UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Simulador de entornos sociales basado en agentes para analizar flujos de información

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Ramon Ruiz Dolz

Tutor: Ana María García Fornes
Director Experimental: Jose Alemany Bordera

Curso 2017-2018

Resumen

En este trabajo de fin de grado se realiza el desarrollo de un simulador basado en agentes con el fin de analizar flujos de información en una red social. Para ello se han analizado y estudiado las herramientas de simulación existentes que más podían aproximarse a las necesidades del trabajo. Sobre las tecnologías escogidas se ha realizado una serie de modificaciones, mejoras y adaptaciones para adecuarlas al ámbito de las redes sociales. También se ha realizado el modelado y la implementación de varios tipos de agentes que determinan el transcurso de la simulación. Finalmente la herramienta de simulación desarrollada ha sido validada mediante dos casos de estudio diferentes.

Palabras clave: Simulador, Entornos sociales, Agentes

Abstract

In this final degree thesis is explained the development of a multi agent system simulator with the objective of analyzing data flows along a social network. First of all a study of the state of the art regarding multi agent simulators has been made. Over the chosen technologies as a starting point, it have been made changes, improvements and adapted to the area of this work. It have been also developed some models of intelligent agents that will define the behavior of the simulation. Finally, the simulator has been validated with two different study cases.

Key words: Simulator, Social environments, Agents

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

1.1 Motivación

En la actualidad, el uso de las redes sociales en nuestro día a día es un hecho totalmente común en la mayor parte de la población, tanto en jóvenes como en adultos. Las redes sociales consisten en espacios virtuales, dónde se construye una estructura social formada por un conjunto de perfiles completa o parcialmente públicos donde los usuarios, individuos o agrupaciones, interaccionan con otros en función del ámbito o tipo de la red social. Las relaciones que establecen sus usuarios pueden ir desde una relación laboral o profesional hasta una relación de amistad o parentesco. Esta información se representa comúnmente mediante teoría de grafos, en la cual los nodos representan los usuarios de la red y las aristas la relación entre ambos.

Las redes sociales han recibido un gran aumento de usuarios en los últimos años. Es por esto que hoy en día existe un gran interés en el análisis e investigación de éstas. Temas como la seguridad de los usuarios menos expertos, los distintos perfiles de actuación o actividad de los usuarios o la medida de la confianza entre distintos usuarios en base a su actividad en la red, preocupan a muchos investigadores.

La necesidad de utilizar un simulador para analizar las redes sociales surge principalmente debido a la gran cantidad de restricciones que ponen las grandes empresas que gestionan las redes más usadas (Facebook, Google+, etc.) a la hora de proporcionar datos. Disponer de esta herramienta también proporciona la posibilidad de trabajar con situaciones ficticias creadas a partir de situaciones reales. Esto permite poder observar y analizar así la evolución de los diferentes usuarios de la red en diferentes situaciones y en qué pueden afectar factores como la personalidad o la confianza.

En este trabajo se ha desarrollado un simulador para recrear situaciones virtualmente en una red social sintética o real. Finalmente se ha validado mediante dos casos de estudio, uno basado en redes con datos sintéticos [2] y el otro basado en una red con datos reales.

Este trabajo está enmarcado dentro del proyecto nacional "Privacidad en Entornos Sociales Educativos durante la Infancia y la Adolescencia" de I+D+I propuesto por el Grupo de Tecnología Informática e Inteligencia Artificial (GTI-IA) del DSIC. Este proyecto, PESEDIA, tiene como objetivo concienciar y asesorar a los potenciales usuarios más jóvenes en el uso de las redes sociales y la gestión de la privacidad.

1.2 Objetivos

El propósito de este trabajo es el desarrollo de una herramienta de simulación basada en agentes capaz de simular interacciones de usuarios y flujos de información propias de las redes sociales con el fin de analizar y validar gran variedad de propuestas. Estas comprenden desde métricas (de confianza, privacidad, etc.) a efectos producidos por cambios en las formas de interacción.

Para ello, el primer objetivo consiste en el estudio y análisis de las herramientas disponibles para comprobar su adecuación al trabajo. Si es posible se seleccionará aquella que más se ajuste a los requisitos del trabajo y se extenderá para cumplir con la totalidad de los requisitos estipulados. Por lo tanto, otro objetivo consistirá en definir estos requisitos e implementar las funcionalidades de la herramienta de simulación para que cumpla con los mismos. Con la herramienta de simulación desarrollada, se procederá al modelado de los agentes que representen usuarios de redes sociales con el fin de aproximar la simulación a la realidad. Para realizar el modelado, se utilizará un sistema de agentes puesto que esta tecnología es la más apropiada para realizar simulaciones de estas características.

Finalmente se llevarán a cabo dos casos de estudio distintos sobre la herramienta de simulación. El primero de ellos con el propósito de validar el propio simulador y el segundo para mostrar un experimento de ejemplo realizado con la herramienta. En ambos casos se realizarán las comparaciones mediante una métrica que evalúa el riesgo de privacidad de los usuarios de una red social teniendo en cuenta la actividad de estos y el flujo de la información.

En conclusión, los principales objetivos de este trabajo de fin de grado son:

- Análisis y estudio de las herramientas existentes en simulación de redes sociales.
- Especificación de los requisitos de la herramienta de simulación desarrollada.
- Diseño y desarrollo de un simulador basado en agentes de entornos sociales.
- Modelado de los agentes que representan a los usuarios de una red social.
- Aplicación de varios casos de estudio para la validación de la herramienta.

1.3 Estructura de la memoria

Este trabajo está estructurado en seis capítulos. En el Capítulo 2 se introducen conceptos fundamentales con los que se ha trabajado y se presenta una selección de las herramientas de simulación más apropiadas para cumplir nuestros objetivos. En el Capítulo 3 se realiza una especificación de los requisitos de la simulación para conocer cuáles han de ser las características y funcionalidades de la herramienta. El Capítulo 4 es el cuerpo del trabajo puesto que en él se presenta el simulador, se habla de las tecnologías y herramientas utilizadas para el proceso de desarrollo, se define su arquitectura y su implementación, y se expone la estructura de los datos de entrada y salida para su posterior análisis. En el Capítulo 5 se plantean dos casos de estudio distintos sobre los cuales se ha hecho uso de la herramienta con el objetivo de validar el funcionamiento de ésta. Finalmente, en el Capítulo 6, se exponen las conclusiones alcanzadas así como ampliaciones o mejoras aplicables al trabajo realizado.

