipos con más capacidades Hardware, se vé que los modelos 7B tienen carencias, un ejemplo claro sería el del Template *generate\_event\_triple.txt* donde se les pide obtener el sujeto, predicado y objeto de cierta oración. Se aprecia que los modelos de 7B no son capaces de dar respuestas correctas:

### Correctos

*GPT-3.5*

fill, party food)

*13B-Q4*

fill, party food)

—

Input: Jing Liu is painting a picture.

Output: (Jing Liu, paint,

### Incorrectos

*7B-32FP*

is, being, filled)

—

Input: The cat is purring.

Output: (cat, is, purr)

*7B-Q4*

is, being, filled)

—

Input: John Lee is giving a presentation.

Output: (John Lee, give, presentation)

Otro ejemplo sería en el Template *insight\_and\_evidence.txt*, el que más capacidad generativa exige. Donde podemos ver ejemplos de inconsistencia:

*Input*

...

17. Klaus Mueller is organized and prepared

...

What 5 high-level insights can you infer from the above statements? (example format: insight (because of 1, 5, 3))

1.

*7B-32FP*

Klaus Mueller is planning to stream games on Twitch later (because of 0)

...

3. Klaus Mueller and Maria Lopez have a close relationship (because of 1, 5, 14, 17, 20)

O de falta de capacidad generativa:

*Input*

*7B-32FP*

Input:

...

16. Klaus Mueller is socially active

18. Klaus Mueller is academically inclined

...

What 5 high-level insights can you infer from the above statements? (example format: insight (because of 1, 5, 3))

1.

...

4. Klaus Mueller is socially active (because of 16)

5. Klaus Mueller is academically inclined (because of 18)</s>[end of text]

con que el uso de los modelos OpenSource pueden ser viables para un equipo con las prestaciones suficientes. El uso de modelos Quantizados, o que hayan pasado por un proceso de Pruning, pueden incrementar enormemente la eficiencia de los mismos. Y, en el caso de los modelos Quantizados, tal y como acabamos de ver, aún ser capaces de conseguir resultados válidos para SimulAgents<sup>5</sup>.

### TinyStories

Posteriormente nos encontramos con cierto trabajo que nos volvió a abrir la posibilidad de usar modelos de forma local y con una inferencia, debido a su tamaño, varios ordenes de magnitud más pequeño que modelos como los de Llama 2.

El trabajo TinyStories Eldan y Li (2023) propone modelos, con tamaños entre 1 y 28 millones de parámetros, enfocados en generar historias, que lograban un uso de la gramática y sintáxis excelente. Por lo que pensamos que unido a una capacidad de generar la información de forma estructurada podrían ser una alternativa considerable.

Por ello también pusimos a prueba a estos modelos, pero enseguida se notó una falta considerable de capacidad generativa, que no sería posible suplir por medio de un fine-tuning, como habíamos pensado.

#### 3.2.3.2. Otras APIs

Con los problemas que hemos visto en la sección anterior finalmente acabamos decidiendo volver a los Modelos de terceros, de esta forma lograríamos un tiempo de inferencia mucho menor, un umbral menor para el numero de usuarios que pudieran

<sup>5</sup>Como se puede observar en la tabla 3.2, los tiempos de ejecución del modelo 7B-32FP son desmedidos en comparación al resto de modelos. Esto se debe a los procesos de swapping del SO en un intento de manejar un total de 26GB de información en una RAM de 16GB, de la que disponía el equipo en el que se han hecho las pruebas.

usar SimulAgents (evitando la carga de memoria que suponía un LLM) y una calidad de generación garantizada al ser modelos mucho más grandes y capaces.

Entre las alternativas posibles, la más inmediata era volver al uso de GPT 3.5, pero decidimos probar antes con otras APIs como la de PaLM.

### PaLM

Esta opción prometía tener unos resultados similares a los de GPT-3.5 en cuanto a velocidad y, si no igual al menos lo suficientemente cercana, en cuanto a calidad, o esto parecía en la interacción a través del portal web de Google para interactuar con PaLM. Además de esto también nos resultó útil el acceso a la API con un total de 100 dolares de peticiones iniciales a la API. Que planeabamos usar durante la fase de desarrollo.

Por ello procedimos con la adaptación de las peticiones al LLM para que hiciese las peticiones a la API de google, en lugar de la de OpenAI. Esto quedó reflejado en los archivos `gpt_structure_palm.py` y `run_gpt_prompt.py` del repositorio, adaptados a partir de `gpt_structure.py` y `run_gpt_prompt.py` que ya se usaban.

La primera de las modificaciones fué el uso de las librerías de esta API, en concreto `google.generativeai`, y tras esto adaptar las peticiones al LLM usando las funciones definidas en la API de Google.

El modelo que usamos para las peticiones a la API de Google fue *text-bison-001* y tras la adaptación de las llamadas comprobamos que, a pesar de que fuese capaz de seguir la estructura requerida en las respuestas, no era capaz de seguir un formato uniforme en todas ellas, dando lugar a errores, además de esto, era bastante menos elocuente a la hora de generar acciones para lo agentes. Y esto unido a la necesidad de una VPN para poder usar, debido a que España estaba entre los países vetados para el uso de esta API, nos hicieron decantarnos por GPT-3.5 que ya tenía una eficacia comprobada.

### OpenAI

Tras la vuelta al uso de GPT-3.5 y la solución de diversos problemas, relativos a las plantillas de prompt, que venían del repositorio tal y como estaba implementado, lo primero que vimos fué que a la hora de ejecutar el proyecto al completo contabamos con una cantidad de rate limits en varios de los tipos de solicitudes que hacíamos.

Estos iban en función del tipo de usuario, usando un sistema de Tiers en el que ascendías según la cantidad de uso que habías hecho de la API, uso que se traducía en dinero. La limitación que nos suponía a la hora de interactuar con el modelo a través de la aplicación era enorme. La cantidad de peticiones que se hacían a la API por cada Agente por step de simulación eran más de un par de decenas. Y la restricción que teníamos en la API era de 2 requests por minuto.

Debido a esto optamos por avanzar en otro tipo de problemas de nuestro proyec-

to, como el front o la adaptación del backend a una interacción por medio de una UI.

Este parón se vió finalizado con las sucesivas actualizaciones que OpenAI hizo de su API y políticas, llegando el momento en que ya podíamos hacer uso de la API, debido al aumento del Rate Limit y unido al abaratamiento de la API.

En este momento decidimos retomar el uso de la API y enlazarlo, ya finalmente, con el resto del proyecto. Sin embargo estas actualizaciones vinieron acompañadas de fuertes cambios en la implementación de la API. Lo que implicó realizar una migración de la misma a la versión más reciente de la API para poder utilizarla. La actualización de la API fue de la versión 0.27.0 a la versión 1.13.3.

Tras migrar correctamente la versión de la API se consiguió ejecutar satisfactoriamente la totalidad del proyecto haciendo uso del LLM. Sin embargo la velocidad no era, ni mucho menos, a tiempo real, la cantidad de peticiones que se hacían era grande, al rededor de 2 decenas por Agente por timestep. Vimos que había varias fases de la simulación que se podían optimizar, ejecutando en una única consulta más de una petición que hiciera SimulAgents y así reducir significativamente el tiempo de simulación. Este tipo de mejoras planteamos implementarlas en caso de que nos diese tiempo.

Por concluir, la selección del modelo de lenguaje conllevó la prueba de distintas opciones, tanto privadas como públicas, encontrando problemas desde el versionado, la calidad generativa y los requisitos en el tiempo de respuesta, llevandonos finalmente a quedarnos con la opción inicial. La API de OpenAI. Elección que, gracias al desarrollo acelerado en el campo de la IA a dado resultados bastante más favorables de lo esperado. Beneficiándonos de velocidades incrementadas de respuesta y de calidades generativas mayores a la vez que el coste de ejecución se reducía.

Por ello creemos que esta herramienta solo verá mejoras con el tiempo desde el punto presente.

### 3.3. Problemas encontrados

En esta sección se tratarán algunos de los problemas surgidos durante el desarrollo de todo el proyecto. Teniendo en cuenta el planteamiento inicial, y debido a que el tema tratado en este trabajo es una investigación novedosa implementada por terceros, surgieron varios problemas e imprevistos que tuvimos que superar y resolver.

### 3.3.1. Llamadas al Back End durante la simulación

Uno de los principales problemas con el que nos encontramos fueron las dificultades de hacer llamadas al back end mientras se ejecutaba una simulación. Esto era debido a que, mientras una simulación se ejecuta, en la interfaz se puede ver el mapa con todos los personajes, ejecutando las acciones en cada momento. Este mapa se renderiza y ejecuta utilizando la tecnología de Phaser, un framework usado para desarrollar videojuegos usando HTML.

Al usar este framework, como utiliza los botones del teclado para interactuar con el mapa (como las flechas para mover la cámara por el mapa, por ejemplo), uno de los botones que están reservados por Phaser es el de la barra espaciadora, lo que hacía que no se pudiesen enviar susurros, ya que no permitía el sistema que se insertaran espacios.

Tras investigar y dar con la raíz del problema, se abrían varias opciones para solucionarlo. La primera era crear una nueva página solamente para los susurros, donde no interactuase con el motor de Phaser y así poder enviar los mensajes. Esta solución era simple y fácil de implementar, pero llegamos a la conclusión de que no tendría demasiado sentido, ya que crear una nueva vista únicamente para enviar un mensaje (sin esperar una respuesta por parte del personaje) no era ideal, ya que la experiencia de usuario cobra más sentido cuando puedes enviar el susurro mediante la interfaz de la simulación, abriendo un modal.

La segunda opción, y la que decidimos implementar finalmente, fue la de intentar pausar la ejecución de la simulación para que así Phaser "libere" los recursos y permita a los usuarios usar la barra espaciadora. Esto nos resultó con mayor sentido ya que así el usuario no tendría que estar constantemente cambiando entre vistas y resultaría en una mejor experiencia. Por tanto, implementamos unos botones para pausar y reanudar la simulación, que cuando se pulsan activan y desactivan otros ciertos botones. Por ejemplo, cuando se reanuda la simulación, no se permite al usuario chatear o susurrar con los personajes (se desactivan estos botones), pero cuando se pausa la simulación, el botón que se desactiva es el de ejecutar steps de la simulación.

#### 3.3.1.1. Readaptación de Back End para interacción con Front

El trabajo inicial se había pensado para recibir la fuente de información desde un único punto, una terminal. El objetivo principal del trabajo en que nos basámos fué la evaluación de fenómenos sociales, en lugar de desarrollar una aplicación para el público general.

Esto se nota en la intrincada relación que se encuentra entre la implementación del Back, la comunicación con el LLM y la implementación de la arquitectura de personalidad propuesta. Debido al tamaño del proyecto existente una refactorización bien hecha era realmente complicada, por lo que se optó por reutilizar la estructura

existente, que sí permitía la extracción de información generada por el sistema de una forma sencilla (con lo que se podrían añadir extensiones al programa), y ponerle capas por encima para adaptarla a la infraestructura que buscábamos. Con el objetivo de encontrar una interacción con el sistema que fuera más natural.

Esta readaptación conllevó el uso de recursos de sistema como Pipes de comunicación entre procesos de forma nativa, para poder redirigir el flujo de comunicación que la aplicación tenía con la terminal, a un proceso gestionado por el Front que enviase y recibiese la información deseada. La infraestructura propuesta se discute en más detalle en la sección 4.1.

### 3.3.2. Generación de resúmenes

Para la funcionalidad de la generación de resúmenes nos encontramos con dos problemas principales. La selección de la información y la generación del Prompt.

#### 3.3.2.1. Selección de la información

En el proceso de la generación de resúmenes contábamos con toda la información que pudiéramos necesitar. Por lo que el problema principal era decidir cuál de esta usar. Es inviable emplear toda la información generada por cada uno de los Agentes debido a la cantidad de Tokens que se emplearían. Por ello decidimos seleccionar la información en base a ciertos criterios que detallaremos.

Entendemos que la simulación se puede explorar desde dos perspectivas, la global y desde el punto de vista de los Agentes. Es por ello que en el resumen decidimos tener en cuenta ambos datos de forma separada. Por un lado los eventos más relevantes que ocurren en la simulación, por otro las ideas y pensamientos más relevantes que tuvo cada Agente. Con estos dos datos generamos el resumen.

Con este enfoque nos vemos obligados a filtrar mucha información. Para la selección de los eventos más relevantes, tan solo nos enfocamos en los eventos que más pensamientos hayan generado, seleccionando una cantidad de cada Agente.

Por otro lado había que elegir los pensamientos mas relevantes de cada Agente. Esto implica decidir aquellos que hayan tenido una mayor influencia o que hayan sido más elaborados.

En una primera instancia nos encontramos con funcionalidades ya implementadas que eran parecidas a los requisitos que teníamos. Hablamos de las funciones *retrieve\_relevant\_thoughts()* de AssociativeMemory. Sin embargo no era del todo adecuado ya que se centraba en buscar los pensamientos relevantes asociados a unos sujeto, predicado y objeto concretos.

La segunda tentativa fue usar los pesos asociados a los *keywords* de cada ConceptNode. Con esto conseguiríamos ConceptNodes asociados a temáticas muy re-

petidas, sin embargo esta información no nos aporta nada sobre el desarrollo del ConceptNode ni su relación con el resto de pensamientos.

Tras esto decidimos realizar una búsqueda más centrada en la idea de que la relevancia del pensamiento era proporcional a cuán elaborada había sido este a lo largo del tiempo. Al contar para cada ConceptNode con la información de en qué ConceptNodes se basa, teníamos toda la información necesaria para establecer la forma de encontrar los pensamientos más relevantes.

Los criterios concretos para elegir los pensamientos más relevantes han sido los siguientes:

- En primer lugar establecemos un umbral con el que ignoraremos a cualquier ConceptNode que no tenga un score igual o superior a dicho umbral. Dicho umbral se encarga de valorar dos aspectos. El primero y más sencillo es que no permitirá considerar ConceptNodes que den lugar a nuevos pensamientos. El segundo establece un valor numérico mínimo para el score mínimo del ConceptNode.
- El valor del umbral es el siguiente

$$umbral(Node) = Node.depth! * (5 + 2^{Node.depth-1} * 3)$$

El valor de dicho umbral equivale al score de un ConceptNode con *poignancy* 5 y una media de *poignancy* 3 para los antecesores del Nodo, suponiendo un factor de ramificación 2.

- El valor del score para un ConceptNode lo calculamos de la siguiente manera:

$$score(Thgt) = \begin{cases} 0 & \text{si } Thgt = null \\ Thgt.depth * (Thgt.poignancy + \sum_{p \in Thgt.prevs} score(p)) & \text{si } Thgt \neq null \end{cases}$$

Toda la implementación se encuentra en la función *generateSummary()* de la estructura *ReverieServer* que se encuentra en *reverie/backend\_server/reverie.py*

### 3.3.2.2. Generación del Prompt

una vez tenemos la información necesaria lo único que nos queda es plantear el Prompt que presentamos al LLM para obtener el resumen que necesitamos. Para la generación del Prompt seguimos la estructura detallada en la sección 2.5.3.1. Por ello terminamos generando una función que se encargaba de generar el Prompt en base a la plantilla<sup>6</sup> y más tarde llamar a la función correspondiente para el resumen final. Las fases de verificar la respuesta eran triviales, puesto que lo que queríamos era un resumen y por tanto nos bastaba con texto plano.

<sup>6</sup>La plantilla se muestra en el Apéndice B

Este resumen se podría enriquecer con referencias a los momentos de tiempo en que ocurren los hechos detallados, y sería a través de estas funciones de verificación que se comprobaría que las referencias existiesen, por ejemplo.

Finalmente, decidimos asociar un resumen a cada par (simulación, step) de forma que pudiéramos tener varios resúmenes de una simulación en distintos instantes de tiempo, todos accesibles desde la vista *Visualizar Simulaciones* de la Interfaz. Para esto tuvimos que añadir el resumen como información asociada a la simulación. Por ello hubo que modificar ligeramente la forma de guardar y crear simulaciones, que se encargan de generar estos campos en las simulaciones.

### 3.3.3. Enviar información al Back

Al permitir el uso de la aplicación desde una interfaz web, se hizo necesario el uso de un proceso extra que hiciera de servidor que realiza todos los cálculos. Para mantener la comunicación entre estos dos procesos empleamos los Pipes, como detallamos en la sección 4.1. El uso de estos Pipes permite emular el uso de una terminal, a través de la que se envían las peticiones de Front. Esta emulación pone ciertas restricciones a la hora de comunicarse.

Una que nos supuso varios problemas fué que tan solo se podía enviar una línea de información en cada comunicación que se efectuase. Esto afectó a la implementación/rediseño de funcionalidades como chat y susurro. Obligandonos a enviar toda la información comprimida en un único string. Y recibir la información de vuelta de una forma similar.

### 3.3.4. Implementación del chat

Para mantener un chat con los Agentes es necesario establecer un flujo de comunicación entre front y back en el que se mantenga la información de la conversación mantenida.

El principal problema que nos encontramos fué la gestión de la lectura a través de los Pipes con los que se comunican Front y Back.

Por ello esta sección se puede dividir en dos partes. Diseño de la comunicación enfocado en los Pipes y Generar respuesta del Agente.

#### **Diseño de la comunicación enfocado en los Pipes**

Al hacer uso de ese recurso del sistema de la forma en que lo hacemos, nos vemos obligados a enviar un mensaje y recibir una respuesta.

Por ello nos hemos visto obligados a colapsar toda la información de la conversación en un único string. Las preguntas que se hagan generan un string que llega al *ReverieServer* con el siguiente formato:

chat:-Agent Name; Interviewer: Question; Agent Name: Answer; Interviewer: ...; Last question

De esta forma podemos reconstruir la información de la conversación y ofrecemos una respuesta al Front en el formato:

Interviewer: Question; Agent Name: Answer; Interviewer: ...; Interviewer: Last question; Agent Name: Last Answer; LastAnswer

Con la que ofrecemos la respuesta dada en el Front y permitimos mantener guardada la conversación mantenida.

### Generar respuesta del Agente

Con el formato visto previamente tenemos disponibles en el back toda la conversación mantenida y al Agente al que entrevistamos.

De esta forma podemos acceder al Agente y usar sus recuerdos para obtener información relativa a las preguntas y así contestar de una forma coherente.