CAPÍTULO 2
Estado del arte

Este capítulo esta separado en dos bloques principales, en el primero se han definido todos los conceptos fundamentales necesarios para la total comprensión de este trabajo. El segundo bloque consiste en una revisión de las herramientas de simulación existentes hoy en día con mayor adecuación a los objetivos del proyecto.

2.1 Conceptos fundamentales

A lo largo de esta memoria se emplean una serie de conceptos propios de las herramientas de simulación y del dominio de aplicación que nos atañe, las redes sociales. Para facilitar la comprensión del trabajo realizado, a continuación se han definido los conceptos fundamentales más importantes relacionados con el desarrollo de esta herramienta y con las técnicas utilizadas.

2.1.1. Agente

Un agente [14] es un sistema informático capaz de actuar de forma autónoma en un entorno determinado con el fin de alcanzar aquellos objetivos que le han sido delegados. Para ello, primero percibe el entorno y a partir de estas percepciones determina y ejecuta las acciones a realizar de forma totalmente independiente que le lleven a alcanzar sus objetivos y que pueden modificar el entorno.

Sus principales características son la reactividad, la proactividad y la sociabilidad. La primera de estas características implica una interacción constante con el entorno en forma de respuestas a los cambios que ocurren en él. La proactividad consiste en la capacidad de generar e intentar conseguir objetivos sin ir dirigido únicamente por eventos, es decir tomando la iniciativa. Finalmente, la sociabilidad es la capacidad de interactuar con otros agentes mediante cooperación, coordinación y negociación.

En este trabajo se ha utilizado esta tecnología para el modelado de los usuarios de una red social. Por lo tanto se han usado indistintamente los términos agente y usuario como un mismo concepto.

2.1.2. Sistema multiagente

Un sistema multiagente está constituido por un conjunto de agentes que interactúan los unos con los otros, habitualmente intercambiando mensajes a través de una infraestructura de red. En general, los agentes actúan en representación de usuarios que tienen

diferentes objetivos y motivaciones. Para ello las capacidades de cooperación, coordinación y negociación son comunes en los agentes que pertenecen a estos sistemas con el propósito de conseguir alcanzar sus objetivos.

Es habitual el uso de estos sistemas en gran cantidad de herramientas de simulación. En nuestro caso, al simular una red social queremos simular los usuarios interaccionando entre ellos. La definición de agente casa perfectamente con las necesidades que pueden surgir a la hora de modelar un usuario puesto que garantiza independencia y autonomía, interacción con el entorno y otros agentes. Es por esto que el uso de un sistema multiagente en una simulación de este tipo es una buena elección.

2.1.3. Redes sociales

En una red social se construye una estructura social formada por perfiles interconectados que se relacionan entre ellos. Para el correcto modelado de esta, se definen a continuación una serie de conceptos que se utilizarán a lo largo del trabajo.

El primero de ellos, denominado *activity*, sirve para definir lo activo que es un agente en la red social. Este parámetro es de gran relevancia, puesto que en una red social no todos los usuarios son igual de participativos ni interactúan con la misma facilidad frente a una publicación determinada.

Por otra parte, el *tie strength* consiste en el grado de amistad o confianza existente entre dos agentes dentro de la simulación en la red social. Este valor también es de gran importancia, ya que en una red social es muy significativa la relación existente entre dos usuarios a la hora de interactuar en ésta. Cuanto mayor sea el grado de confianza entre dos usuarios mayor será la posibilidad de que estos interactúen entre ellos y viceversa.

Finalmente, el *Privacy Risk Score* (PRS), [2] es la métrica que evalúa el riesgo de privacidad de los usuarios teniendo en cuenta los flujos de información. Esta métrica se ha utilizado para evaluar los casos de estudio llevados a cabo en el Capítulo 5 con la herramienta de simulación desarrollada. Más concretamente, el PRS se define como el indicador del riesgo de amenaza para la privacidad de un agente a_i que realiza una acción de difusión sobre la red social. Conforme mayor sea el valor de esta métrica mayor será el riesgo de amenaza a la privacidad de a_i.

Para el calculo de esta métrica se define T como el número de etapas por las que pasa la publicación. Esta variable representa el número máximo de niveles de profundidad partiendo del agente iniciador por el que una publicación ha fluido desde su creación. También se define la matriz de alcance γ_i, una matriz de $T \times N$ siendo N el número de usuarios. Esta matriz esta asociada a cada usuario para representar el número de mensajes que este usuario ha difundido en una etapa t determinada.

Se toma el valor de la matriz $\gamma_{i_{t,a_j}}$ para referirnos al valor de la matriz γ de un agente i en una etapa t determinada respecto a otro agente j. Este valor se encuentra en la t fila y en la j columna, siendo éste el número de mensajes que el agente i le ha enviado a j en una etapa t.

Se utiliza $L_{a_i}(l)$ para hacer referencia al conjunto de agentes que se encuentran a un nivel de profundidad l tomando el agente i como punto de partida.

Dada una etapa t de la simulación en un nivel de profundidad l a partir del agente a_i se define la Ecuación 2.1 como la media del número de agentes que en este nivel de profundidad ven la publicación creada en la etapa t:

$$p(a_i, t, l) = \frac{\sum_{a_j \epsilon L_{a_i}(l)} \gamma_{i_{t,a_j}}}{\gamma_{i_{t,a_i}}} \tag{2.1}$$

Mediante el valor obtenido en la ecuación anterior, se puede evaluar el PRS de un agente a_i en un nivel de profundidad l como se indica en la Ecuación 2.2, como el porcentaje de agentes de este nivel que muy probablemente vean la publicación realizada por a_i en cualquier etapa:

$$PRS(a_i, l) = \frac{1}{T} \sum_{t=1}^{T} \left(\frac{p(a_i, t, l)}{|L_{a_i}(l)|} \right) \tag{2.2}$$

Combinando estas dos ecuaciones se puede obtener el valor del PRS teniendo en cuenta toda la población de la red social. La Ecuación 2.3 permite obtener este valor para un agente a_i determinado como el porcentaje de agentes que probablemente sean capaces de ver la publicación de este agente en cualquier etapa:

$$PRS(a_i) = \frac{1}{T} \sum_{t=1}^{T} \left(\frac{\sum_{a_j \epsilon N} \gamma_{i_{t,a_j}}}{\gamma_{i_{t,a_i}} |N|} \right) \tag{2.3}$$

El cálculo de esta métrica se ha incorporado en la herramienta de simulación. Esta se obtiene una vez finalizada la simulación. En la Subsección 4.4.5 se explica la implementación del cálculo de esta métrica en el simulador.

2.1.4. Teoría de grafos

Dada la naturaleza del dominio de las redes sociales, gran parte de este trabajo se ha fundamentado en la teoría de grafos. En este apartado se definirá el significado de los elementos de este grafo en la simulación.

Por una parte tenemos la topología de la red, su forma. Una red social esta formada por usuarios interconectados que interactúan entre ellos. Para modelar esto se considera el conjunto de vértices como los propios usuarios y el conjunto de aristas tal que toda arista (u, v) implica que u es amigo de v y viceversa. Por lo tanto, mediante este grafo tenemos gran parte de la información necesaria para realizar la simulación, como son las relaciones de amistad entre usuarios siendo los amigos los nodos vecinos en el grafo.

Por otra parte, el hecho de trabajar sobre un grafo también facilita el posterior análisis de los datos, ya que queremos observar el flujo de una publicación por la red social. La teoría de grafos nos facilita este análisis puesto que el flujo de esta publicación a través de la red social es un subgrafo del grafo que define la topología de la red social. Este subgrafo se define como un conjunto de vértices que conforman los usuarios de la red social y un conjunto de aristas dónde cada arista (u, v) significa que la información ha fluido entre los usuarios u y v.

2.2 Trabajos relacionados

En el dominio de aplicación de este trabajo, las redes sociales, existen gran cantidad de herramientas útiles para el modelado y la ejecución de simulaciones [1]. Estas herramientas nos permiten modelar el conjunto de agentes participantes en la red social de distintas formas. Debido a la naturaleza de nuestra simulación nos interesa tener la capacidad de modelar la red con un grafo dónde los agentes estén desplegados en los nodos

y las aristas representen los vínculos entre ellos. También es interesante el uso de agentes reactivos, puesto que el comportamiento de éstos durante el transcurso de la simulación debe ir guiado por las alteraciones en su entorno y no es necesaria una actividad continuada por parte de cada agente. Finalmente, también interesa poder realizar simulaciones de gran envergadura, ya que realizando una simulación con muchos usuarios, permite apreciar mejor el alcance de una publicación a usuarios desconocidos o inesperados. A continuación se presentan las herramientas más destacadas en esta área.

2.2.1. Agent Factory

Agent Factory [10] es un framework que nos proporciona un conjunto de herramientas, plataformas y lenguajes que permiten el desarrollo y el despliegue de sistemas multiagente. Este framework tiene dos partes, una que permite desplegar los agentes en ordenadores, portátiles y servidores; y otra que permite su despliegue en dispositivos como teléfonos móviles o sensores. Las características principales de este framework son su entorno de ejecución, una plataforma FIPA que permite ajustar los agentes a distintos dominios de aplicación; un lenguaje común, este framework esta dotado de una colección de componentes que facilitan el diseño e implementación de varios lenguajes de programación de agentes; su integración en eclipse, un estándar de interfaz de entorno que facilita vincular los agentes al entorno, y finalmente la interacción basada en ACRE, una herramienta de razonamiento y conversación entre agentes que permite la implementación de protocolos en lenguajes orientados a agentes.