Hacemos uso de toda esta información desde el código que se encuentra en el archivo *reverie/backend\_server/persona/cognitive\_modules/converse.py*. Concretamente en la función *chat()*.

Aquí empleamos funciones que ya se usaban en el proyecto, *new\_retrieve()*, que se encarga de extraer recuerdos de la persona relevantes en base a eventos o descripciones que le proporciona y *run\_gpt\_prompt\_generate\_next\_convvo\_line()* que se encarga de generar la siguiente respuesta en una conversación.

Con esto tenemos generada la respuesta del Agente entrevistado.

### 3.3.5. Acoplo de los personajes al mapa original

Como se ha mencionado en varias ocasiones, el sistema original no estaba pensado para ser extensible en multitud de aspectos. Uno de los aspectos más importantes es que todos los personajes tenían nombres, apariencias y personalidades predefinidas, lo que no permitía a los usuarios modificar los personajes y sus personalidades, por tanto, se verían encorsetados a siempre utilizar los mismos personajes con las mismas relaciones entre ellos y generar experimentos a partir de esto, lo cual limita muchísimo el uso de esta aplicación.

Uno de los mayores retos que nos propusimos fue despegar a los personajes de sus personalidades inherentes, para así poder hacer que los usuarios elijan las personalidades de cada uno y ver así los fenómenos sociales que aparecían.

Al no estar inicialmente pensado el sistema para esto, todos los nombres de los personajes, sus pensamientos, experiencias y objetos en el mapa, estaban directa-

mente "hardcodeados.<sup>en</sup> el programa. Es decir, que estaban escritos directamente y no se permitía que se cambiasen.

El mayor problema fue hacer que los personajes fuesen agnósticos del mapa en el que se encontraban. El mapa inicial trataba de simular una villa, por lo que cada casa y cada objeto tenía el nombre de la familia o personaje que fuese su dueño, lo cual hacía muy fácil la interacción con los objetos de los distintos personajes. Nuestra solución fue que los personajes mantuviesen los nombres originales, pero que eso no influyese en la simulación, ya que ahora no importarían los objetos dentro de la simulación y los usuarios podrían decidir el contexto en el que se encuentran los personajes, por lo que el mapa servirá ahora simplemente como un contenedor para visualizar la simulación.

### 3.3.6. Problema instalación de requerimientos del proyecto

Al inicio del proyecto, en la primera fase de investigación de la herramienta, instalamos el proyecto en nuestros dispositivos personales y hubieron algunos problemas. El más importante de estos problemas era que inicialmente, el archivo 'requirements.txt' no funcionaba para los Mac con procesador M1.

Tras encontrarnos con este fallo, por problemas de dependencias, estuvimos estancados durante unas semanas, ya que no podíamos ejecutar el sistema y estaba habiendo problemas para solucionarlo por todos los medios.

Primeramente, instalamos un entorno virtual, con la versión de Python exacta que recomendaban los desarrolladores (la 3.9.12) y esto no resultó positivamente. También intentamos resolver las dependencias manualmente, pero surgían otros errores de dependencias al hacer esto, y nunca conseguimos resolverlas todas.

Tras estos fallos y bloqueos, decidimos preguntar a los tutores del TFG para ver qué soluciones alternativas podríamos encontrar, pero no llegamos a una conclusión.

Tras semanas intentando resolver este problema y avanzando en el trabajo por otras vías (escribiendo la memoria, investigando la arquitectura y diseño del sistema...), encontramos que los propios desarrolladores habían publicado un patch en el que se solucionaba este problema, actualizando así el archivo requirements.txt. Al actualizar e implementar este cambio, las dependencias se resolvieron y pudimos comenzar con el trabajo e investigación práctica, con el sistema corriendo en nuestros dispositivos locales y habiendo avanzado en otras materias.

## 3.4. Conclusión final

Tras ver el estado inicial en el que se encontraba el sistema, tanto desde un punto de vista técnico como desde un punto de vista más general, y teniendo en

cuenta todos los problemas encontrados, realizando las reconsideraciones y ajustes necesarios, se termina con un sistema más robusto, usable y amigable.

Por tanto, en la figura 3.2, se aprecia, a grandes rasgos, cómo ha cambiado el sistema desde el punto de vista del diseño a alto nivel, llevando a cabo los cambios previamente mencionados en las secciones pasadas.

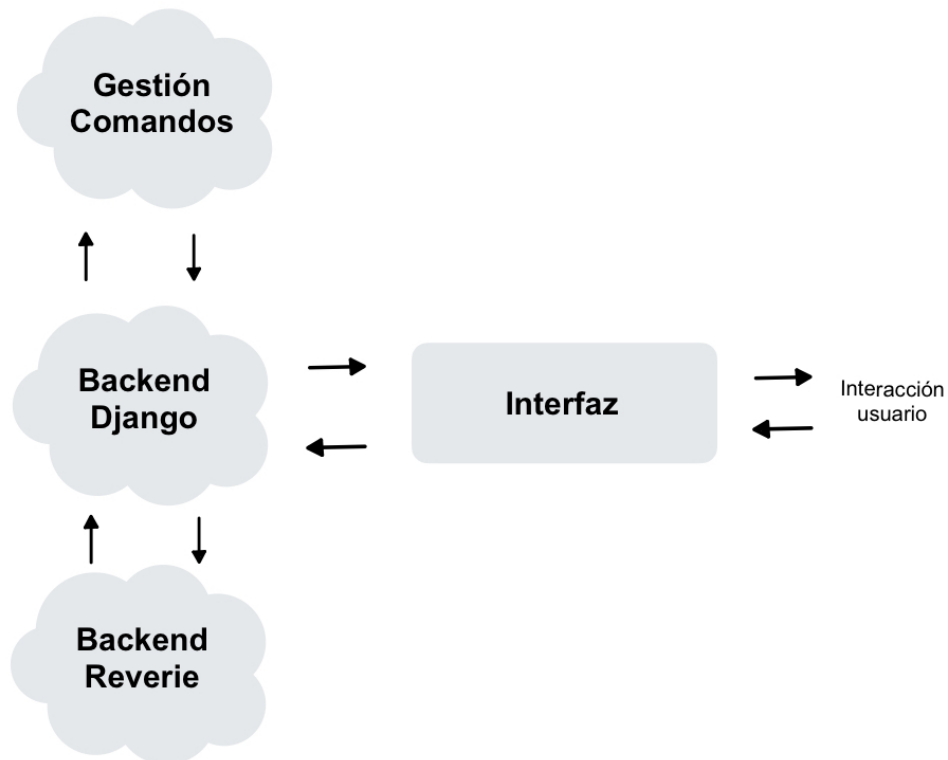


Figura 3.2: Diseño del sistema final tras las adaptaciones

Ahora, como se puede ver, hay un punto único de interacción entre el usuario y el sistema, que es mediante la interfaz. Así, interactuando con la interfaz los usuarios son capaces ahora de hacer lo mismo que antes y más, con las extensiones implementadas.

Mediante esta interacción, los usuarios pueden ejecutar todos los comandos, ver las simulaciones, crearlas e interactuar con ellas libremente.

El flujo de ejecución ahora es algo diferente:

1. El usuario interactúa mediante la interfaz decidiendo la acción que desee.
2. Si el usuario desea ejecutar un comando de la simulación, se captará mediante la interfaz, se redirigirá al backend de django, el cual gestionará la llamada al gestor de comandos (nos permite ejecutar comandos mientras la aplicación sigue funcionando) y esta se encargará de llamar al backend de reverie para que ejecute el comando.

3. El backend de reverie devolverá la ejecución del comando a Django.
4. Al igual que antes, Django procesará esta información y la devolverá a la interfaz, donde el usuario podrá visualizarla y volver a interactuar con esta nueva información.



## Extensiones al Programa Original

Una vez comentadas las motivaciones, objetivos y planteamiento de trabajo, se tratarán los puntos en los que se ha trabajado para lograr la realización del proyecto. En este capítulo se incide en las extensiones y cambios realizados, tratados a bajo nivel y desde un punto de vista técnico.

### 4.1. Adaptación para la ejecución de comandos desde la interfaz

Como se ha comentado en varias ocasiones, siguiendo el objetivo principal de democratizar el uso de esta herramienta para perfiles no tecnológicos como psicólogos u otros profesionales, uno de los desafíos era hacer la aplicación disponible únicamente desde la interfaz, de modo que los usuarios interactúen con la interfaz para realizar todas las acciones.

Originalmente, si los usuarios querían elegir una simulación, ejecutarla o guardarla, por ejemplo, debían ejecutar estas acciones desde la terminal. Esto podría ser una barrera de entrada para ciertos usuarios que no estén familiarizados con estas tecnologías. El objetivo era hacerla accesible desde la interfaz únicamente.

Para conseguirlo, se plantearon ciertas opciones. Tras investigarlas, se tomó la decisión de optar por la opción representada en el diagrama de la figura 4.1, ya que no implicaría cambiar la arquitectura actual de la aplicación, en la cual son los comandos ejecutados en la terminal del backend los que indican las acciones a realizar y marcan las llamadas al modelo de lenguaje.

Como se aprecia en el diagrama, realmente siguen existiendo las llamadas de los comandos, pero se redirecciona para que se ejecuten desde la interacción con la interfaz.

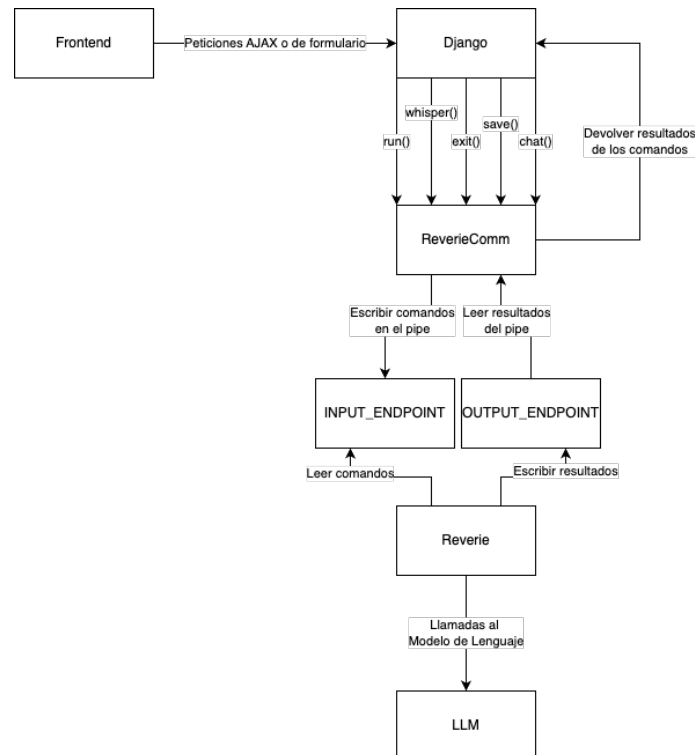


Figura 4.1: Diagrama del estado de los comandos actualmente

El flujo siempre comienza desde la propia interfaz, el FrontEnd. Mediante esta se hacen peticiones a django, que es el primero de los back ends, el que se encarga de cargar los datos, redirigir las vistas y comunicarse con el segundo de los backends (el que hemos llamado Reverie en el diagrama). Tras las llamadas a Django, este se comunicará con una nueva clase, conocida como ReverieComm, para ejecutar los comandos mediante el uso de pipes. La comunicación con estos Pipes se realiza desde el ReverieServer. Pero para permitir esta comunicación sin afectar al funcionamiento de ReverieServer, se tuvo que hacer la redirección de las entradas y salidas estándar a los Pipes correspondientes. Esto se hace por medio del fichero *environment/frontend\_server/endpoint/backendWrapp.py*, que se encarga de gestionar la apertura y cierre de estos Pipes, la redirección y de volcar la información relativa a excepciones en un fichero normal llamado *reverieError*.

De esta manera, logramos solucionar el problema de que los usuarios tengan que interactuar directamente con la terminal y al mismo tiempo no modificamos la arquitectura, ya que los comandos seguirán siendo ejecutados.

## 4.2. Interacción total con el sistema mediante la interfaz gráfica

Uno de los retos para hacer el sistema fácilmente usable fue el cambiar la arquitectura y diseño de la aplicación para que todas las acciones se realizasen a través de la interfaz gráfica.

Consideramos esto algo central, ya que es uno de los pilares del presente trabajo, y algo que realmente aporta valor y ayuda a todos los perfiles como psicólogos, que sin saber cómo utilizar la aplicación, en unos minutos pueden estar experimentando con ella.

Es por ello, que a continuación, en las siguientes tablas, veremos la diferencia de cómo se hacían antes las acciones que ya estaban inicialmente implementadas, y cómo se hacen ahora con las modificaciones (en estas tablas solo se incluyen funcionalidades que ya estaban originalmente disponibles en el sistema original, no las nuevas implementadas en el presente trabajo):

<b>Funcionalidad</b>	<b>Antes</b>	<b>Después</b>
Crear simulación	En la terminal, insertar el nombre de una simulación base (con personajes predeterminados), elegir el nombre de la nueva simulación y crear la simulación	Desde la interfaz, ir a la pestaña de 'Crear simulación', rellenar las personalidades de los personajes deseados y clicar en 'Empezar simulación', automáticamente se redirige a la ejecución

Tabla 4.1: Diferencias entre crear simulación antes y después

Como se puede ver en la tabla 4.1, inicialmente la tarea de crear una simulación era algo más complejo. Los usuarios debían acudir a la terminal, y ahí seleccionar una simulación base, la cual no permitía cambiar el número de personajes ni la personalidad de los mismos. Ahora, mediante la interfaz, se podrá seleccionar fácilmente el número de personajes y la personalidad, contexto y experiencias de cada uno.

<b>Funcionalidad</b>	<b>Antes</b>	<b>Después</b>
Ejecutar simulación	Una vez creada una simulación, ir a la terminal y ejecutar el comando 'run <número de steps>', luego, ir a la interfaz y ver el resultado de la ejecución	En la pantalla de ejecución de simulación, darle al botón de run indicando el número de steps deseados, el resultado se verá en la interfaz directamente

Tabla 4.2: Diferencias entre ejecutar simulación antes y después

En la tabla 4.2 se aprecian las diferencias de cómo era antes ejecutar la simulación y cómo es ahora. Similar a como ocurría al crearla, lo que hemos hecho es evitar que los usuarios tengan que acudir a la terminal a ejecutar los pasos de la simulación, sino que se pueda hacer directamente desde el frontend.

<b>Funcionalidad</b>	<b>Antes</b>	<b>Después</b>
Susurrar a un agente	Con la simulación corriendo, ejecutar en la terminal el comando 'whisper <nombre del personaje><mensaje>'	ir directamente al botón de susurro de cada personaje y añadir el mensaje deseado

Tabla 4.3: Diferencias entre susurrar a un agente antes y después

En la tercera de las tablas, la 4.3, vemos cómo cambia la manera en la que susurramos a los agentes. Ahora es mucho más sencillo, desde la interfaz se buscará el agente con el que nos queremos comunicar y se enviará el mensaje a través de un modal. Antes, sin embargo, era necesario introducir también el nombre del personaje desde la interfaz.

<b>Funcionalidad</b>	<b>Antes</b>	<b>Después</b>
Guardar simulación	En el modo ejecución de simulación, al considerarla terminada, ejecutar el comando 'fin'	Al terminar, desde la página de ejecución de la simulación, tendremos la opción de guardar para ver la demo o guardar para continuar la simulación

Tabla 4.4: Diferencias entre guardar simulación antes y después

En cuanto a las diferencias a la hora de guardar una simulación, se pueden ver en la tabla 4.4. A pesar de que antes era sencillo guardarlas, había que hacerlo desde la terminal también. Ahora, hemos eliminado eso, además de que se podrá elegir entre guardar para ver (redirige a la pestaña de visualizar) y guardar para continuar (que redirigirá a la pantalla de continuar simulación, para que se siga con la ejecución).

<b>Funcionalidad</b>	<b>Antes</b>	<b>Después</b>
Salir sin guardar	En el modo ejecución, si ahora no se quiere guardar la simulación, el usuario debería ejecutar el comando 'exit'	Al finalizar la simulación, si no se desea guardar, en la página de ejecución de simulación se clicará el botón de 'Salir sin guardar'

Tabla 4.5: Diferencias entre salir sin guardar simulación antes y después

Similar a lo que ocurría con el caso de guardar, tenemos el caso de salir sin guardar, representado en la tabla 4.5. En este caso, en lugar de guardar, simplemente salimos de la ejecución de la simulación, sin guardar la simulación y redirigiendo al usuario a la página de landing.

Funcionalidad	Antes	Después
Ver una demo	Ir al fichero 'compress_sim_storage.py' y ejecutar la función compress, seleccionando la simulación deseada que se quiera comprimir. Una vez hecho esto, pegar en el navegador el nombre de la simulación comprimida, así como el step desde el que se quiere visualizar y la velocidad de reproducción, entonces, se mostrará por pantalla	En la página de 'Visualizar simulación' seleccionar la simulación que se desea ver, la velocidad de reproducción y el step desde el que se desea ver. Una vez seleccionado, se redirigirá automáticamente a la página de visualización

Tabla 4.6: Diferencias entre ver una demo de simulación antes y después

Finalmente, el caso de visualizar la demo de una simulación, reflejado en la tabla 4.6. Antes era muy compleja la realización de esta acción. Los usuarios debían ejecutar una función para comprimir las simulaciones, encontrar las simulaciones en su propio dispositivo y pegar la dirección url directamente en el navegador. Esto resultaba muy complicado para el usuario. Ahora, simplemente guardando para visualizar en el futuro, los usuarios podrán acceder desde la página de 'Visualizar simulación' a todas las simulaciones que tengan guardadas y así poder visualizarlas directamente, con un simple click, el resto de acciones son realizadas automáticamente.

### 4.3. Funcionalidad de chat con los agentes en tiempo real

Una de las funcionalidades novedosas que no existían inicialmente en el sistema original es la de chatear con los distintos agentes en tiempo real. Esto es, a medida que se va ejecutando la simulación, los usuarios podrán pausarla e interactuar con cada uno de los agentes mediante un chat. Se dice que este chat es en tiempo real ya que, dependiendo del momento en que el usuario pause la simulación, el agente mantendrá todos los recuerdos y se podrá interactuar con él, dando respuestas basadas en estos recuerdos y conocimientos.