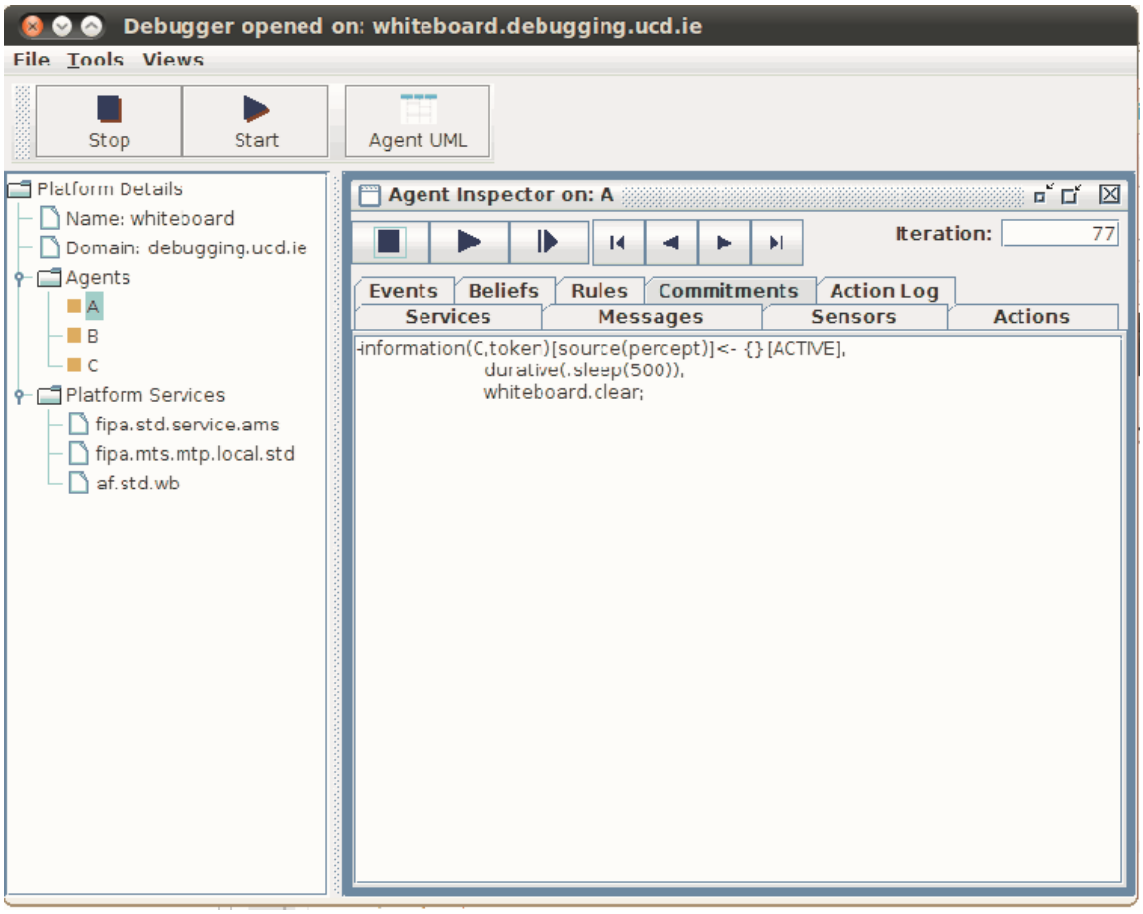

Figura 2.1: Aplicación Agent Factory

La programación con esta herramienta se realiza mediante Java. Por otra parte, el modelado de los agentes mediante Agent Factory no implica ninguna dificultad mas allá de la habitual de estos sistemas. Esta herramienta hace uso de agentes *Belief–Desire–Intention* (BDI). Podemos observar en la Figura 2.1 la interfaz de esta herramienta ejecutándose en Ubuntu, con un panel a la izquierda con los componentes y un panel central donde editar el código.

2.2.2. Agent Sheets

AgentSheets [12] es una herramienta de simulación completamente libre que permite crear juegos y simulaciones de pequeña envergadura y publicarlas en la web. Esto es posible mediante una interfaz fácil de usar gracias a un lenguaje de programación con un sistema *drag-and-drop* para facilitar su uso a los menos expertos en lenguajes de programación basados en reglas. Visual AgenTalk, el lenguaje de programación descrito anteriormente, nos permite modelar agentes reactivos, reproducir sonidos o texto, interactuar con ratón y teclado, mostrar valores como colores o gráficas y realizar consultas a sitios web.

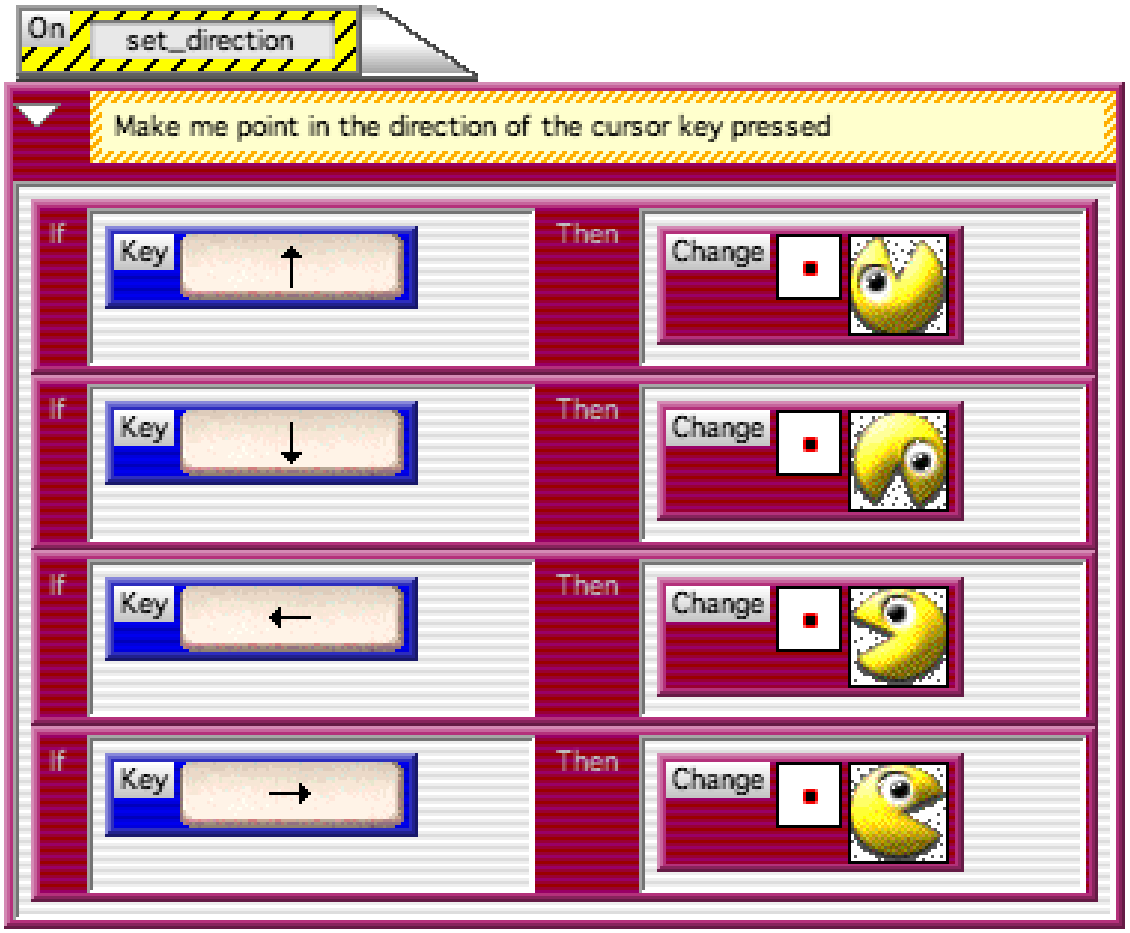

Figura 2.2: Aplicación AgentSheets

En la Figura 2.2 se puede observar la interfaz de la aplicación, concretamente la programación de una función. Se puede apreciar la simpleza del lenguaje de esta herramienta para el uso de los más inexpertos en programación, mediante el uso de símbolos y colores, arrastrando casillas a los espacios vacíos.

2.2.3. Behaviour Composer

Behaviour Composer es una herramienta de modelado que forma parte del proyecto *Modelling4All* de la universidad de Oxford[1] enfocada al aprendizaje del uso y programación de sistemas multiagente. Behaviour Composer proporciona librerías de *"micro-behaviours"* genéricos organizados en categorías para definir el estado inicial de cada agente, su movimiento, apariencia, reproducción, muerte, relaciones o redes sociales. Además también nos permite crear grafos, histogramas, botones y registrar eventos. Los modelos se construyen añadiendo estos *"micro-behaviours"* a los agentes prototipo.

Figura 2.3: Aplicación BehaviourComposer

Estos elementos modulares tienen una interfaz propia que debe ser cuidadosamente combinada con otros. Realmente son fragmentos de código que se ejecutan únicamente si otro fragmento los invoca, sin embargo se lanzan como procesos independientes, hilos o eventos planificados. Este sistema permite facilitar la creación de nuevos comportamientos a usuarios poco habituados a programar. Esta herramienta se encuentra en la propia web del proyecto *Modelling4All* la cual nos permite modelar y lanzar nuestras propias simulaciones. En la Figura 2.3 podemos observar el menú principal de esta herramienta, con los prototipos o agentes que pueden ser modelados mediante el uso de los *"micro-behaviours"* en la pestaña de edición de estos.

2.2.4. AnyLogic

AnyLogic[2] es un software de simulación multiagente enfocado a una gran variedad de dominios, desde dominios relacionados con empresas o negocios (líneas de producción, logística, campañas de marketing, etc.) hasta otros relacionados con la investigación (procesos sociales, relaciones, etc.). Este entorno permite realizar el modelado mediante tres métodos de simulación, los eventos discretos, puesto que los procesos a simular se pueden representar como una secuencia de eventos discretos consecutivos, los sistemas de agentes y la dinámica de sistemas, una forma abstracta de modelar. Esta herramienta también permite animar la simulación en forma de vídeos en 2D y 3D para facilitar la visualización, además de todo tipo de gráficas, histogramas y tablas y cuenta con una serie de librerías específicas para la industria como pueden ser fluidos, raíles o tráfico.

[1] http://m.modelling4all.org
[2] https://www.anylogic.com/

El lenguaje de programación de esta herramienta es Java y los agentes son objetos implementados en este mismo lenguaje. El modelado de los agentes mediante AnyLogic no implica un gran esfuerzo ni dificultad a la hora de su uso y nos permite realizar simulaciones de gran envergadura. AnyLogic no es una herramienta libre, aunque dispone de una versión limitada de prueba para estudiantes. Las versiones completas para empresas o universidades son de pago.

2.2.5. Eve

Eve[3] es una plataforma web totalmente libre basada en agentes con múltiples aplicaciones posibles. Esta serie de herramientas web nos proporcionan un entorno para desarrollar a los agentes. Eve se puede definir como un modelo de agentes y un protocolo de comunicación que se puede implementar en múltiples lenguajes de programación, comúnmente java, y múltiples entornos de ejecución. El modelo de agente en Eve parte de su definición como software que representa entidades existentes, externas o abstractas. Para su funcionamiento adecuado, debe ser capaz de ejecutarse de manera independiente. Para ello es necesario tener una serie de capacidades que Eve proporciona en forma de: independencia temporal o planificación independiente de la entidad representada, memoria para poder almacenar un modelo del estado del mundo en un momento determinado, y comunicación en forma de lenguaje común que permita la comunicación efectiva entre todos los agentes.

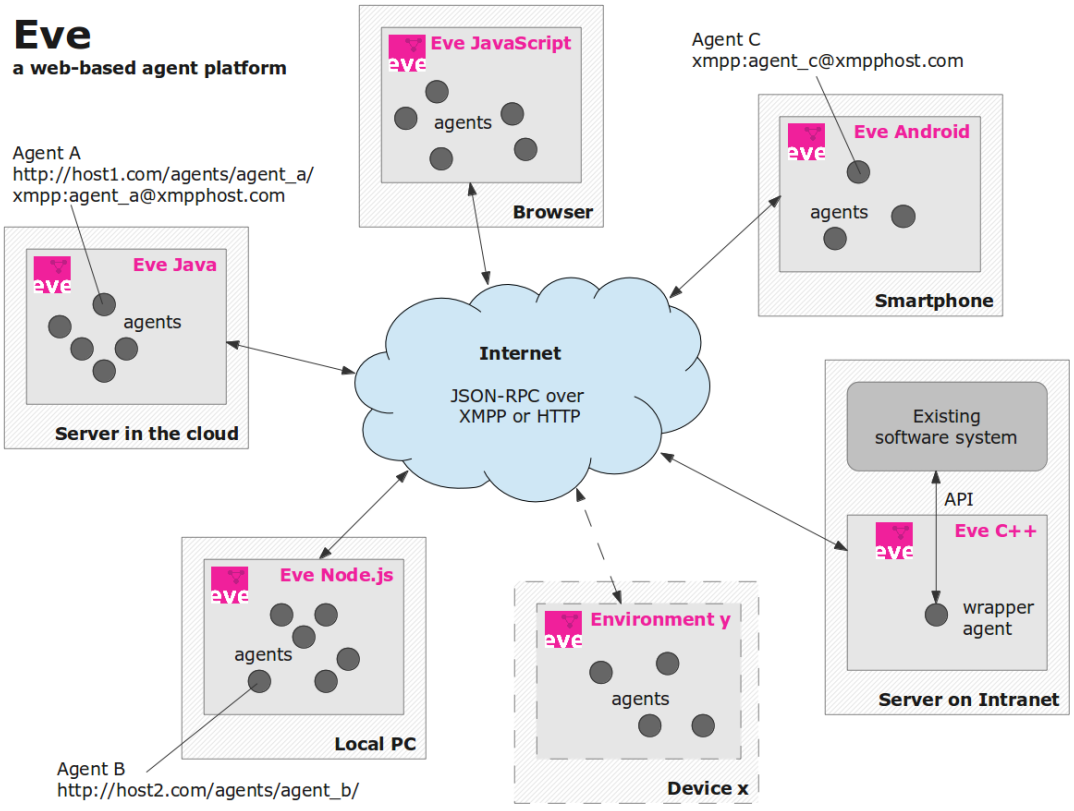

Figura 2.4: Esquema Eve

[3]https://eve.almende.com/

Como podemos observar en la Figura 2.4, mediante esta plataforma los agentes son modelados como clases de Java y las comunicaciones mediante Javascript principalmente. Eve nos permite tener a los agentes desplegados en distintas plataformas como pueden ser la nube, dispositivos móviles, navegadores web o robots y facilita la comunicación entre ellos mediante simples protocolos como *JSON-RPC* sobre capas de transporte como *HTTP* o *XMPP*.