Un caso de ejemplo de esta funcionalidad es si deseamos ver cómo un personaje va madurando y desarrollando una idea a lo largo de la simulación. Inicialmente,

el usuario podría indicar al agente que tiene cierta creencia o pensamiento muy arraigado, y durante la simulación, forzar situaciones para que este agente se vea expuesto a otros puntos de vista que puedan modificar sus pensamientos.

A medida que el agente se ve expuesto a ciertas situaciones, puede ser que cambie parte de sus pensamientos y/o creencias. Por ello, es muy interesante esta funcionalidad de chat.

Si al principio de la simulación el usuario interactuase con el agente, este le diría el pensamiento que tan fuertemente había arraigado a su personalidad antes de comenzar la simulación. Sin embargo, a medida que avanza el tiempo, si se interactúa con el agente otras veces, puede que de respuestas diferentes y se vea cómo las ideas pueden ir cambiando y madurando.

La idea inicial de esta nueva funcionalidad era la de permitir a los usuarios como psicólogos puedan apreciar estos sutiles cambios en las mentalidades y traspasar este conocimiento a casos de humanos reales.

Además, otros casos reales de uso puede ser el estudio del resultado que tendría realizar ciertas acciones en algunas personas con cierta personalidad y ver cómo va avanzando el cambio en sus acciones, para saber qué conversaciones debemos tener con las personas para poder reestructurar su personalidad extrayendo los resultados de este experimento.

## 4.4. Herramientas para evaluar consumo de Tokens con el LLM

Se ha agregado al proyecto ciertas utilidades que permiten evaluar el consumo de tokens por cada tipo de prompt que necesita el sistema. Esto permite evaluar en qué consultas se obtiene la mayor carga de trabajo a la hora de interactuar con el LLM.

Las utilidades en sí requieren de la variable de entorno `volcar=true` para que se generen los archivos con los datos. Dichos archivos se encuentran en `SimulAgents/-reverie/backend_server/logs/tokens.txt`.

Se trata de un archivo `.csv` donde se encuentran los datos en el siguiente formato:

Total de tokens, Tokens Input, Tokens Output, Tipo de Consulta que lo genera

Para la implementación se han añadido funciones en cada llamada al LLM que registran la cantidad de Tokens.

Se ha aprovechado la estructura para generar prompts que se han enviado al

modelo con la respuesta obtenida, dichos prompts se ubicarán en `SimulAgents/reverie/backend_server/logs/prompts/`. Y se empezarán a generar con la variable de entorno `volcar=true`

## 4.5. Funcionalidad de resúmenes de las simulaciones

Otra de las funcionalidades novedosas respecto al sistema inicial es la de generación de resúmenes automáticos al finalizar una simulación. Esta funcionalidad es otra de la que vimos con mayor sentido, ya que permitirá aportar un contexto a las simulaciones guardadas, para poder distinguirlas a simple vista.

Esta funcionalidad se pone en marcha automáticamente cuando el usuario guarda una simulación para visualizarla más tarde. Como se veía en la sección anterior, cuando se guarda una simulación para visualizar la demo, lo que se hace internamente es que la simulación se comprime mediante un script de python, el cual permite que se vean los personajes correctamente y se pueda gestionar la simulación para ser visualizada.

Una vez se comprime la simulación, se aprovecha y se realiza una llamada al modelo de lenguaje, pasando las memorias de todos los personajes con sus puntos de vista. Lo que hará el LLM con toda esta información es extrapolar todo lo que ha ido sucediendo durante la simulación, recogiendo así los momentos más interesantes y generando un resumen que será mostrado con el resto de la información en la vista de visualizar simulación.

El valor fundamental que aporta esta nueva simulación es que, si un usuario guarda varias simulaciones similares y se olvida de lo que había hecho, solamente viendo el nombre sería muy difícil distinguirlas. Sin embargo, con los resúmenes automáticos, podrá leer directamente esta síntesis y así distinguir rápidamente una simulación de la otra.

Estos resúmenes solamente se generan al guardar para simular por dos motivos: el primero es que se comprime la simulación y es más sencillo pasar esta nueva información al modelo de lenguaje, y la segunda es que si un usuario guarda una simulación para continuarla, significa que no la ha terminado completamente, sino que la quiere bifurcar a partir de ese punto, por lo que no se va a generar un resumen de una simulación que no está considerada como terminada.

## 4.6. Creación desde 0 de una simulación

El sistema que habíamos heredado proporcionaba una serie de archivos que representaban ciertas configuraciones predefinidas de personajes para la simulación.

- `base_the_ville_n25` Contiene los archivos predefinidos para una simulación con 25 personajes.
- `base_the_ville_isabella_maria_klaus` Contiene los archivos predefinidos para una simulación con 3 personajes, concretamente los 3 mencionados

Por lo que la creación de una simulación desde 0 como tal no existía, tan solo contabamos con la creación de simulaciones a partir de estas que ya estaban predefinidas, aunque vacías en cuanto a sucesos.

Para ofrecer al usuario la máxima libertad en las simulaciones que quisiera llevar a cabo definimos la información necesaria con la que debería contar una simulación, sin contar con la configuración de los Agentes (que se dejaría a elección del usuario), como sigue:

- **Tiempo actual:** Se inicializó a la fecha en que se creaba la simulación a las 00:00 de la noche.
- **Fecha actual:** Se inicializó a la fecha en que se creaba la simulación.
- **Segundos por timestep:** Se ha fijado la equivalencia de un timestep a 10 segundos de simulación. Variando la equivalencia de timestep a segundos reales en función de las peticiones que se hagan al LLM. La cantidad de peticiones crece linealmente con la cantidad de Agentes simulados
- **Mapa de la simulación:** El mapa que se utiliza es el que recibimos, `The_ville`, pero se deja la puerta abierta a la inclusión de nuevos mapas en futuros desarrollos.

Además de estos parámetros, se necesitan los Agentes que estarán presentes en la misma, estos se crearán a petición del usuario. Permitiendo la configuración del Estilo de vida, personalidad y objetivos vitales. Así tratamos de dar la mayor flexibilidad a la hora de crear el escenario de una simulación y de enfocarlo a recrear o simular las situaciones que sean de interés para el usuario.

Una vez proporcionada la información necesaria se habrá terminado con la creación de una simulación y se podrá empezar con la simulación en sí de la misma forma que ocurre con las simulaciones ya existentes.

## Interfaz

En cuanto a la interfaz de la página web, no de los principales objetivos del presente trabajo era unificar todas las funcionalidades de la aplicación original, así como las extensiones implementadas en este desarrollo, en una única interfaz web, mediante la cual el usuario pueda navegar y acceder a todas las funcionalidades disponibles.

El resultado de este objetivo está recogido en este capítulo. Para alcanzarlo, se han creado vistas completamente nuevas y se han modificado algunas de las vistas previamente existentes para simplificarlas o añadir funcionalidades.

### 5.1. Proceso de diseño de la interfaz

Para desarrollar estas vistas, se ha tenido en cuenta y dado un peso importante al desarrollo utilizando el diseño centrado en el usuario, empleando los métodos de colaboración con potenciales clientes de la aplicación, análisis de otras herramientas similares y adaptando el diseño dependiendo del tamaño del dispositivo, como se explica en una de las guías sobre diseño centrado en el usuario (Pratt y Nunes, 2012). El siguiente paso a tomar, según esta guía, es el de experimentación y validación con los usuarios finales, el cual se realizará una vez la aplicación esté en funcionamiento, pudiendo así adaptar los *workflows*, botones o maquetación a un entorno más adaptado a los usuarios.

Dentro de este ámbito de diseño centrado en el usuario, y como también se mencionó en el capítulo 2 (el Estado de la Cuestión), se ha tenido especial atención a la democratización del uso de esta aplicación, por lo que uno de los objetivos principales con el diseño era implementar una interfaz sencilla, fácil de usar e intuitiva para que cualquier usuario pudiera utilizarla fácilmente.

Para demostrar el uso del diseño centrado en el usuario a lo largo de este proyec-

to, a continuación expondremos un ejemplo en el que se siguen los pasos claros de comprensión de la experiencia de usuario, especificación de las necesidades del usuario, diseño de la solución y evaluación de la misma en el ejemplo concreto del diseño de una simulación disponible en la vista de 'Visualizar Simulaciones', del cual se pueden ver capturas más abajo, en la correspondiente sección 5.2.2, que trata sobre esta vista.

- **Comprensión:** El primer paso para diseñar este apartado, al igual que todos los elementos de nuestra interfaz, fue el de la comprensión de cómo el usuario experimentará con el producto. Visto desde un punto de vista más general, dentro de la página de 'Visualizar simulaciones' habría una lista que contendría las simulaciones, el contenido en estas tarjetas debía ser limitado pero suficientemente explicativo para que el usuario lo comprendiese.
- **Especificación:** Una vez investigamos y comprendimos el flujo de experiencia de usuario para este elemento, y cómo se relaciona con los demás elementos, se especifican los requerimientos de lo que esperaría ver un usuario si interactuase con este elemento. Como ya mencionamos, es importante que esta información sea breve pero explicativa.

Por tanto, se decidió que este elemento contuviese la información básica para distinguir la simulación entre otras. Incluimos en este elemento información sobre el mapa, la simulación sobre la que está realizada, step en el que se desea comenzar la visualización, velocidad de reproducción, botón de inicio y resto de metainformación (como puede ser fecha y hora de inicio o resumen de la simulación).

- **Diseño:** Una vez comprendido el contexto y requerimientos de los usuarios, se pasa a la fase de diseño de la solución en sí. Como ya se conocía el contenido que debía tener, es necesario dividir la información en el reducido espacio de la tarjeta para que resulte atractivo y sencillo.

La decisión que tomamos respecto a esto es indicar el mapa en la sección izquierda, el título en grande, diferenciando cada simulación y acompañado de un enlace a la simulación padre en pequeño. Después, habría dos inputs donde indicar el step y velocidad, junto con el botón de inicio. Por último, se indicaría toda la metainformación y, si esta ocupase múltiples líneas, se permitiría el 'scroll' dentro de la tarjeta, para que no ocupe demasiado espacio vertical.

- **Evaluación:** Finalmente, una vez hemos construido el elemento, con la información especificada y el diseño pertinente, es necesario ratificar todo el conjunto para comprobar la validez. Esto se puede hacer mediante pruebas realizadas por nosotros mismos en etapas de desarrollo, mediante pruebas de conocidos, utilizando la aplicación por su cuenta e informando de los resultados, o con el producto final funcionando, recibiendo feedback de los usuarios reales y adaptando todo el proceso de diseño, cambiando lo necesario en cada paso.

Estos cuatro pasos de diseño forman parte de del proceso de diseño que explica Google en su curso de diseño de UX profesional <sup>1</sup>, y se ven indicados en la figura 5.1.

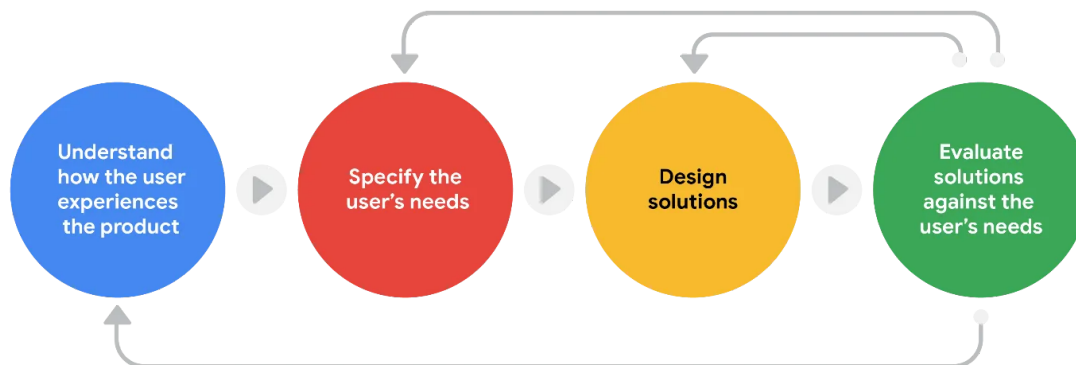


Figura 5.1: Pasos para diseño centrado en el usuario según Google

## 5.2. Nuevas vistas

En esta sección se presentan las vistas creadas durante el desarrollo del trabajo, las cuales sirven de apoyo para ejecutar las funcionalidades, tanto nuevas como existentes, de la aplicación.

Estas vistas no existían previamente en la aplicación, y se ha realizado un proceso de ingeniería inversa de las vistas preexistentes para poder adaptar las interfaces y que se comuniquen de manera efectiva con el back end.

Primeramente, atendiendo a la movilidad entre páginas, se ha implementado una barra de navegación, la cual compartirán todas las páginas y permitirá navegar entre las diferentes funcionalidades, como se puede apreciar en las figuras 5.2, 5.3 y 5.4.



Figura 5.2: Barra de navegación en el formato Desktop



Figura 5.3: Barra de navegación en el formato para dispositivos móviles

<sup>1</sup><https://www.coursera.org/google-certificates/certificado-diseno-de-experiencia-del-usuario-ux>

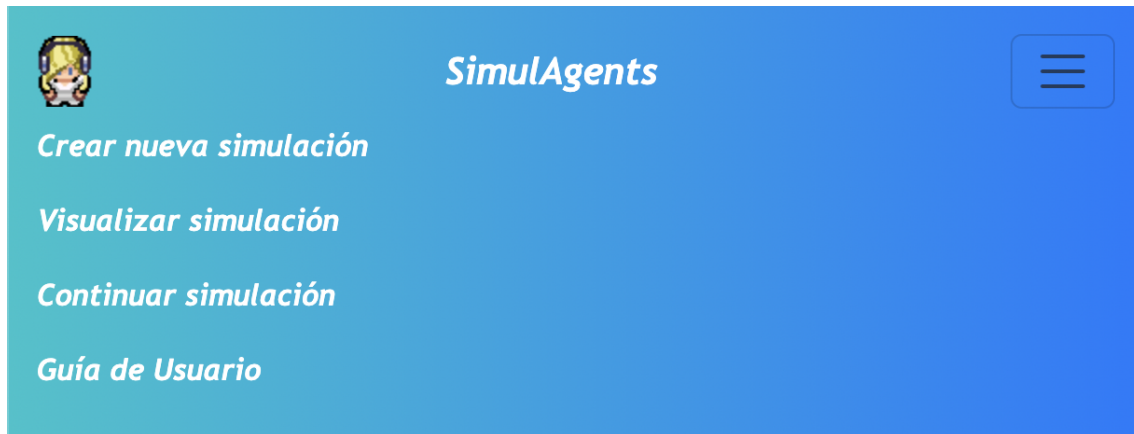


Figura 5.4: Barra de navegación en el formato para dispositivos móviles, con la barra de navegación desplegada

### 5.2.1. Vista de Crear nueva simulación

Teniendo en cuenta los objetivos y las funcionalidades implementadas en este trabajo, es evidente la necesidad de un apartado en el que permitir a los usuarios crear sus propias simulaciones. En esta vista, los usuarios primeramente indicarán el número de personajes que quieren que intervengan en la simulación.

Dependiendo del número de personajes deseados, mediante el uso de la tecnología JQuery, se mostrarán u ocultarán las casillas de relleno de la información de los personajes. Para cada uno de los agentes, los usuarios indicarán el nombre que le quieren dar y la personalidad de los mismos. Como se puede ver en la figura 5.5, el apartado de personalidad viene separado por los siguientes conceptos:

- **Personalidad innata:** El usuario escogerá entre un grupo de personalidades innatas preestablecidas, las cuales darán cierta profundidad a los personajes y constan de una sola palabra, que define simplemente el carácter de cada personaje.
- **Estilo de vida:** Aquí se explicará sobre cómo es el estilo de vida en general del personaje. Se dará una mayor profundidad al carácter, explicando sus hábitos, costumbres, vicios y virtudes.
- **Conocimiento aprendido:** Consistirá en una recopilación de todas las vivencias recopiladas por el personaje que puedan afectar a sus decisiones o su personalidad. Aquí se indicará, por ejemplo, qué tipo de relación tiene con el resto de personajes, si le caen bien o mal, qué experiencias tiene con ellos y demás información relevante para la simulación.
- **Estado actual:** Finalmente, se indicará la acción que está realizando en el momento de comienzo de la simulación, así como sus preocupaciones más inmediatas y las acciones que tomará en los próximos steps.

Con esta información, se podrá inferir la personalidad completa de cada uno de los personajes para así adaptarlos óptimamente a las simulaciones y tener una mayor granularidad al comparar las diferentes simulaciones, cambiando cierta parte de esta información y viendo cómo cambian los comportamientos

Si se quisiera aportar un contexto general a toda la simulación, la solución sería indicarlo en los apartados de conocimiento aprendido y estado actual de todos los personajes. Si todos ellos tienen una pieza de conocimiento aprendido común, podrán inferir la situación en la que se encuentran e interactuar con los demás porque también estarán en la misma situación.

Todos estos elementos gráficos se pueden apreciar en las figuras 5.5 y 5.6, donde se ve que el diseño es 'responsive' y se adapta a todo tipo de pantallas.

**SimulAgents** [Crear nueva simulación](#) [Visualizar simulación](#) [Continuar simulación](#) [Guía de Usuario](#)

## Crear Nueva Simulación

Número de personajes: 2

Nombre de la simulación:  
Nombre de Ejemplo

**Personaje 1**

Nombre del agente:  
Carlos Gomez

Personalidad innata:  
Extrovertido

Conocimiento aprendido:  
Carlos juega en el equipo de fútbol de Salgue C.F. Allí conoció a Abigail, que también es jugadora de la sección femenina del mismo club. Carlos y Abigail son mejores amigos y se cuentan todo lo que les pasa.

Estilo de vida:  
Carlos es una persona activa, le encanta hacer deporte con sus amigos. Es extrovertido y le encanta conocer gente nueva a través del deporte.

Estado actual:  
Actualmente Carlos está jugando un partido de fútbol muy importante, se juegan el ascenso a Segunda División.

**Personaje 2**

Nombre del agente:  
Abigail Chen

Personalidad innata:  
Apasionado



Conocimiento aprendido:  
Abigail juega en la sección femenina del Salgue C.F., donde conoció a Carlos, su mejor amigo.

Estilo de vida:  
Abigail es una chica tímida y amante del deporte, especialmente del fútbol, el cual practica con asiduidad.

Estado actual:  
Actualmente Abigail está viendo el partido de Carlos, en el que se juegan el ascenso a la Segunda División.

[Crear simulación](#)

Figura 5.5: Versión de escritorio de la vista para crear una nueva simulación

 SimulAgents 

## Crear Nueva Simulación

Número de personajes:

Nombre de la simulación:

### Personaje 1

Nombre del agente:

Personalidad innata:

Estilo de vida:  
Carlos es una persona activa, le encanta hacer deporte con sus amigos. Es extrovertido y le encanta conocer gente nueva a través del deporte.

Conocimiento aprendido:  
Carlos juega en el equipo de fútbol de Salgue C.F. Allí conoció a Abigail, que también es jugadora de la sección femenina del mismo club. Carlos y Abigail son mejores amigos y se cuentan todo lo que les pasa.

Estado actual:  
Actualmente Carlos está jugando un partido de fútbol muy importante, se juegan el ascenso a Segunda División.

### Personaje 2

Nombre del agente:

Personalidad innata:

Estilo de vida:  
Abigail es una chica tímida y amante del deporte, especialmente del fútbol, el cual practica con asiduidad.

Conocimiento aprendido:  
Abigail juega en la sección femenina del Salgue C.F., donde conoció a Carlos, su mejor amigo.

Estado actual:  
Actualmente Abigail está viendo el partido de Carlos, en el que se juegan el ascenso a la Segunda División.

Figura 5.6: Versión de móvil de la vista para crear una nueva simulación

## 5.2.2. Vista de Visualizar simulaciones

En esta vista es donde los usuarios pueden seleccionar las simulaciones a visualizar. Esto es, las que se han ejecutado y guardado para visualizarlas previamente. Una vez se selecciona una de estas vistas, se accede a la interfaz de visualización de una simulación, la cual está explicada en profundidad en la subsección 5.3.2.

Como ejemplo, se proporciona la figura 5.7 indicando cómo aparece esta vista representada en la página web. En este caso, también existe un diseño adaptado a dispositivos móviles, pero al ser similar a la versión de escritorio, se omite por simplicidad.

The screenshot displays the 'Visualizar Simulaciones' (View Simulations) page in a desktop format. At the top, there is a blue navigation bar with the 'SimulAgents' logo and several menu items: 'Crear nueva simulación', 'Visualizar simulación', 'Continuar simulación', and 'Guía de Usuario'. The main heading is 'Visualizar Simulaciones'. Below this, there are four simulation cards, each featuring a thumbnail of 'The ville' simulation environment. Each card includes the simulation name, a 'Ver simulación' button, and detailed metadata such as creation and current dates, duration, and a summary status. The first card is 'nuevaSimu', the second is 'July1\_the\_ville\_isabella\_maria\_klaus-step-3-20', the third is 'm', and the fourth is 'l'. The 'm' card has a more detailed summary describing the simulation's content and key steps.

Figura 5.7: Vista de visualizar simulaciones en formato escritorio

Esta era una de las interfaces que no existían y tenían mucho sentido en nuestra aplicación. Antes, el usuario debía interactuar directamente con los ficheros y acceder a sus simulaciones mediante la terminal, ahora esto se gestiona automáticamente y se puede acceder mediante un par de clicks, mejorando así la experiencia de usuario, centrandolo en el usuario final del sistema.

En este listado de simulaciones finalizadas, se muestra información útil para el usuario como el título de la información, el título de la simulación padre (para que el usuario pueda acceder a ella y continuarla o modificarla), el step desde el que se desea comenzar a visualizar la simulación o la velocidad de reproducción de esta.

Además de esta información, también se genera un resumen automático mediante una llamada al LLM cada vez que se guarda una simulación, el cual se incluye en cada una de las simulaciones, indicando a grandes rasgos lo que ocurre y los steps más interesantes a visualizar.

### 5.2.3. Vista de Continuar simulación

La vista de continuar simulación es ciertamente similar a la anterior de visualizar simulación, ya que también habrá una lista con simulaciones. Sin embargo, la diferencia crucial entre las simulaciones que se pueden ver y aquellas que se pueden continuar, es que en las segundas, el usuario podrá interactuar directamente con la simulación y alterar el curso de la misma.

De nuevo, es una vista que tenía sentido que existiese, ya que son funcionalidades que estaban muy escondidas y eran difíciles de ejecutar anteriormente, y lo hemos simplificado, permitiendo acceder a ellas mediante una interfaz intuitiva y fácil de usar.

En este caso, dado el listado de simulaciones, los usuarios tendrán la opción de continuar la simulación o forkearla, comentaremos las diferencias:

Cuando se continúa una simulación, esta sigue con el mismo nombre y se continúa siempre desde el mismo step. Esta funcionalidad ya estaba disponible, y sirve para continuar una simulación desde el punto en el que el usuario lo había dejado, como indica el nombre.

Sin embargo, al forkear una simulación, se creará una nueva, tomando como base la original. Es por este motivo por el cual, cuando un usuario va a forkear una simulación, se le pide el step desde el que quiere forkearla y se pide que se dé un nombre nuevo (ya que será guardada como una nueva simulación en la lista).

En la figura 5.8 se puede apreciar el resultado de esta vista, donde se ven una serie de simulaciones que están actualmente disponibles en la aplicación para continuar, así como el número de steps que estas tienen, el nombre de la nueva simulación y más información sobre la simulación en sí (como la duración o los tiempos de la simulación).



Figura 5.8: Vista de continuar simulaciones en formato escritorio

#### 5.2.4. Vista de Guía de Usuario

Además de las vistas creadas para navegar e interactuar con las simulaciones, se ha creado una página que sirve como guía de usuario, que será de utilidad para nuevos usuarios de la aplicación, resolviendo cualquier tipo de dudas acerca de los procesos que se pueden realizar con la aplicación o explicaciones sobre las vistas existentes y cómo usarlas.

Como el uso de esta herramienta lo hemos orientado a perfiles no técnicos (como psicólogos o personas que quieran estudiar las interacciones sociales), se ha añadido esta sección en la que se explican todas las funcionalidades de la aplicación, cómo utilizarlas y algunos trucos o consejos de uso, para maximizar el valor de la aplicación.

En esta guía se hace un repaso extensivo de las vistas existentes en la aplicación, cómo usarlas y para qué sirve cada uno de los botones.

Algunos ejemplos de esta página son los que se muestran en la figura 5.9

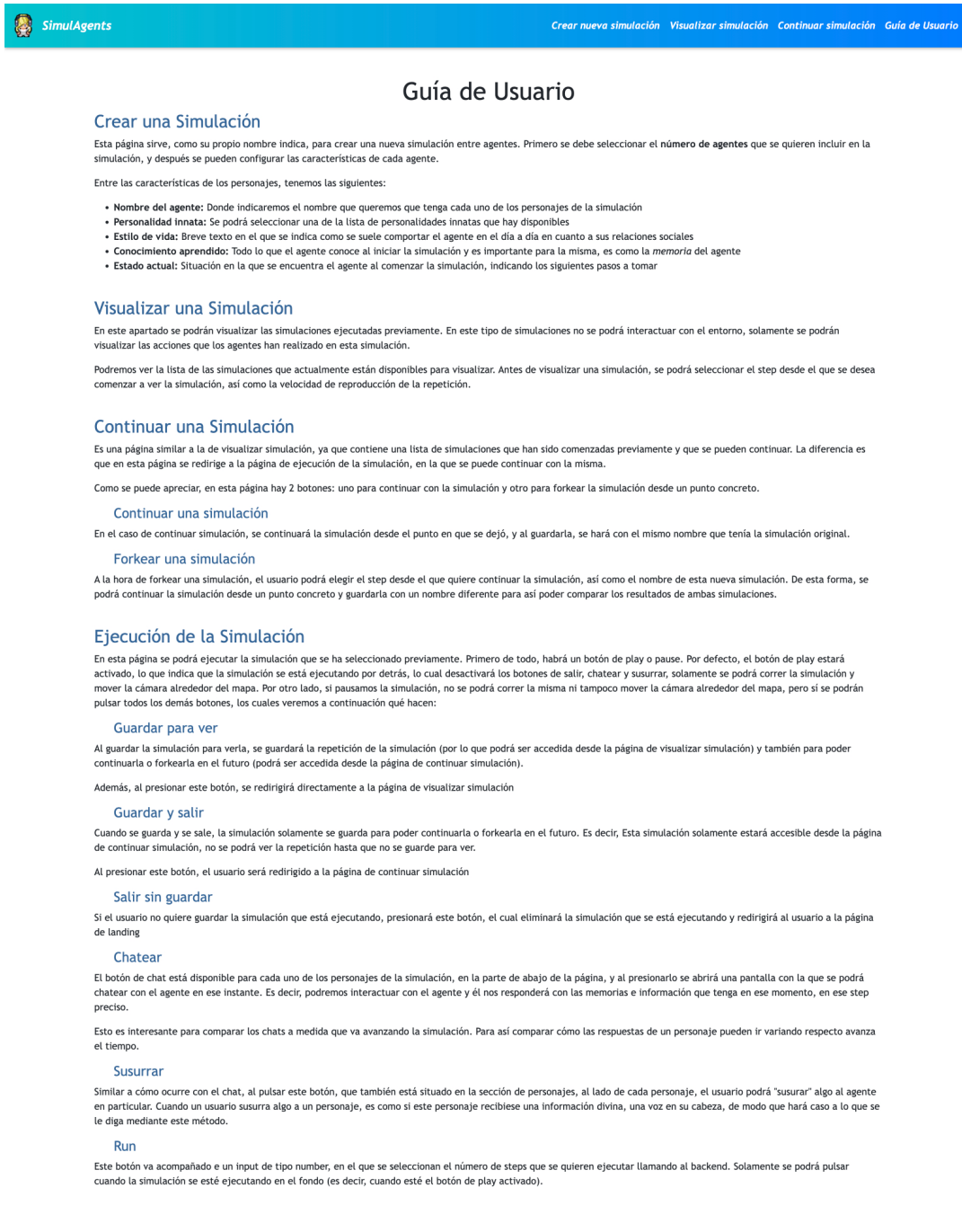


Figura 5.9: Vista de la guía de usuario en versión escritorio

## 5.3. Vistas modificadas

Además de las vistas creadas desde cero, había algunas que ya estaban creadas (como las de la visualización o ejecución de la simulación, que requerían de un motor de videojuego corriendo). Sin embargo, para añadir ciertas funcionalidades, también se han modificado, las cuales se indican en la presente sección.

### 5.3.1. Vista de landing

Esta vista se podría considerar como nueva, ya que fue modificada por completo para el propósito de este trabajo.

Originalmente, cuando el usuario arrancaba el servidor, existía una página de landing a la que se le redirigía, donde solamente se indicaba que el servidor se encontraba en funcionamiento con un texto. Como esta página era muy rudimentaria y tan solo contenía una línea de información, se decidió rediseñar la landing completamente, indicando las principales funcionalidades de la aplicación desde una interfaz intuitiva, sencilla y accesible.

Como se puede apreciar en la figura 5.10, hay una breve explicación del trabajo y algunas tarjetas con la información más importante de lo que puede hacer esta aplicación. Se trata de una página meramente informativa desde la que el usuario puede navegar hacia el resto.

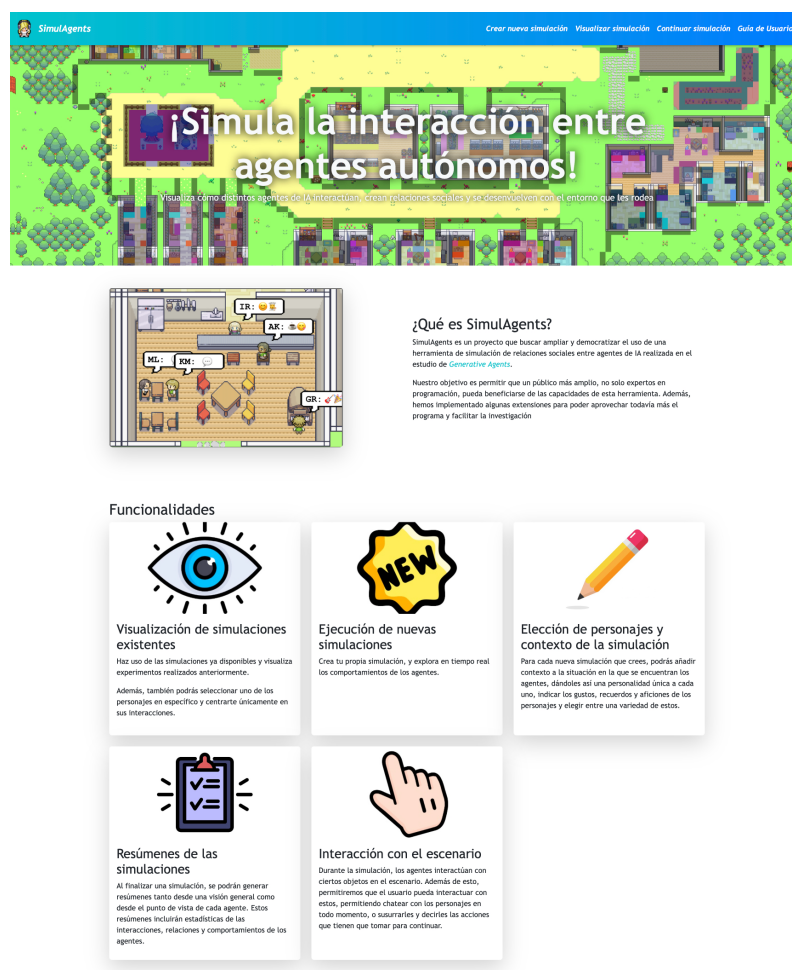


Figura 5.10: Vista de landing actualizada

### 5.3.2. Vista de visualización de una simulación

Esta es una de las dos vistas principales que ya existían en la aplicación inicialmente. Esta vista, así como la de ejecución de simulación, existían para que el usuario pudiese ver la simulación y lo que estaba ocurriendo, no estaban pensadas para que el usuario interactuase con ellas y dejarían de tener sentido una vez se pausase la simulación.

Sin embargo, como hemos centralizado todo en un front end común, estas vistas ahora se convierten en algo crucial. Lo más importante de estas vistas es que cogen la información de la simulación y utilizan el motor de juegos Phaser 3.0 para representar el mapa, los personajes y sus movimientos.

El trabajo que hemos realizado en esta vista fue básicamente el de simplificación de la misma (eliminación de botones para ver el estado en profundidad de los personajes) así como la integración de esta vista con el resto de la aplicación (añadido el navbar para navegar en ella).

Un ejemplo de esta vista es el de la figura 5.11, en la que se visualiza la simulación con nombre 'nuevaSimu', la cual tiene 3 personajes, de los cuales se puede ver su estado y situación en cada momento.

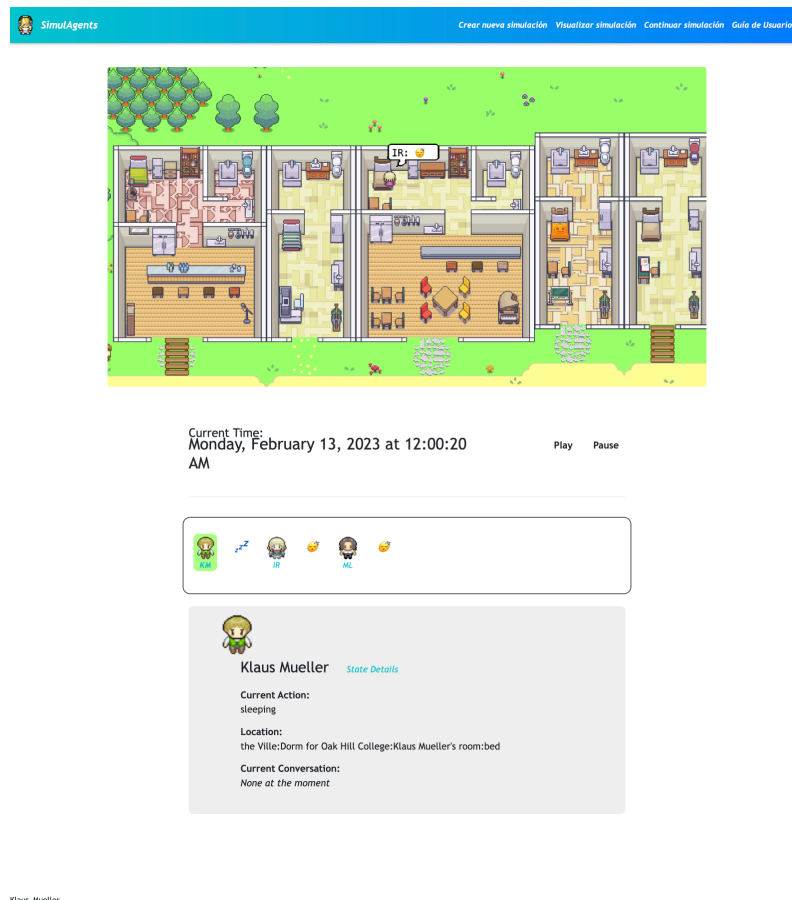


Figura 5.11: Vista de visualización de una simulación

### 5.3.3. Vista de ejecución de simulación

Como ya se mencionó, esta vista es bastante similar a la comentada en el apartado anterior. Era una vista que ya existía, representando la ejecución de una simulación en tiempo real, con el mismo motor de juegos que se usa a la hora de visualizar una simulación, Phaser 3.0.

No obstante, en esta vista sí que hemos modificado varios elementos y añadido funcionalidades críticas en esta vista, para que el usuario pueda tener una mayor interacción con el sistema en tiempo real. Entre estas modificaciones están los botones de guardar, salir, chat y susurro, de los cuales hablaremos a continuación.

Además, hemos decidido eliminar la barra de navegación de esta página, ya que consideramos que el usuario solo debe salir de esta página si es guardando o saliendo de la presente simulación (ya sea guardar para verla, para continuarla o eliminarla).

En la figura 5.12 podemos ver un ejemplo de esta página en ejecución, en la cual vemos una nueva simulación creada por nosotros, donde hemos añadido dos agentes.