2.2.6. GAMA

GAMA [4] es un entorno de modelado y desarrollo para crear simulaciones espaciales basadas en agentes. Con esta herramienta se pueden modelar y desarrollar simulaciones en gran variedad de dominios de aplicación como por ejemplo simulaciones de transportes, crecimiento de población, epidemología o medioambientales. GAMA proporciona un lenguaje de programación de alto nivel basado en agentes, facilitando el uso a los menos expertos en programación. El lenguaje utilizado es GAML un lenguaje basado en agentes programado en Java. GAMA además facilita la carga de sistemas de información geográficos (GIS) y permite importar grandes cantidades de datos como archivos de texto, CSV o mapas y archivos en 3D. Finalmente, como podemos observar en la Figura 2.5 la interfaz de GAMA proyecta la información de forma muy visual, y permite su personalización para tener interfaces diferentes para el mismo modelo en función de los datos que puedan resultar interesantes.

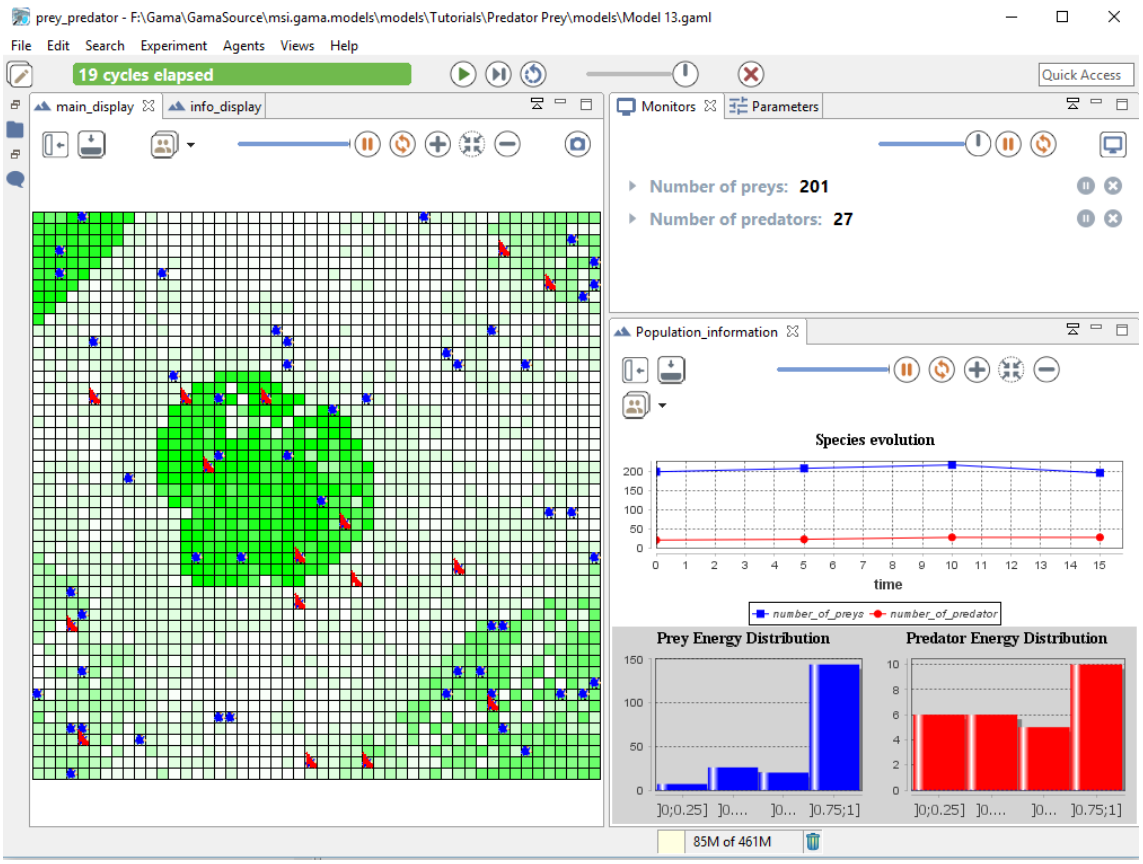

Figura 2.5: Aplicación GAMA

2.2.7. MASON

MASON es una librería de Java que permite modelar simulaciones multiagente basadas en eventos discretos. También contiene una serie de herramientas de visualización 2D y 3D. El principal propósito de esta herramienta es la simulación de agentes conectados en una red o un vecindario. Desarrollada en colaboración por George Mason University's Evolutionary Computation Laboratory y GMU Center for Social Complexity [9] consiste en una opción realmente interesante puesto que es completamente libre. Sin embargo, el modelado de los agentes y de la propia simulación es de gran complejidad. En esta herramienta los agentes son objetos implementados como clases de java, los modelos son totalmente independientes de la visualización de estos y permiten su almacenamiento y recuperación, así como su migración dinámica entre plataformas. En la Figura 2.6 se puede observar la interfaz MASON. Esta herramienta dispone de una visualización gráfica de la simulación.