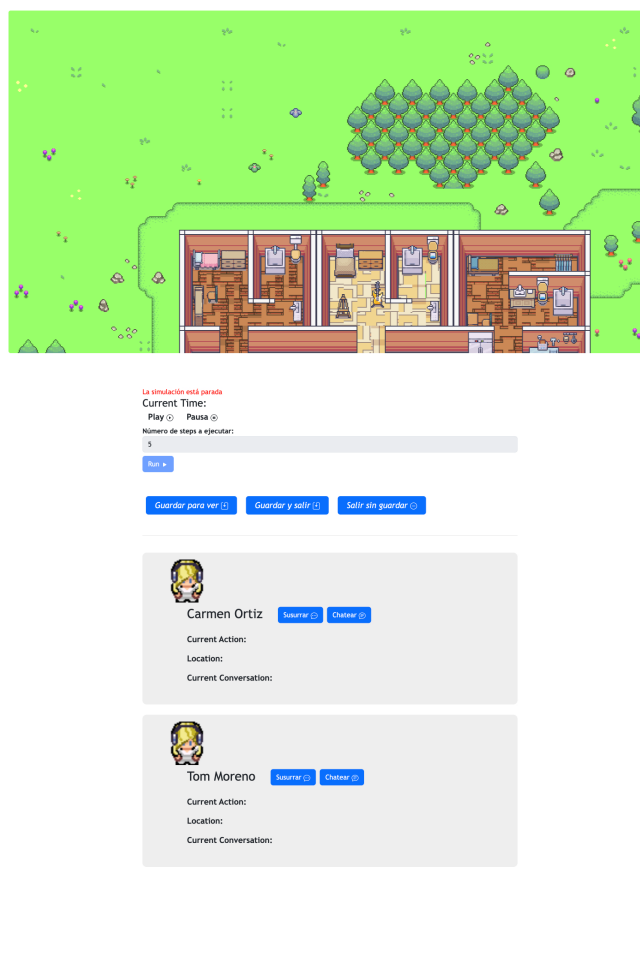


Figura 5.12: Vista de ejecución de una simulación

Las modificaciones que hemos realizado en esta página de ejecución de la simulación se explican a continuación en detalle:

### 5.3.3.1. Botón de 'play'

Para empezar, se distinguen dos estados de la simulación. Cuando la simulación está corriendo en segundo plano, se podrá hacer run de la simulación, pero no se podrá interactuar directamente con los personajes ni guardar ni salir de la simulación.

Cuando se carga la página o cuando el usuario pulsa este botón de 'play', la simulación entrará en este modo, en que está corriendo en segundo plano, y se indicará con el texto en color verde de 'La simulación se está ejecutando', como se muestra en la figura 5.13.

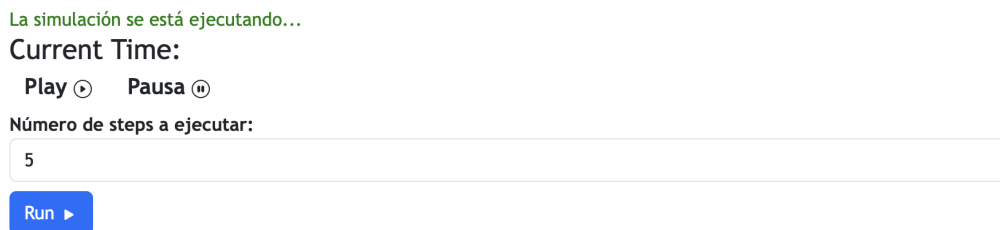


Figura 5.13: Texto indicativo de que la simulación está ejecutando en segundo plano

### 5.3.3.2. Botón de 'pause'

Por otro lado, existe otro estado en que la simulación está parada, por lo que no se puede ejecutar el botón de 'run', pero sí se podrá guardar, salir, chatear o susurrar a los personajes. Esto se puede ver en el apartado encima de «Current time» en la figura 5.12, donde se aprecia en rojo el texto 'La simulación está parada'.

### 5.3.3.3. Botón de 'run'

Antes, para ejecutar steps de una simulación, era necesario ir a la terminal y ejecutar el comando 'run <número de steps>'. Ahora, hemos implementado un botón de 'run', situado al lado de un input de tipo número, que indicará el número de steps a ejecutar.

Por ejemplo, si el usuario quiere ejecutar 3 steps, insertará el número en el input y le dará a play, ejecutando así 3 steps y pudiendo verlos en tiempo real en la ventana que contiene la simulación.

#### 5.3.3.4. Botones de guardado y salida

Una vez el usuario considera que ha terminado la simulación, tiene 3 diferentes opciones, que servirán para distintas finalidades, explicadas a continuación:

- **Guardar para ver:** Si el usuario desea guardar la simulación y que esté disponible entre aquellas que se pueden visualizar (desde la página de visualizar simulación) deberá darle a este botón. Una vez clicado, se guardará la repetición, la simulación también estará disponible para ser continuada y se redirigirá al usuario a la página de "visualizar simulación", para que pueda ver la nueva repetición.
- **Guardar y salir:** Si el usuario quiere guardar la simulación, pero no la repetición, deberá clicar este botón. Al hacerlo, se guardará la simulación para poder continuarla o forkearla, y se redirigirá al usuario a la página de continuar simulación.
- **Salir sin guardar:** Al pulsar este botón, el usuario considera que ha terminado la simulación y no desea guardarla. Por lo tanto, se le redirigirá a la página de landing y la simulación se eliminará del sistema. Esta es una funcionalidad nueva ya que no estaba implementada en el sistema original.

#### 5.3.3.5. Botones de chat y susurro en la vista del personaje

Estos botones están en la sección de personajes. En cada tarjeta de personaje, habrá un botón de chat y otro de susurro. Al clicar en ellos automáticamente se detectará el personaje al cual se quiere susurrar o con el que se quiere chatear en cada caso, y se procederá a ello.

Estas dos funcionalidades son el ejemplo más claro y directo que hay en todo el presente trabajo del uso del procesamiento del lenguaje natural, que se comenta en el capítulo 2 (el Estado de la Cuestión). De todo el sistema, es la única funcionalidad que permite interactuar directamente con el modelo de lenguaje, el cual interpretará el lenguaje natural y lo procesará para devolver una respuesta. En el caso del chat, se comprende la petición del usuario, se interpreta, y se devuelve una respuesta sobre lo que responde el agente. Por otro lado, para el susurro, el usuario interactuará indicando una orden o directriz y el sistema la interpretará para que el usuario la tenga en cuenta y la realice en los siguientes steps de la simulación.

El botón de susurro permite hablar directamente a los pensamientos del personaje e instarle a tomar ciertas decisiones o realizar ciertas acciones. Esta era una funcionalidad preexistente pero la hemos pasado completamente al front. Se puede ver en la figura 5.14, cómo el usuario susurra al personaje de Carmen Ortiz cómo debe reaccionar y actuar a partir de ese momento en la simulación, implementándolo así en su personalidad.

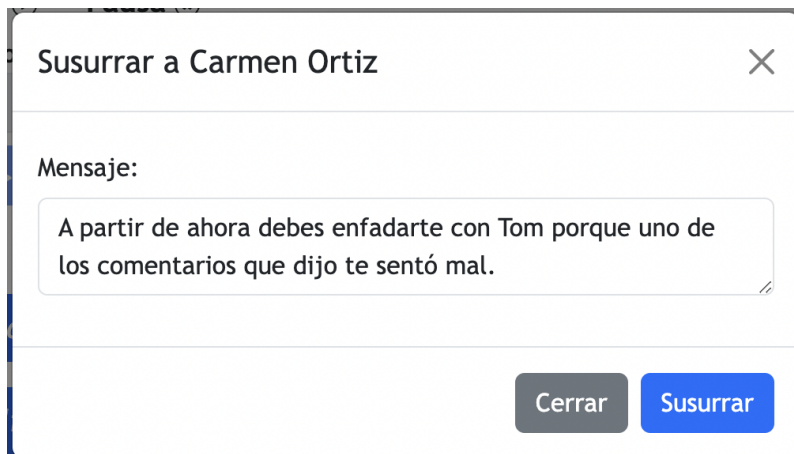


Figura 5.14: Modal de susurro

Por su parte, el botón de chat abrirá una pantalla para chatear directamente con el personaje en tiempo real. Esta funcionalidad es novedosa y servirá para poder entrevistar a los personajes en cada punto de la simulación, viendo así sus cambios de ideas, pensamientos u opiniones a lo largo de la simulación.

En la figura 5.15 se puede apreciar cómo sería la interacción con uno de los personajes de la simulación. Mediante esta interfaz de chat, se interpretaría la entrada del usuario y, mediante procesamiento del lenguaje natural y empleando el LLM, trataríamos de obtener una respuesta coherente a la interacción.

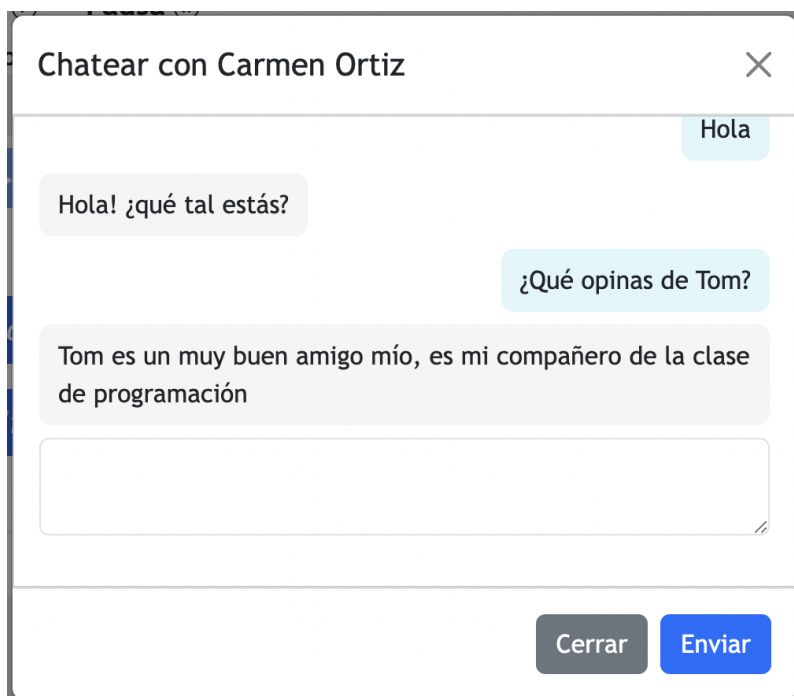


Figura 5.15: Modal de chat con respuestas simuladas

## Conclusiones y Trabajo Futuro

En el presente trabajo se ha desarrollado un sistema que permite el acceso a la simulación multiagente para el estudio de las relaciones sociales e investigación de la emergencia de fenómenos sociales. Esto permitirá que sea una herramienta muy útil para que cualquier tipo de usuario sin conocimiento de informática, como psicólogos (especialmente interesados en las simulaciones sociales) puedan analizar el comportamiento humano simulado y predecir ciertas situaciones.

En este capítulo se describen tres subapartados. El primero de ellos es una especie de discusión en la que, tras exponer el sistema original en el capítulo del Estado de la Cuestión y tras ver las implementaciones y trabajo que hemos realizado a lo largo del resto de capítulos, pretendemos mostrar el impacto que los cambios realizados han tenido en el resultado final del sistema, mostrando así las diferencias en la aplicación desde el principio hasta ahora. En el segundo subapartado se tratan las conclusiones tras analizar el trabajo una vez finalizado. Finalmente, en el último subapartado se profundiza en algunas de las posibles futuras extensiones que se habrían incluido en este trabajo si se hubiese dispuesto del tiempo necesario, ya que sería interesante verlas implementadas en el futuro.

### 6.1. Diferencias entre el Sistema Original y el Actual

Teniendo en cuenta las funcionalidades que existían originalmente en el sistema, discutidas en profundidad en la sección 2.5.2, y conociendo la serie de desarrollos que se han realizado para modificar esta aplicación discutidos en los capítulos anteriores, dividimos estas modificaciones en dos apartados: las funcionalidades ampliadas y las nuevas funcionalidades.

### 6.1.1. Funcionalidades ampliadas

Este primer apartado aborda las funcionalidades que ya se encontraban originalmente desarrolladas. Para estas, existen algunas nuevas características que se han implantado durante el desarrollo de este trabajo que se basan en ellas para funcionar, las cuales se mencionan a continuación.

#### 6.1.1.1. Susurros a los personajes mediante la interfaz

Como se mencionó en el apartado 2.5.2, la funcionalidad de susurrar a los personajes ya estaba implementada en el sistema original. Sin embargo, para poder hacer esto era necesario acudir a la terminal y desde ahí seleccionar el personaje y la información.

Tal y como se ha extendido, se podrá acceder a uno de los personajes clicando en su persona en la parte inferior de la simulación y una vez hecho esto, habrá un botón en el que el usuario se podrá comunicar directamente y susurrar la acción o la personalidad que quiera que interprete el personaje.

Este flujo de acción se ha visto en el capítulo 5 de la interfaz, más específicamente en el apartado 5.3.3.5, donde se puede ver el botón de susurro y el modal resultante al clicar este botón.

#### 6.1.1.2. Nuevas vistas

Además de las vistas existentes, se han creado nuevas páginas, de modo que la aplicación se pueda usar exclusivamente desde la interfaz gráfica, sin necesidad de acceder a la terminal (esto ayuda a usar la aplicación a las personas no técnicas) y uniéndolas todas en diferentes vistas.

Por un lado, se han creado vistas como la landing, una página para crear simulaciones desde la web, y otras páginas para visualizar y continuar simulaciones para poder acceder directamente desde ahí a ellas, simplemente clicando. Por el otro, también se han enriquecido las vistas ya existentes añadiendo botones como los de susurrar o chatear con los agentes durante la simulación.

Además, se ha añadido una barra de navegación mediante la cual se podrá navegar por la aplicación y acceder a todas las vistas necesarias para ejecutar y visualizar simulaciones.

### 6.1.2. Nuevas funcionalidades

Finalmente, existen funcionalidades que no existían en la aplicación original y hemos añadido, aportando así valor al sistema ya existente y reiterando su impor-

tancia para el uso en investigación psicológica.

#### **6.1.2.1. Interacción total mediante la interfaz**

Además de lo visto anteriormente, en cuanto al botón de susurrar a los personajes, y siguiendo con el objetivo principal de acercar esta herramienta a perfiles menos tecnológicos, una adaptación que se ha realizado es que todas las interacciones con la herramienta se pueden desarrollar desde el frontend de la aplicación.

Antes era necesario crear una instancia del backend, mediante la cual conocer los archivos que componían el sistema e interactuar con el programa mediante la terminal. Ahora, los usuarios serán agnósticos de la implementación y cuando se ejecute una simulación, se creará e interactuará con la terminal mediante el frontend. Esta funcionalidad está explicada a bajo nivel en el capítulo 4.

#### **6.1.2.2. Función de chat en tiempo real con los agentes**

Similar a la funcionalidad de susurro, hemos implementado una nueva característica en la que los usuarios podrán, en medio de una simulación, chatear con cualquiera de los personajes para preguntarles su opinión sobre ciertos temas, para ver cómo va avanzando a medida que avanza la simulación.

En este caso, se diferencia del susurro ya que la interacción en el chat no influirá en el comportamiento del agente en la simulación.

Al igual que ocurría con el botón de susurro, el flujo de interacción con el botón y la ventana emergente del chat se puede ver en el apartado 5.3.3.5.

#### **6.1.2.3. Guardado y gestión de nuevas simulaciones**

En cuanto al guardado de simulaciones, tal y como estaba organizado inicialmente, mediante el uso de los comandos se podía salir de la ejecución de la simulación y esta se guardaba en una carpeta dentro del propio sistema del usuario. El problema a la hora de volver a estas simulaciones, tanto para visualizarlas como para continuarlas, era que se guardaban todas en la misma carpeta, y era difícil de recuperarlas, ya que tenías que añadir el nombre exacto de la simulación mediante el uso de la consola.

Ahora, las simulaciones se seguirán guardando en el mismo sitio y reteniendo la misma información, pero para recuperarlas y poder ejecutarlas o verlas, estarán disponibles en una sola vista todas las que el usuario tenga guardadas, y con un simple clic podrá acceder a ellas para realizar la acción que desee. Ya sea continuar con la simulación o visualizarla. Además, en el apartado de visualizar simulación, también habrá un link que redirigirá a los usuarios a la simulación de la que parte

esa repetición, para que así puedan modificarla a su gusto si quisiesen experimentar con ella.

Además, se añade la funcionalidad de la gestión de las nuevas simulaciones, pudiendo crearlas y editarlas fácilmente, así como visualizarlas, ejecutarlas o modificarlas puramente desde la interfaz gráfica.

#### 6.1.2.4. Resúmenes de simulaciones

Para aportar valor y contexto a las simulaciones, se ha añadido la función de la generación automática de resúmenes. Una vez el usuario realiza una simulación y la desea guardar para poder visualizarla en el futuro, dentro del proceso de guardado de la simulación y compresión de los datos, también se generará un resumen automáticamente, llamando al modelo de lenguaje para ello, en el que se indique a alto nivel lo ocurrido durante la simulación y los momentos en los que ocurren las situaciones más interesantes a estudiar (momentos de interacción entre agentes, en los que el usuario ha interactuado con ellos, en los que cambia algo importante en la simulación, etc.) para que el investigador pueda visualizar los momentos más interesantes o modificar la simulación desde ese punto.

Estos resúmenes estarán disponibles una vez se finalice y se guarde la simulación, a modo de indicativo sobre lo que ha ocurrido durante esta a alto nivel. Si tras leer el resumen el usuario quiere modificar la simulación, puede tomar uno de los tiempos de referencia y editar la simulación desde ese punto, para así estudiar las diferencias dependiendo de qué cambie a partir de ciertas modificaciones en determinados puntos clave de la simulación.

## 6.2. Conclusiones

La intención inicial del presente Trabajo de Fin de Grado era la democratización del uso de un sistema de simulaciones multi agentes, el cual ya existía previamente, para que perfiles como psicólogos pudiesen estudiar los fenómenos sociales y experimentar con las relaciones entre diferentes agentes de Inteligencia Artificial.

Para ello, al comienzo de esta memoria, introdujimos los objetivos alrededor de los cuales se centraría el trabajo de desarrollo. En este apartado de conclusiones, reflexionaremos sobre si se han cumplido estos objetivos, explicando brevemente los resultados de los mismos.