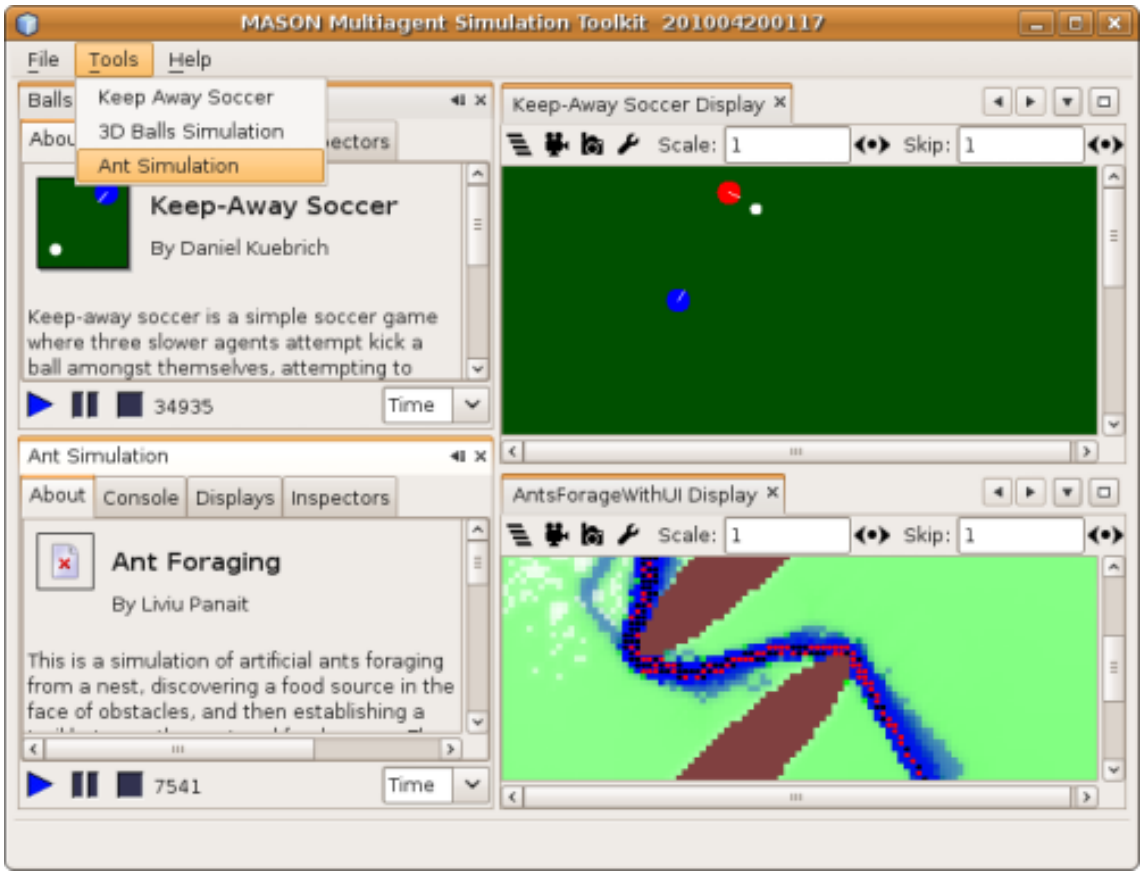

Figura 2.6: Aplicación MASON

2.2.8. NXsim

NXsim[4] es una librería de Python para la simulación de agentes conectados por una red, donde los agentes son objetos implementados como hilos de Python situados en un grafo. Esta librería nos permite realizar simulaciones de gran envergadura. Concretamente, NXsim es la unión de dos conocidas librerías de Pyhton, NetworkX y SimPy.

NetworkX[5] es una librería de Python que permite la creación, manipulación y estudio de la estructura, las dinámicas y las funciones de redes complejas. Mediante esta

[4]https://github.com/kentwait/nxsim
[5]https://networkx.github.io/

herramienta, NXsim crea el grafo base que define la topología de la red de la simulación. Además, NXsim puede desplegar los agentes en los nodos del grafo, por lo tanto NetworkX conforma el pilar estructural de esta herramienta de simulación.

Por otra parte, SimPy[6] es un framework de Python que permite realizar simulaciones basadas en eventos discretos por medio de hilos. NXsim utiliza este framework para crear a los agentes como hilos de Python.

Al ser Pyhton el lenguaje de programación, la dificultad del modelado es relativamente asequible e intuitiva para cualquier usuario que este familiarizado con la programación. Esta librería nos permite definir el comportamiento de los agentes que van interactuando unos con otros en función del transcurso de la simulación y de su propio modelado. Otra de las ventajas de este lenguaje es la gran comunidad que tiene. En Python existen gran cantidad de librerías que pueden combinarse entre sí, como se ha expuesto, permitiendo así ampliar cómodamente las funcionalidades de la herramienta según los requisitos del trabajo.

2.3 Conclusiones

Como hemos podido observar, hay muchas opciones para simular una red social y posteriormente analizar los datos. En la Tabla 2.1 se han recogido las características más relevantes de las herramientas presentadas en el punto anterior.

Herramienta	Código fuente	Tipo de agentes	Lenguaje de programación	Dificultad de Modelado	Escala de la simulación	Libre	Aplicaciones
Agent Factory	Java	Reactivos, BDI, deliberativos	Java	Moderada	Grande	Sí	Múltiple
AgentSheets	Java	Reactivos	Visual AgenTalk	Fácil	Pequeña-media	Sí	Educativo
AnyLogic	Java	Genérico	Java	Moderada	Grande	No	Simulaciones para empresas
Behaviour Composer	Java	Genérico	Ajax/Netlogo	Fácil	Pequeña-media	Sí	Educativo
Eve	Java	Genérico	Eve-API/Java	Moderada	Pequeña-media	Sí	Múltiples
GAMA	YourKit Java	Reactivos	GAML	Moderada	Media-grande	Sí	Múltiples
MASON	Java.net	Genérico	Java	Compleja	Media-grande	Sí	Redes o vecindarios
NXsim	Python	Genérico	Python	Fácil	Media-grande	Sí	Agentes conectados mediante una red

Tabla 2.1: Comparación de herramientas

En este capítulo se han presentado un total de ocho herramientas de simulación de agentes. En nuestro caso es interesante trabajar con una herramienta que tenga una curva de aprendizaje no muy grande, permita lanzar gran cantidad de agentes sin suponer una gran carga para la máquina, y tenga un modelado muy flexible para poder ajustarlo a nuestras necesidades y no tener que trabajar con un esquema ya preestablecido.