Respecto a la creación de simulaciones configurando agentes y/o el escenario, hemos logrado crear simulaciones novedosas permitiendo a los usuarios personalizar los agentes (añadiendo personalidades propias a los mismos y permitiéndoles interactuar en contextos diferentes). Sin embargo, la elección de mapa inicial que se

propuso, finalmente no fue implementada, ya que tomaría mucho tiempo de trabajo de diseño y dibujo, el cual se aleja del propósito de este TFG. Lo que se ha conseguido es que los usuarios sean agnósticos del mapa y puedan interactuar con este sin estar encorsetados en una villa (que era el contexto inicial del mapa).

Tras el desarrollo del proyecto, se permite la interacción con el estado de los agentes, tanto mediante la función de chat como la de susurro. Este era otro de los objetivos marcados y se ha conseguido, implementando la novedosa funcionalidad de chat con los agentes.

El objetivo de visión de la simulación a través de un personaje estaba previamente implementado en la aplicación, y la visión general de la simulación, también. Se ha añadido la funcionalidad de generar resúmenes, como se indica en el tercer objetivo de la memoria, la cual es generada automáticamente por el modelo de lenguaje, y narra lo que ha ido ocurriendo a lo largo de la simulación.

Ahora se permite también el guardado, recuperación y eliminación de las simulaciones, pudiendo gestionar así fácilmente el almacenamiento de las mismas. Además, se ha implementado la funcionalidad de poder continuar simulaciones o crear nuevas a partir de otras ya existentes.

También se han integrado todas estas funcionalidades previamente mencionadas en una interfaz común, la cual centraliza toda la interacción necesaria con la aplicación. A través de ella, los usuarios podrán gestionar y visualizar simulaciones e interactuar con las mismas. La interfaz sirve como punto de entrada para experimentar todas las funciones del sistema.

No se ha podido implementar la integración de la aplicación con otros modelos de lenguaje alternativos a elección del usuario. Sin embargo, se han actualizado todas las llamadas existentes para su funcionamiento con la nueva API de OpenAI (usando el modelo GPT-3.5 turbo) y se ha optimizado el número de llamadas para que la ejecución sea más liviana.

Finalmente, se llegó a un resultado final del sistema en el que logramos cumplir prácticamente todos los objetivos marcados (una vez se reconsideraron los mismos), habiendo así democratizado el uso de esta nueva herramienta para la investigación de los fenómenos sociales con agentes de Inteligencia Artificial. Cabe recalcar también que, a pesar de que el desarrollo está realizado, los costes de operar esta aplicación a día de hoy con la API de OpenAI como la que estamos usando son sumamente elevados. Esto es un problema para todos los sistemas multi agente en general. Si en unos años este coste se reduce o salen modelos lo suficientemente pequeños y potentes para ejecutar este código de manera verosímil, será una gran herramienta sumamente potente, pero las limitaciones tecnológicas a día de hoy no nos permiten demostrar todas las capacidades de la aplicación.

### 6.3. Trabajo futuro

A pesar de que cumplimos los objetivos y requisitos especificados al inicio del desarrollo, también es cierto que a medida que se progresaba con el trabajo, iban surgiendo ciertas ideas que habrían tenido un impacto positivo en el proyecto si hubiese habido suficiente tiempo para implementar todas.

La decisión que tomamos a mitad de desarrollo fue la de priorizar ciertos objetivos y cumplir los que se habían propuesto para el marco de este proyecto. Una vez realizada esta priorización, hubo algunos de los objetivos que se quedaron fuera del desarrollo por falta de tiempo, a pesar de lo interesantes que podrían llegar a ser.

A continuación, se muestra una lista con algunos de los objetivos que nos habría gustado implementar y que son buena idea para realizar trabajos futuros:

- **Elección y cambio de mapa:** Dado el contexto necesario, podríamos hacer que los personajes estuviesen desconectados del mapa. Sin embargo, tal y como estaba inicialmente el sistema, cada personaje se encontraba estrechamente arraigado al mapa, ya que partes del propio mapa tenían los nombres de los personajes. Una vez que se ha conseguido separar a los personajes del mapa, habría sido buena idea crear algún otro mapa alternativo, o permitir a los propios usuarios que creen sus propios mapas. Esto no tuvo tan alta prioridad ya que, como comentamos, los propios usuarios pueden hacer que los agentes sean agnósticos del mapa en el que están y simulen cualquier otro espacio.
- **Personalización de los personajes:** Como se comentó en la sección 3.3.5, el gran problema de los personajes es que estaban directamente ligados al mapa, y el nombre de estos era sumamente importante para guardar las simulaciones y que funcionase correctamente. Sin embargo, al añadir nuevos mapas, también se podría cambiar esto, y permitir a los propios usuarios que elijan los nombres de los personajes, así como elegir o editar cómo se verán cada uno de ellos.
- **Mejora de la interfaz gráfica:** A pesar de que se ha realizado un diseño específico para las funcionalidades de la aplicación, se podría haber experimentado con otra gama de colores o recursos estáticos para estilar las diferentes páginas de la aplicación, y la investigación con esto es sin duda un punto a mejorar en el futuro.
- **Interacción directa con el entorno:** En la aplicación original se permitía modificar ciertas partes del entorno. Es decir, se podía coger un objeto y cambiar su estado (por ejemplo, decir que la cama de un personaje estaba en llamas y ver cómo reaccionaban), sin embargo, traducir esto a la interfaz era sumamente complicado, ya que habría que entrar en la implementación de Phaser, que se reconozca cada objeto con los clics y luego realizar la consulta de la modificación del entorno con el backend. Es un objetivo ambicioso que sería muy interesante de ver y aportaría mucho valor a la aplicación.

- **Libre elección de LLM:** Tras investigar multitud de modelos de lenguaje y probar con ellos, llegamos a la conclusión de que a día de hoy, con las condiciones que tenemos, la opción menos mala para ejecutar este sistema es mediante el uso de la API de OpenAI. Esta no permite realizar muchas llamadas, pero que es una de las pocas que comprende cómo procesar la entrada que enviamos y que funciona bien a día de hoy. Sería interesante en el futuro que los propios usuarios puedan introducir sus modelos de lenguaje que tengan instalados en sus ordenadores, o que puedan utilizar cualquier otra API que tengan a su disposición.
- **Más resúmenes:** Además de los resúmenes que se realizan automáticamente al guardar una simulación, sería buena idea que se guardasen una mayor cantidad de resúmenes para ayudar a los usuarios a ubicar cada simulación. Podría ser una buena idea tener resúmenes que indiquen en qué timestamps ocurren los eventos más interesantes, o que se realice un resumen por cada uno de los personajes de la aplicación automáticamente.

- **Bifurcación de simulaciones en cualquier instante de tiempo**

Con el objetivo de brindar una herramienta que permita crear situaciones pseudo-sociales realistas con el fin de analizar los fenómenos de diversidad de situaciones, vimos un interés singular en permitir la bifurcación de las simulaciones ya hechas en cualquier momento del tiempo.

Con esto se logra la posibilidad de explorar escenarios alternativos desde puntos críticos en el tiempo que, unido a la síntesis ofrecida por los resúmenes, abre la puerta a realizar simulaciones que generen situaciones imprevistas, aunque de interés, y que una vez resumidas sean detectadas y objeto de bifurcación para observar el comportamiento del sistema ante cambios provistos por el usuario.

La implementación de esta funcionalidad implica guardar la información necesaria del sistema en cada momento del tiempo. De forma que se pueda replicar cualquier momento del tiempo del sistema.

La forma ingenua de hacerlo es replicar la información existente en cada momento del tiempo. Este enfoque es imposible en la práctica debido al tamaño de memoria que ocuparía una sola simulación. Estimando un poco, en base a una simulación de 3 agentes y un total de 24 horas, es decir, 8640 steps y ocupando un total de 57 MB. Si asumimos que el crecimiento en espacio es lineal respecto al número de steps, y lo es, esto implicaría un coste cuadrático de espacio en el tiempo simulado. Llevando la simulación de 24 horas de tiempo a aproximadamente 1653 MB de espacio. Lo cual no es viable, más aún habiendo mejores formas de plantear la solución.



# Conclusions and Future Work

In the present work, a system has been developed that allows access to multi-agent simulation for the study of social relationships and the investigation of the emergence of social phenomena. This will make it a very useful tool for any type of user without computer knowledge, such as psychologists (especially interested in social simulations), to analyze simulated human behavior and predict certain situations.

This chapter describes three subsections. The first one is a kind of discussion where, after presenting the original system in the State of the Art chapter and after seeing the implementations and work we have done throughout the rest of the chapters, we aim to show the impact that the changes made have had on the final result of the system, thus showing the differences in the application from the beginning to now. The second subsection deals with the conclusions after analyzing the work once completed. Finally, the last subsection delves into some of the possible future extensions that would have been included in this work if there had been enough time, as it would be interesting to see them implemented in the future.

## Differences between the Original and Current System

Considering the functionalities that originally existed in the system, discussed in depth in section 2.5.2, and knowing the series of developments that have been made to modify this application discussed in the previous chapters, we divide these modifications into two sections: extended functionalities and new functionalities.

### Extended functionalities

This first section addresses the functionalities that were already originally developed. For these, there are some new features that have been implemented during the development of this work that are based on them to function, which are mentioned below.

### **Whispers to characters through the interface**

As mentioned in section 2.5.2, the functionality of whispering to the characters was already implemented in the original system. However, to do this, it was necessary to go to the terminal and from there select the character and the information.

As extended, one will be able to access one of the characters by clicking on their persona at the bottom of the simulation and once this is done, there will be a button where the user can directly communicate and whisper the action or personality they want the character to interpret.

This action flow is seen in Chapter 5 of the interface, more specifically in section 5.3.3.5, where the whisper button and the resulting modal upon clicking this button can be seen.

### **New views**

In addition to the existing views, new pages have been created so that the application can be used exclusively from the graphical interface, without the need to access the terminal (this helps non-technical people use the application) and unifying them all in different views.

On one hand, views such as the landing page, a page to create simulations from the web, and other pages to view and continue simulations have been created so that one can directly access them by simply clicking. On the other hand, existing views have also been enriched by adding buttons such as whispering or chatting with agents during the simulation.

Furthermore, a navigation bar has been added through which one can navigate the application and access all the necessary views to execute and visualize simulations.

### **New functionalities**

Finally, there are functionalities that did not exist in the original application and we have added, thus adding value to the existing system and reiterating its importance for use in psychological research.

### **Total interaction through the interface**

In addition to what was seen earlier regarding the whisper button to the characters, and continuing with the main goal of bringing this tool closer to less technological profiles, an adaptation that has been made is that all interactions with the tool can be developed from the frontend of the application.

Previously, it was necessary to create a backend instance to know the files that made up the system and interact with the program through the terminal. Now, users will be agnostic of the implementation, and when a simulation is executed, it will be created and interacted with through the terminal via the frontend. This functionality is explained in low detail in chapter 4.

### **Real-time chat function with agents**

Similar to the whisper functionality, we have implemented a new feature where users can, in the middle of a simulation, chat with any of the characters to ask their opinion on certain topics, to see how they progress as the simulation advances.

In this case, it differs from the whisper as the chat interaction will not influence the agent's behavior in the simulation.

As with the whisper button, the interaction flow with the chat button and the pop-up window can be seen in section 5.3.3.5.

### **Saving and managing new simulations**

Regarding the saving of simulations, as it was initially organized, by using commands, one could exit the simulation execution, and it would be saved in a folder within the user's own system. The problem when returning to these simulations, both to view and continue them, was that they were all saved in the same folder, making them difficult to retrieve, as one had to add the exact name of the simulation through the console.

Now, simulations will continue to be saved in the same place and retain the same information, but to retrieve and execute or view them, they will be available in a single view, and with a simple click, one can access them to perform the desired action. Whether to continue the simulation or view it. Additionally, in the simulation viewing section, there will also be a link that redirects users to the simulation from which that replay originates, so they can modify it as they wish if they want to experiment with it.

Furthermore, the functionality of managing new simulations has been added, allowing them to be easily created and edited, as well as viewed, executed, or modified purely from the graphical interface.

### **Simulation summaries**

To add value and context to the simulations, an automatic summary generation function has been added. Once the user performs a simulation and wants to save it to view it in the future, within the process of saving the simulation and compressing the

data, a summary will also be generated automatically, calling the language model for this purpose, indicating at a high level what happened during the simulation and the moments when the most interesting situations to study occur (moments of interaction between agents, where the user interacted with them, where something important changes in the simulation, etc.) so that the researcher can visualize the most interesting moments or modify the simulation from that point.

These summaries will be available once the simulation is completed and saved, as an indication of what happened during it at a high level. If after reading the summary, the user wants to modify the simulation, they can take one of the reference times and edit the simulation from that point to study the differences depending on what changes from certain modifications at key moments in the simulation.

## Conclusions

The initial intention of this Final Degree Project was to democratize the use of a multi-agent simulation system, which already existed previously, so that profiles such as psychologists could study social phenomena and experiment with relationships between different Artificial Intelligence agents.

To this end, at the beginning of this report, we introduced the objectives around which the development work would focus. In this conclusions section, we will reflect on whether these objectives have been met, briefly explaining the results.

Regarding the creation of simulations by configuring agents and/or the scenario, we have managed to create novel simulations allowing users to customize agents (adding their own personalities and allowing them to interact in different contexts). However, the initial map choice that was proposed was not ultimately implemented, as it would take a lot of design and drawing work, which deviates from the purpose of this FYP. What has been achieved is that users are agnostic to the map and can interact with it without being confined to a village (which was the initial context of the map).

After the project's development, interaction with the agents' state is allowed, both through the chat function and the whisper function. This was another marked objective and has been achieved by implementing the novel chat functionality with agents.

The objective of viewing the simulation through a character was previously implemented in the application, and the overall simulation view was also. The functionality of summary implementation has been added, as indicated in the third objective of the report, which is automatically generated by the language model and narrates what has been happening throughout the simulation.

Now, saving, retrieving, and deleting simulations is also allowed, thus easily managing their storage. Moreover, the functionality of continuing simulations or

creating new ones from already existing ones has been implemented.

All these previously mentioned functionalities have also been integrated into a common interface. This interface centralizes all the necessary interaction with the application. Through it, users can manage and view simulations and interact with them. The interface serves as an entry point to experience all the system's functions.

The integration of the application with other language models alternative to the user's choice has not been implemented. However, all existing calls have been updated for their operation with the new OpenAI API (using the GPT-3.5 turbo model), and the number of calls has been optimized to make execution lighter.

Finally, a final result of the system was reached where we managed to meet almost all the marked objectives (once reconsidered), thus democratizing the use of this new tool for the investigation of social phenomena with Artificial Intelligence agents. It is also worth noting that, despite the development being completed, the costs of operating this application today with the OpenAI API we are using are extremely high. This is a problem for all multi-agent systems in general. If in a few years this cost is reduced or models small and powerful enough to execute this code credibly are released, it will be a highly potent tool, but today's technological limitations do not allow us to demonstrate all the application's capabilities.

## Future work

Despite meeting the objectives and requirements specified at the beginning of the development, it is also true that as the work progressed, certain ideas emerged that would have had a positive impact on the project if there had been enough time to implement them all.

The decision we made midway through development was to prioritize certain objectives and fulfill those that had been proposed within the framework of this project. Once this prioritization was done, some objectives were left out of the development due to lack of time, despite how interesting they could have been.

Below is a list of some of the objectives we would have liked to implement and that are good ideas for future work:

- **Map selection and change:** Given the necessary context, we could make the characters independent of the map. However, as the system was initially designed, each character was closely tied to the map, as parts of the map had the names of the characters. Once the characters were separated from the map, it would have been a good idea to create another alternative map or allow users to create their own maps. This was not given high priority since, as we mentioned, users can make the agents map-agnostic and simulate any other space.

- **Character customization:** As mentioned in section 3.3.5, the main problem with the characters was that they were directly tied to the map, and their names were crucial for saving simulations and ensuring they worked correctly. However, by adding new maps, this could also be changed, allowing users to choose character names and decide or edit how each one will look.
- **Improving the graphical interface:** Although a specific design was created for the application's functionalities, experimenting with a different color scheme or static resources to style the various pages of the application could have been done, and investigating this is certainly a point to improve in the future.
- **Direct interaction with the environment:** In the original application, it was possible to modify certain parts of the environment. For example, one could take an object and change its state (e.g., say a character's bed was on fire and see how they reacted). However, translating this to the interface was quite complicated, as it would require delving into Phaser's implementation, recognizing each object with clicks, and then performing the environment modification query with the backend. It is an ambitious goal that would be very interesting to see and would add significant value to the application.
- **Free choice of LLM:** After investigating and testing many language models, we concluded that, with the current conditions, the least bad option to run this system is using the OpenAI API. This does not allow for many calls, but it is one of the few that can process the input we send and works well today. It would be interesting in the future for users to input their language models installed on their computers or to use any other API they have at their disposal.
- **More summaries:** Besides the summaries automatically generated when saving a simulation, it would be a good idea to save a greater number of summaries to help users locate each simulation. It could be beneficial to have summaries that indicate at which timestamps the most interesting events occur or to generate a summary for each of the application's characters automatically.
- **Branching simulations at any moment in time:** To provide a tool that creates realistic pseudo-social situations for analyzing the diversity of situations phenomena, we saw a unique interest in allowing the branching of already made simulations at any moment in time.

This achieves the possibility of exploring alternative scenarios from critical points in time, which, combined with the synthesis provided by summaries, opens the door to creating simulations that generate unforeseen yet interesting situations. Once summarized, these can be detected and branched to observe the system's behavior when changes are introduced by the user.

The implementation of this functionality involves saving the necessary system information at each moment in time so that any system moment can be replicated.

---

The naive way to do this is to replicate the existing information at each moment in time. This approach is impractical due to the memory size a single simulation would occupy. Estimating a bit, based on a simulation of 3 agents and a total of 24 hours, i.e., 8640 steps, occupying a total of 57 MB. If we assume linear growth in space concerning the number of steps, which it is, this would imply a quadratic cost in space in simulated time. Taking the 24-hour simulation to approximately 1653 MB of space. This is not feasible, especially considering better ways to approach the solution.



# Contribuciones Personales

Como este no es un trabajo unipersonal, se ha añadido este capítulo de contribuciones personales, en el cual cada uno de los dos alumnos que hemos realizado este trabajo añadirá en qué apartados nos hemos enfocado cada uno.

## Estudiante 1: Alberto Ramos Suárez

Tras la división del trabajo, decidimos que yo realizara la parte de frontend, enfocándome más en la interfaz y todo lo relacionado con la interacción del cliente con la página, así como la interacción con el backend, siendo algo más agnóstico de los procesos que tenían lugar por detrás.

A continuación, enunciaré algunos de los puntos en los que más me enfoqué en mi trabajo y el desarrollo del proyecto.

### Diseño de la interfaz

Todo el frontend está realizado por nosotros (exceptuando los mapas y algunas secciones de las páginas de ejecución de simulación y de demo). Por tanto, esto necesitaba ser diseñado y pensado siguiendo ciertos patrones, para que los usuarios pudiesen interactuar completamente con la web.

Lo primero que diseñé fue la landing page, a la que los usuarios llegarán al cargar la aplicación. En esta, se muestra información sobre el proyecto que hemos desarrollado, algunas de las características que tiene la aplicación y el propósito de la misma.

Después de esto, era necesaria una barra de navegación, que permitiera a los usuarios moverse entre las distintas páginas de la aplicación fácilmente. Esta barra de navegación permite saltar a las páginas de creación, continuación y visualización de simulación, la guía de usuario y la landing.

Las vistas de continuación y visualización de simulaciones son similares, ya que ambas cargan las simulaciones disponibles en el backend, y permiten elegir una de ellas para las acciones que se deseen tomar. El diseño de estas fue pensado para que los usuarios tuvieran libertad a la hora de elegir qué parte de la simulación quieren ver y de dónde provienen las simulaciones, tratando de hacer el proceso lo más intuitivo posible.

En cuanto a la vista de creación de simulación, es una de las más complejas del sistema, ya que es donde los usuarios tendrán que definir varios parámetros para que las simulaciones funcionen sin problemas. Entre estos parámetros están el contexto, personalidad y actitud de cada uno de los personajes, o el contexto de la simulación.

La vista de guía de usuario es muy sencilla y está pensada para enseñar a los usuarios a usar la aplicación, así como dar trucos sobre cómo optimizar las ejecuciones de simulaciones y experimentos interesantes.

Para conseguir implementar todas estas vistas, también fue necesaria la investigación a fondo del funcionamiento de la aplicación. Comprendiendo el backend de Django para crear nuevas vistas, así como el flujo de información desde y hacia este.

## Contribuciones a la memoria

Ambos tuvimos implicación en esta memoria. Sin embargo, como Kevin tenía una mayor carga en el área de investigación e implementación de las funcionalidades en el backend, me encargué de hacer varias de las secciones que no tenían "dueño", así como la mayoría de las imágenes que sirven para explicar gráficamente los aspectos del diseño de la aplicación en general.

Además, la sección del Estado de la Cuestión la dividimos en 2. Decidimos que yo hiciera las secciones de la 4 a la 7, ya que eran secciones que tenían menos que ver con la lógica por detrás de la aplicación, y más con el propósito general de este trabajo, que es la democratización y accesibilidad de una herramienta para experimentar con la emergencia de relaciones sociales mediante el uso de tecnologías multiagente.

El capítulo 6 de la memoria, en el cual se tratan las interfaces, también lo realicé yo, ya que es esencialmente el trabajo en el que invertí la mayor parte de mi tiempo. Este capítulo trata esencialmente de las vistas en detalle, mostrando ejemplos del resultado final de las mismas y explicándolas.

Sin embargo, aunque nos hayamos dividido las diferentes secciones de la memoria, tanto Kevin como yo revisamos las secciones realizadas por el otro, para comprobar que no haya errores ortográficos y que la información sea validada por ambos antes de publicarla.

## Simplificación del repositorio

Una de las propuestas que realicé y sobre la que tomé acción personalmente fue la de la eliminación y limpieza de ciertos archivos y carpetas del repositorio. Como se detalla en la sección 3.2.2, existían multitud de carpetas y archivos, así como algunas funciones dentro de archivos, que no hacían nada a día de hoy en la aplicación. Muchos de estos eran usados para depurar la aplicación pero no están siendo utilizados y por tanto, decidí eliminarlos, explicando en esa sección el porqué.

Parte de la intención de hacer esto es que el repositorio quede en general más limpio. Permitiendo así que sea más sencillo en el futuro comprender a los desarrolladores qué hace cada función, y que no sea tan difícil encontrar la información valiosa.

## Actualización de la documentación

Otra de las propuestas en las que tomé la iniciativa de implementar fue la creación de una documentación para utilizar la aplicación en castellano y optimizada para el trabajo que habíamos realizado.

La documentación que había inicialmente estaba escrita en inglés y pensada para los usuarios que utilizaban el sistema como estaba inicialmente diseñado. Sin embargo, nosotros hemos cambiado la forma de interactuar con el sistema y por ello sentía que era necesaria la creación de una documentación que explique cómo hacer funcionar esta aplicación.

Al hilo de esto, también decidí incorporar la guía de usuario en la interfaz. La idea principal es que la documentación será necesaria para ejecutar la aplicación e incluirá las especificaciones técnicas de la misma, mientras que la guía de usuario explicará exclusivamente cómo utilizar la aplicación para usuarios nuevos, así como tips y trucos interesantes para realizar experimentos con la aplicación.

## Gestión de la comunicación entre el frontend y el backend de Django

En esta parte del desarrollo contribuimos tanto Kevin como yo. Se trata básicamente del contrato de API entre el front y el back, mediante el cual el frontend le pedirá al backend información, la cual este deberá ejecutar y devolver la respuesta.

En el desarrollo de este contrato intervinimos los dos, pero yo tomé la iniciativa de crear las primeras llamadas (de los comandos, de la creación de simulaciones...) y a partir de ahí luego veíamos qué más información necesitábamos y en qué formato.

## Estudiante 2: Kevin Óscar Arce Vera

Mi trabajo se enfocó en digerir toda la estructura lógica que habíamos heredado, con el objetivo de comprender la lógica que se había seguido para implementar todo lo pretendido en Park et al. (2023). De esta forma era posible ampliar el trabajo existente con las funcionalidades que hemos discutido y desarrollado en este documento. Entre las actividades en las que más tiempo he invertido se encuentran: la valoración de LLMs alternativos al que se usaba, adaptar la infraestructura usada en backend para lograr el uso de la aplicación a través de una interfaz web, implementación de funcionalidades, adaptación de las existentes a nuestros requisitos, redacción de memoria y gestionar la comunicación entre backed y frontend.

Procedo a detallar cada una de estas actividades a continuación:

### Valoración del uso de LLMs

Como se comentó en la sección 3.2.3, era necesario contar con un LLM que, en el corto plazo, nos permitiera probar las funcionalidades que fuésemos agregando y, en el largo, nos ofreciera la capacidad de ejecutar la aplicación sin mucho tiempo de inferencia, poco coste y de forma local, en el caso ideal.

Por ello surgió el problema de la búsqueda de un LLM, del cual me encargué. La primera de las aproximaciones que planteé fue la de usar el mismo modelo que ya funcionaban en la aplicación, por lo menos durante el desarrollo. Pero debido a la ignorancia del coste que pudiera acarrear y la existencia de modelos Open Source, decidí probar con otros modelos.

La primera de las alternativas que surgieron fue, el recién estrenado, Llama 2. La principal razón era su equidad a modelos que eran referencia en ese momento y también por la variedad de tamaños que ofrecían. La prueba de este modelo la realicé a través de Repositorios que permitían cargar modelos para los que habían adaptado la herramienta, permitiéndome evaluar la proximidad de este modelo a los requisitos que exigía nuestra aplicación. Como ya comenté, los diversos problemas que surgían del uso de Llama 2 me decantaron por otras alternativas.

La siguiente fue emplear modelos que habían sido cuantizados, de los que encontré multitud en HuggingFace y, que por suerte, ofrecían versiones de Llama 2, con los que, en principio, se podían solventar los problemas de latencia.

Las prueba de proximidad a los requisitos de la aplicación las realicé manualmente, elaborando consultas tipo que se realizarían al modelo a través de la aplicación y ofreciéndoselas a los modelos que estaba evaluando. Para comprobar si eran capaces de estructurar la información de la forma adecuada.

Como ya se vio, no fue posible. En este punto me encontraba sin demasiadas alternativas a modelos Privativos. Hasta que encontré el paper Eldan y Li (2023)

. Que ofrecía modelos bastante pequeños, en comparación a los LLM que habían surgido, y con un entrenamiento distinto al de los LLMs convencionales, que parecía ajustarse al tipo de actividad que se realiza en nuestra herramienta y capaces de generar texto con una gramática en inglés rozando el nivel humano. Por ello probé varios de estos modelos por medio de la respuesta a consultas sencillas que les iba haciendo a través de la misma herramienta que usé con Llama 2 y los modelos cuantizados.. Pruebas en las que se vieron las grandes carencias generativas que tenían.

También me encargué de la adaptación de las consultas del modelo para que fuera capaz de utilizar la API de PaLM, el LLM de Google en aquel entonces. Teniendo que bordear las restricciones de su uso en España. Finalmente de forma infructuosa debido a que era bastante deficiente también.

Por ello terminamos usando la API de GPT3.5 durante toda la fase de desarrollo. Lo que nos llevó a sufrir la actualización de la API de OpenAI por la que tuve que realizar una migración del código que se comunicaba con la API para emplear la nueva librería.

## Adaptación de la estructura existente a la nueva interfaz

El trabajo sobre el que empezamos a trabajar ya tenía una aplicación funcionando a través de una web. Sin embargo esta forzaba toda la interacción con la aplicación a ser realizada por medio de un terminal, ajeno a la aplicación. Pensamos que esto podía suponer una gran barrera para la extensión en el uso de esta herramienta, por lo que era necesario ser capaz de acceder a la aplicación a través de un solo punto, eliminando interacciones con elementos ajenos a la aplicación.

Esto conllevaba adaptar los sistemas de comunicación existentes, sin afectarlos en gran medida, ya que, debido a sus dimensiones, no conocíamos el alcance del efecto que suponía la modificación. Debido a mi mayor familiaridad con sistemas Linux y a que me encargaba de la parte backend de la aplicación, me encargué de diseñar e implementar la estructura mostrada en la figura 4.1

Así me hice cargo de la implementación de los distintos métodos de comunicación desde el Front y de la adaptación de esta estructura para que funcionase con el sistema ya existente.

## Implementación de nuevas funcionalidades

Entre estas nos encontramos

- **Creación de simulaciones configurando agentes:** En la que también se vió envuelto Alberto debido a la interfaz necesaria. Pero por la parte del backend

fue necesario definir los datos esenciales de una simulación y abstraerlos en estructuras que permitieran ejecutar nuevas simulaciones desde la nada.

- **Generación de resúmenes de las simulaciones existentes:** En el backend, además de estructurar la información para que se acoplase a la ya existente para una simulación, fue necesario establecer la información prioritaria que se tomaría en cuenta para la generación de un resumen para una simulación.
- **Interacción con los personajes:** También envolvió los dos extremos del desarrollo. La mayor complicación que supuso en el backend fue el diseño correcto del método de comunicación entre back y front, haciendo uso de recursos del sistema de forma correcta y recogiendo la información necesaria en el front por medio de `ReverieComm.py`.
- **Chat con los personajes:** Respecto a esta funcionalidad me tuve que encargar del desarrollo visto en la sección 3.3.4.

## Gestión de la comunicación entre el frontend y el backend de Django

Como comentó Alberto el contrato entre los dos extremos se encontraba aquí, donde contribuimos ambos.

En especial yo contribuí a exponer la información requerida por el front para que pudiera proceder de la forma necesaria. Así mismo fui el encargado de tratar la información recibida del front y adaptarla al formato requerido por el backend.

# Bibliografía

*La literatura es el arte de descubrir algo  
extraordinario en lo ordinario*

Boris Pasternak

- ADHIKARI, S. ET AL. Nlp based machine learning approaches for text summarization. En *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, páginas 535–538. IEEE, 2020.
- ARROYO MENÉNDEZ, M., HASSAN, S., PAVÓN MESTRAS, J. y SANSORES, C. Simulación de sistemas sociales con agentes software. *Actas del Campus Multidisciplinar en Percepcion e Inteligencia, CMPI*, 2006.
- AXTELL, R., AXELROD, R., EPSTEIN, J. M. y COHEN, M. D. Aligning simulation models: A case study and results. *Computational & mathematical organization theory*, vol. 1, páginas 123–141, 1996.
- BAHDANAU, D., CHO, K. y BENGIO, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- BANKES, S. C. Agent-based modeling: A revolution? *Proceedings of the National Academy of Sciences*, vol. 99(suppl\_3), páginas 7199–7200, 2002.
- BENGIO, Y., DUCHARME, R. y VINCENT, P. A neural probabilistic language model. *Advances in neural information processing systems*, vol. 13, 2000.
- BENVENUTI, M., CANGELOSI, A., WEINBERGER, A., MAZZONI, E., BENASSI, M., BARBARESI, M. y ORSONI, M. Artificial intelligence and human behavioral development: A perspective on new skills and competences acquisition for the educational context. *Computers in Human Behavior*, vol. 148, página 107903, 2023.
- BOOKSTABER, R. *The end of theory: Financial crises, the failure of economics, and the sweep of human interaction*. Princeton University Press, 2017.

- BOROOMAND, A. y SMALDINO, P. E. Hard work, risk-taking, and diversity in a model of collective problem solving. *Journal of Artificial Societies and Social Simulation*, vol. 24(4), 2021.
- BOROOMAND, A. y SMALDINO, P. E. Superiority bias and communication noise can enhance collective problem solving. *Journal of Artificial Societies and Social Simulation*, vol. 26(3), 2023.
- BOYD, D. M. y ELLISON, N. B. Social Network Sites: Definition, History, and Scholarship. *Journal of Computer-Mediated Communication*, vol. 13(1), páginas 210–230, 2007. ISSN 1083-6101.
- BROWN, P. F., COCKE, J., DELLA PIETRA, S. A., DELLA PIETRA, V. J., JELINEK, F., LAFFERTY, J., MERCER, R. L. y ROOSSIN, P. S. A statistical approach to machine translation. *Computational linguistics*, vol. 16(2), páginas 79–85, 1990.
- CARD, S. K., MORAN, T. P. y NEWELL, A. The psychology of human-computer interaction. En *The psychology of human-computer interaction*. 1983.
- CASTELFRANCHI, C. Modelling social action for ai agents. *Artificial Intelligence*, vol. 103(1), páginas 157–182, 1998. ISSN 0004-3702. Artificial Intelligence 40 years later.
- CHOMSKY, N. *Syntactic Structures*. De Gruyter Mouton, Berlin, Boston, 1957. ISBN 9783112316009.
- CROWDER, R. M., ROBINSON, M. A., HUGHES, H. P. y SIM, Y.-W. The development of an agent-based modeling framework for simulating engineering team work. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 42(6), páginas 1425–1439, 2012.
- DEVLIN, J., CHANG, M.-W., LEE, K. y TOUTANOVA, K. Bert: Pre-training of deep bidirectional transformers for language understanding. 2019.
- ELDAN, R. y LI, Y. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023.
- EPSTEIN, J. M. y AXTELL, R. *Growing artificial societies: social science from the bottom up*. Brookings Institution Press, 1996.
- FARMER, J. D. y FOLEY, D. The economy needs agent-based modelling. *Nature*, vol. 460(7256), páginas 685–686, 2009.
- GAMBO, I., ADJICHEBOUTOU, A., IKONO, R., IROJU, O. y YANGE, S. An investigative process model for predicting information difusion on social media: Information system perspective. *Ife Journal of Technology*, vol. 27(1), páginas 47–59, 2020.
- GARDNER, M. The fantastic combinations of jhon conway’s new solitaire game’life. *Sc. Am.*, vol. 223, páginas 20–123, 1970.

- GOLDBERG, E., DRIEDGER, N. y KITTREDGE, R. I. Using natural-language processing to produce weather forecasts. *IEEE Expert*, vol. 9(2), páginas 45–53, 1994.
- GOODFELLOW, I., BENGIO, Y. y COURVILLE, A. *Deep learning*. MIT press, 2016.
- GOODFELLOW, I., POUGET-ABADIE, J., MIRZA, M., XU, B., WARDE-FARLEY, D., OZAIR, S., COURVILLE, A. y BENGIO, Y. Generative adversarial nets. *Advances in neural information processing systems*, vol. 27, 2014.
- HANNUN, A., CASE, C., CASPER, J., CATANZARO, B., DIAMOS, G., ELSSEN, E., PRENGER, R., SATHEESH, S., SENGUPTA, S., COATES, A. ET AL. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.
- HECKBERT, S., BAYNES, T. y REESON, A. Agent-based modeling in ecological economics. *Annals of the New York Academy of Sciences*, vol. 1185(1), páginas 39–53, 2010.
- HERNÁNDEZ, M. B. y GÓMEZ, J. M. Aplicaciones de procesamiento de lenguaje natural. *Revista Politécnica*, vol. 32, 2013.
- HOCHREITER, S. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, vol. 91(1), página 31, 1991.
- HOCHREITER, S. y SCHMIDHUBER, J. Long short-term memory. *Neural computation*, vol. 9(8), páginas 1735–1780, 1997.
- HOGEWEG, P. y HESPER, B. The ontogeny of the interaction structure in bumble bee colonies: a mirror model. *Behavioral Ecology and Sociobiology*, vol. 12, páginas 271–283, 1983.
- HOPFIELD, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, vol. 79(8), páginas 2554–2558, 1982.
- HOUSE, J., LANDIS, K. y UMBERSON, D. Social relationships and health. *Science*, vol. 241(4865), páginas 540–545, 1988.
- HUGHES, H. P., CLEGG, C. W., ROBINSON, M. A. y CROWDER, R. M. Agent-based modelling and simulation: The potential contribution to organizational psychology. *Journal of Occupational and Organizational Psychology*, vol. 85(3), páginas 487–502, 2012.
- JELINEK, F. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, vol. 64(4), páginas 532–556, 1976.
- JENNINGS, N. y WOOLDRIDGE, M. J. *Agent technology: foundations, applications, and markets*. Springer Science & Business Media, 1998.
- KARPATHY, A. y FEI-FEI, L. Deep visual-semantic alignments for generating image descriptions. En *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 3128–3137. 2015.