Todas estas herramientas tienen un enfoque que permite la implementación y el modelado de una simulación como la que se pretende realizar en este trabajo, sin embargo es importante saber seleccionar la que permita realizar este modelado de la forma más cómoda posible.

Podemos observar que no todas las herramientas proporcionan el mismo tipo de agentes. Agent Factory o GAMA disponen de agentes reactivos, sin embargo en MASON o NXsim son objetos implementados en un lenguaje de programación, aportando esto mayor flexibilidad a la hora de trabajar con estos agentes. En este aspecto pueden

[6] https://simpy.readthedocs.io/en/latest/

resultar interesantes agentes reactivos puesto que deben reaccionar a estímulos externos. Sin embargo, la ventaja de los agentes genéricos sobre estos es que no únicamente se limitan a este tipo y permiten realizar mayor variedad de experimentos combinándolos con otros tipos de agente como por ejemplo, agentes con creencias o estados. Es por esto que las herramientas que más se adecuarían en este aspecto son AnyLogic, Behaviour Composer, Eve, MASON y NXsim

Por otra parte, el lenguaje en el que se programa puede ser muy variado, desde Java o Pyhton hasta lenguajes propios de las herramientas como GAML o Visual AgenTalk. Respecto a esta característica, interesa usar un lenguaje que no sea propio de la herramienta como Python o Java ya que nos permitirán implementar con mayor comodidad y rapidez la simulación puesto que se han aprendido a lo largo del grado. De las herramientas seleccionadas según el tipo de agentes, teniendo en cuenta el lenguaje de programación resultarían interesantes AnyLogic, Eve, MASON y NXsim.

Respecto a la escala de simulación, para realizar los experimentos finales de este trabajo interesa que la herramienta sea capaz de lidiar con 1000 agentes. En este aspecto, es deseable que la herramienta tenga una escala de simulación media-grande como mínimo. Así pues, de las herramientas anteriores únicamente serían interesantes AnyLogic, MASON y NXsim.

Otro aspecto a tener en cuenta es la política del software de cada herramienta. Debido a la naturaleza de este trabajo es necesario que sea una herramienta de libre uso. Por lo tanto, MASON y NXsim compondrían las herramientas más cercanas a los requisitos de este trabajo.

Sin embargo, teniendo en cuenta el dominio de aplicación, aunque ambas herramientas se encuentran en un dominio similar, NXsim se encuentra exactamente en el dominio funcional que se espera de una simulación como la de este trabajo.

Es por esto que la herramienta seleccionada para realizar el modelado y la implementación de la simulación es NXsim. Esta herramienta nos proporciona una plataforma de agentes muy flexible, modificable y eficiente. Mediante el uso de esta herramienta y de otras librerías combinadas se ha conseguido cumplir con el desarrollo de este trabajo y todos sus objetivos.

CAPÍTULO 3
Especificación de requisitos

3.1 Introducción

3.1.1. Propósito

La especificación de requisitos que se ha realizado para el desarrollo de la herramienta de simulación se ha llevado a cabo siguiendo el estándar IEEE830. El principal propósito de este capítulo es recopilar todas aquellas funciones o características necesarias para que la herramienta funcione de la manera esperada. También se definirán una serie de restricciones y dependencias que se deben cumplir para el adecuado funcionamiento del simulador.

3.1.2. Ámbito del sistema

La herramienta desarrollada en este trabajo consiste en una herramienta de simulación multiagente enfocada al flujo de la información en una red social. Esta herramienta será capaz de simular usuarios interactuando con publicaciones que fluyen por la red social simulada. El propósito del desarrollo de esta herramienta es la obtención de datos sintéticos provenientes de la interacción entre usuarios en una red social. Esta información es de gran valor hoy en día puesto que las grandes redes sociales limitan bastante el acceso a sus datos. La meta de este trabajo es el desarrollo y la validación de la herramienta y la experimentación en un caso de estudio real para ilustrar el funcionamiento de esta.

3.1.3. Visión general de la especificación

Para el desarrollo de este trabajo son necesarios toda una serie de requisitos que nos aseguren la correcta interacción entre agentes, el correcto procesado de la información, la medición del riesgo de privacidad, así como la representación gráfica de los resultados. Es por ello que a continuación, en la Sección 3.2, se va a realizar una descripción de todos aquellos factores que afecten a la herramienta y a sus requisitos. Desde la descripción de las funciones del simulador, hasta la descripción de las restricciones de este a la hora de su funcionamiento. Por otra parte, en la Sección 3.3 se ha realizado una detallada descripción de los requisitos a cumplir por parte de la herramienta desarrollada. En este caso se ha enfocado a una especificación de requisitos funcionales.

3.2 Descripción general

3.2.1. Perspectiva del trabajo

La herramienta de simulación a desarrollar es un proyecto totalmente independiente. Esta herramienta no es parte de otro sistema mayor ni desempeña funciones para otro sistema. El simulador está desarrollado en Python3, por lo tanto deberá funcionar en cualquier sistema con Python3 instalado y que cumpla todas las restricciones.

3.2.2. Funciones del simulador

La herramienta desarrollada debe ser capaz de simular el comportamiento de usuarios interconectados en una red social frente al paso de una publicación. Para ello, esta herramienta debe ser capaz de cumplir las siguientes funcionalidades:

- Generación de topologias de red basadas en modelos reconocidos.
- Importación de una topología de red existente para ser utilizada como entrada en el simulador.
- Exportanción de la topología de red utilizada.
- Simulación discreta basada en unidades de tiempo.
- Interacción entre agentes conectados por la topología de red de entrada.
- Interacción entre agentes mediante *shares*, *likes*, comentarios y etiquetados.
- Modelado de los agentes de la simulación.
- Capacidad de lanzar un número determinado de simulaciones.
- Selección de los agentes iniciadores de la simulación.
- Ejecución selectiva y progresiva de los agentes involucrados en la simulación.
- Configuración de los