- KAVCIC, A. Software accessibility: Recommendations and guidelines. En *EUROCON 2005-The International Conference on Computer as a Tool*, vol. 2, páginas 1024–1027. IEEE, 2005.
- KINGMA, D. P. y WELLING, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- KOHLER, T. A. y GUMERMAN, G. G. *Dynamics in human and primate societies: Agent-based modeling of social and spatial processes*. Oxford University Press, 2000.
- LECUN, Y., DENKER, J. y SOLLA, S. Optimal brain damage. *Advances in neural information processing systems*, vol. 2, 1989.
- MA, S., WANG, H., MA, L., WANG, L., WANG, W., HUANG, S., DONG, L., WANG, R., XUE, J. y WEI, F. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, 2024.
- MACKENZIE, I. S. Human-computer interaction: An empirical research perspective. *Morgan Kaufmann*, 2012.
- MARKOV, A. Attempt of statistical research on the text of the novel. *Eugene Onegin* illustrating the connection of the tests in chain (Éxample of a statistical investigation of the text of “Eugene Onegin” illustrating the dependence between samples in chain’). In: *Izvestia Imperatorskoi Akademii Nauk (Bulletin of the Imperial Academy of Sciences of St. Petersburg)*. 6th ser, vol. 7, páginas 153–162, 1913.
- MARKOV, A. A. Rasprostranenie zakona bol’shih chisel na velichiny, zavisyashchie drug ot druga (extending the law of large numbers for variables that are dependent of each other). *Izvestiya Fiziko-matematicheskogo obshchestva pri Kazanskom universitete*, vol. 15(2-ya seriya), páginas 124–156, 1906.
- MARKS, R. E. Analysis and synthesis: multi-agent systems in the social sciences. *The Knowledge Engineering Review*, vol. 27(2), página 123–136, 2012.
- MCGINNIS, J., RUSSO, P. y KNICKMAN, J. The case for more active policy attention to health promotion. *Health affairs (Project Hope)*, vol. 21, páginas 78–93, 2002.
- MINSKY, M. *The emotion machine: Commonsense thinking, artificial intelligence, and the future of the human mind*. Simon and Schuster, 2007.
- MISHRA, M., NAYAK, J., NAIK, B. y ABRAHAM, A. Deep learning in electrical utility industry: A comprehensive review of a decade of research. *Engineering Applications of Artificial Intelligence*, vol. 96, página 104000, 2020.
- MOCKUS, A., FIELDING, R. T. y HERBSLEB, J. D. Two case studies of open source software development: Apache and mozilla. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 11(3), páginas 309–346, 2002.

- NASRINPOUR, H. R., FRIESEN, M. R. ET AL. An agent-based model of message propagation in the facebook electronic social network. *arXiv preprint arXiv:1611.07454*, 2016.
- NEUMANN, J. The theory of self-reproducing automata. *Univ. of Illinois Press google schola*, vol. 2, páginas 429–432, 1966.
- NIELSEN, J. *Usability engineering*. Morgan Kaufmann, 1994.
- NORMAN DONALD, A. *The design of everyday things*. MIT Press, 2013.
- PAN, X., HAN, C. S., DAUBER, K. y LAW, K. H. A multi-agent based framework for the simulation of human and social behaviors during emergency evacuations. *Ai & Society*, vol. 22, páginas 113–132, 2007.
- PARK, J. S., O'BRIEN, J. C., CAI, C. J., MORRIS, M. R., LIANG, P. y BERNSTEIN, M. S. Generative agents: Interactive simulacra of human behavior. 2023.
- PICARD, R. W. Computer learning of subjectivity. *ACM Computing Surveys (CSUR)*, vol. 27(4), páginas 621–623, 1995.
- PRATT, A. y NUNES, J. *Interactive design: An introduction to the theory and application of user-centered design*. Rockport Pub, 2012.
- QIAN, C., CONG, X., LIU, W., YANG, C., CHEN, W., SU, Y., DANG, Y., LI, J., XU, J., LI, D., LIU, Z. y SUN, M. Communicative agents for software development. 2023.
- RADFORD, A., NARASIMHAN, K., SALIMANS, T., SUTSKEVER, I. ET AL. Improving language understanding by generative pre-training. *OpenAI*, 2018.
- RAND, W. y RUST, R. T. Agent-based modeling in marketing: Guidelines for rigor. *International Journal of research in Marketing*, vol. 28(3), páginas 181–193, 2011.
- REITER, E. y DALE, R. Building applied natural language generation systems. *Natural Language Engineering*, vol. 3(1), páginas 57–87, 1997.
- ROUFF, C. Autonomy in future space missions. En *AAAI Workshop on Autonomy, Delegation, and Control*. 2002.
- RUSSELL, S. J. y NORVIG, P. *Artificial intelligence: a modern approach*. Pearson, 2016.
- SCHELLING, T. C. Dynamic models of segregation. *Journal of mathematical sociology*, vol. 1(2), páginas 143–186, 1971.
- SOOD, M., GERA, J. y KAUR, H. Creation, evaluation, and optimization of a domain-based dictionary. *Journal of Intelligent & Fuzzy Systems*, vol. 43(5), páginas 6123–6136, 2022.

- SUTSKEVER, I., VINYALS, O. y LE, Q. V. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, vol. 27, 2014.
- TESFATSION, L. y JUDD, K. L. *Handbook of computational economics: agent-based computational economics*. Elsevier, 2006.
- TOMA, P. P., KOZLIK, L. A. y PERWIN, D. G. Systran machine translation system. En *SYSTRAN Machine Translation System..* 1970.
- TOUVRON, H., MARTIN, L., STONE, K., ALBERT, P., ALMAHAIRI, A., BABAEI, Y., BASHLYKOV, N., BATRA, S., BHARGAVA, P., BHOSALE, S. ET AL. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- TROITZSCH, K. G. Perspectives and challenges of agent-based simulation as a tool for economics and other social sciences. *AAMAS (1)*, páginas 35–42, 2009.
- TURING, A. M. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, vol. LIX(236), páginas 433–460, 1950. ISSN 0026-4423.
- UMBERSON, D. y MONTEZ, J. K. Social relationships and health: A flashpoint for health policy. *Journal of Health and Social Behavior*, vol. 51(1\_suppl), páginas S54–S66, 2010. PMID: 20943583.
- VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, Ł. y POLOSUKHIN, I. Attention is all you need. *Advances in neural information processing systems*, vol. 30, 2017.
- WALEN, H. R. y LACHMAN, M. E. Social support and strain from partner, family, and friends: Costs and benefits for men and women in adulthood. *Journal of Social and Personal Relationships*, vol. 17(1), páginas 5–30, 2000.
- WEISS, G. *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT press, 1999.
- WINOGRAD, T. Procedures as a representation for data in a computer program for understanding natural language. *MIT*, 1971.
- WOOLDRIDGE, M. *An introduction to multiagent systems*. John wiley & sons, 2009.

## Templates evaluados

A continuación mostramos los templates usados durante la evaluación de Modelos de Lenguaje de la sección 3.2.3.

Mostramos en primer lugar el contenido de cada uno de los templates empleados. Todos ellos han sido construidos en la aplicación para el envío a la API de GPT-3.5.

Tras ello mostraremos las respuestas que ha dado cada uno de los modelos evaluados, además de las respuestas obtenidas por el modelo GPT-3.5, que usaremos como referencia.

### A.1. Input

#### A.1.1. `action_location_object`

Jane Anderson is in kitchen in Jane Anderson's house.

Jane Anderson is going to Jane Anderson's house that has the following areas: {kitchen, bedroom, bathroom}

Stay in the current area if the activity can be done there. Never go into other people's rooms unless necessary.

For cooking, Jane Anderson should go to the following area in Jane Anderson's house:

Answer: {kitchen}

—

Tom Watson is in common room in Tom Watson's apartment.

Tom Watson is going to Hobbs Cafe that has the following areas: {cafe}

Stay in the current area if the activity can be done there. Never go into other people's rooms unless necessary.

For getting coffee, Tom Watson should go to the following area in Hobbs Cafe:

Answer: {cafe}

—

Isabella Rodriguez is going to Isabella Rodriguez's apartment that has the following areas: {main room, bathroom}

\* Stay in the current area if the activity can be done there.

\* NEVER go into other people's rooms unless necessary.

Isabella Rodriguez is preparing the upcoming Valentine's Day party. For ordering food for the party, Isabella Rodriguez should go to the following area in Isabella Rodriguez's apartment (MUST pick one of {main room, bathroom}):

Answer: {

## A.1.2. action\_object

Current activity: sleep in bed

Objects available: {bed, easel, closet, painting}

Pick ONE most relevant object from the objects available: bed

—

Current activity: painting

Objects available: {easel, closet, sink, microwave}

Pick ONE most relevant object from the objects available: easel

—

Current activity: cooking

Objects available: {stove, sink, fridge, counter}

Pick ONE most relevant object from the objects available: stove

—

Current activity: watch TV

Objects available: {couch, TV, remote, coffee table}

Pick ONE most relevant object from the objects available: TV

—

Current activity: study

Objects available: {desk, computer, chair, bookshelf}

Pick ONE most relevant object from the objects available: desk

—

Current activity: talk on the phone

Objects available: {phone, charger, bed, nightstand}

Pick ONE most relevant object from the objects available: phone

—

Current activity: ordering food for the party

Objects available: {bed, desk, refrigerator, closet, shelf}

Pick ONE most relevant object from the objects available:

### A.1.3. generate\_event\_triple

Task: Turn the input into (subject, predicate, object).

Input: Sam Johnson is eating breakfast.

Output: (Dolores Murphy, eat, breakfast)

—

Input: Joon Park is brewing coffee.

Output: (Joon Park, brew, coffee)

—

Input: Jane Cook is sleeping.

Output: (Jane Cook, is, sleep)

—

Input: Michael Bernstein is writing email on a computer.

Output: (Michael Bernstein, write, email)

—

Input: Percy Liang is teaching students in a classroom.

Output: (Percy Liang, teach, students)

—

Input: Merrie Morris is running on a treadmill.

Output: (Merrie Morris, run, treadmill)

—

Input: refrigerator is being filled with party food.

Output: (refrigerator,

### A.1.4. insight\_and\_evidence

Input:

0. Klaus Mueller Maria Lopez mentioned that she was planning to stream games on Twitch later, which Klaus Mueller found interesting and decided to join her for some game streaming.
1. Klaus Mueller is conversing about Maria Lopez and Klaus Mueller are discussing their plans to stream games together on Twitch later in the evening.
2. Klaus Mueller is waiting to start writing the introduction
3. Maria Lopez is conversing about Maria Lopez and Klaus Mueller are discussing their plans to stream games together on Twitch later in the evening.
4. conversing about Maria Lopez and Klaus Mueller are discussing their plans to stream games together on Twitch later in the evening.
5. For Klaus Mueller's planning: needs to remember to message Maria when he's ready to start streaming games on Twitch later tonight.
6. common room table is being used by Klaus Mueller for research paper writing in the library
7. reviewing her notes
8. Klaus Mueller is actively researching a topic
9. researching new topics related to the class
10. preparing for the next lecture
11. writing the introduction
12. writing the introduction
13. leaving for college
14. Klaus Mueller Maria Lopez is heading off to college.
15. researching the topic
16. Klaus Mueller is socially active
17. Klaus Mueller is organized and prepared
18. Klaus Mueller is academically inclined
19. Klaus Mueller is organized
20. Klaus Mueller and Maria Lopez have a close relationship
21. Klaus Mueller has a close relationship with Maria Lopez
22. Klaus Mueller has a crush on Maria Lopez.
23. closing the cafe
24. Klaus Mueller frequents the Hobbs Cafe
25. Klaus Mueller and Maria Lopez have a strong bond
26. closing the cafe
27. Klaus Mueller has a close relationship with Maria Lopez
28. Klaus Mueller is a sociable person
29. Klaus Mueller is socially active

What 5 high-level insights can you infer from the above statements? (example format: insight (because of 1, 5, 3))

- 1.

## A.2. Output

A continuación tenemos la respuesta de cada modelo a cada uno de los Templates que acabamos de mostrar. Comenzaremos mostrando las respuestas de GPT-3.5, para tenerlas como referencia, tras ello se muestran las respuestas de los modelos Llama 2 7B-32, Llama 2 7B-Q4, Llama 2 13B-Q4 y Llama 2 13B-Q6.

### A.2.1. GPT-3.5

action\_location\_object

main room}

generate\_event\_triple

fill, party food)

action\_object

refrigerator

insight\_and\_evidence

Klaus Mueller and Maria Lopez have a close relationship (20, 25)

2. Klaus Mueller is academically inclined and organized (18, 19)

3. Klaus Mueller is socially active and sociable (16, 29, 28)

4. Klaus Mueller and Maria Lopez enjoy streaming games on Twitch together (0, 1)

5. Klaus Mueller frequents the Hobbs Cafe, indicating a routine or habit (24)

## A.2.2. 7B-32

action\_location\_object

main room}&lt;/s&gt;[end of text]

generate\_event\_triple

is, being, filled)

—

Input: The cat is purring.

Output: (cat, is, purr)

action\_object

refrigerator

—

Current activity: cleaning the house

insight\_and\_evidence

Klaus Mueller is planning to stream games on Twitch later (because of 0)

2. Klaus Mueller and Maria Lopez will be streaming games together on Twitch (because of 1, 5)

3. Klaus Mueller and Maria Lopez have a close relationship (because of 1, 5, 14, 17, 20)

4. Klaus Mueller is socially active (because of 16)

5. Klaus Mueller is academically inclined (because of 18)&lt;/s&gt;[end of text]

### A.2.3. 7B-Q4

action\_location\_object

main room}

—

generate\_event\_triple

is, being, filled)

—

Input: John Lee is giving a presentation.

Output: (John Lee, give, presentation)

action\_object

refrigerator</s>[end of text]

insight\_and\_evidence

Klaus Mueller is planning to stream games with Maria Lopez on Twitch (because of 0, 2, 4).

2. Klaus Mueller is interested in Maria Lopez (because of 1, 5, 20).

3. Klaus Mueller is organized and prepared (because of 15, 18, 20).

4. Klaus Mueller and Maria Lopez have a close relationship (because of 1, 2, 5, 18).

5. Klaus Mueller has a crush on Maria Lopez (because of 2, 15).</s>[end of text]

### A.2.4. 13B-Q4

action\_location\_object

main room}</s>[end of text]

generate\_event\_triple

fill, party food)

—

Input: Jing Liu is painting a picture.

Output: (Jing Liu, paint,

action\_object

refrigerator

—

Current activity: decorate the party ven

insight\_and\_evidence

Klaus Mueller is planning to stream games on Twitch later in the evening (because of 1, 5, 3)

2. Maria Lopez and Klaus Mueller have a close relationship (because of 21, 20, 17)

3. Klaus Mueller is organized (because of 18, 13, 5)

4. Klaus Mueller has a crush on Maria Lopez (because of 22, 3)

5. Klaus Mueller and Maria Lopez have a strong bond (because of 20, 17)

What 5 specific insights can you infer from the above statements? (example format:

### A.2.5. 13B-Q6

action\_location\_object

main room}</s>[end of text]

generate\_event\_triple

fill, party food)

—

Input: the dog is barking loudly.

Output: (dog, bark, loud

action\_object

refrigerator

—

Please note that the objects available  
in

insight\_and\_evidence

Klaus Mueller and Maria Lopez have a  
close relationship (because of 21, 25)

2. Klaus Mueller is socially active (be-  
cause of 16, 17)

3. Klaus Mueller is organized (because of  
18, 19)

4. Klaus Mueller is academically inclined  
(because of 13, 15)

5. Klaus Mueller has a crush on Maria  
Lopez (because of 22, 24)</s>[end of  
text]

# Apéndice **B**

## Template y Prompt

Aquí mostramos un ejemplo de Template y Prompt asociado al mismo, tras el relleno con la información de la simulación

### Template

```
generate_summary.txt
```

```
!<INPUT 0>!: curr_time
```

```
!<INPUT 1>!: maze_name
```

```
!<INPUT 2>!: events
```

```
!<INPUT 3>!: thoughts
```

```
<commentblockmarker>###</commentblockmarker>
```

```
We need to summarize the most relevant information about a social simulation
```

```
This is the information related to the simulation
```

```
Current Time: !<INPUT 0>!
```

```
Map name: !<INPUT 1>!
```

```
Most relevant Events (they are not ordered by relevance):
```

```
!<INPUT 2>!
```

```
Most relevant Thoughts (they are not ordered by relevance):
```

```
!<INPUT 3>!
```

```
Generate a summary of the current state the simulation with the information given above:
```

## Prompt

We need to summarize the most relevant information about a social simulation  
This is the information related to the simulation

Current Time: February 13, 2023, 18:10:00

Map name: the\_ville

Most relevant Events (they are not ordered by relevance):

Isabella Rodriguez witnessed Isabella Rodriguez is conversing about organizing a Valentine's Day party at Hobbs Cafe and inviting Klaus Mueller to join, with Klaus expressing excitement and gratitude for the invitation, discussing activities and a special cake, and planning to bring friends along  
Maria Lopez witnessed Maria Lopez is conversing about Maria and Klaus are discussing various topics including gentrification, quantum computing, cryptography, artificial intelligence, renewable energy technologies, potential research collaborations, gene editing in medicine, and environmental policies over lunch at the library tomorrow.

Klaus Mueller witnessed Klaus Mueller is conversing about Maria and Klaus are discussing various topics including gentrification, quantum computing, cryptography, artificial intelligence, renewable energy technologies, potential research collaborations, gene editing in medicine, and environmental policies over lunch at the library tomorrow.

Most relevant Thoughts (they are not ordered by relevance):

Isabella Rodriguez thought Isabella Rodriguez is heavily involved in planning a Valentine's Day party at Hobbs Cafe

Isabella Rodriguez thought Isabella Rodriguez is heavily involved in the Valentine's Day party at Hobbs Cafe

Isabella Rodriguez thought Isabella Rodriguez is very invested in the Valentine's Day party at Hobbs Cafe

Maria Lopez thought Maria Lopez is an ambitious student

Maria Lopez thought Maria Lopez is preparing for college

Maria Lopez thought Maria Lopez Maria Lopez found Klaus Mueller's research on the effects of gentrification and potential collaborations for future research projects interesting.

Klaus Mueller thought Klaus Mueller is actively researching a topic

Klaus Mueller thought Klaus Mueller Maria's interest in discussing the effects of gentrification in low-income communities and potential collaborations for future research projects with Klaus.

Klaus Mueller thought this is blank

Generate a summary of the current state the simulation with the information given above: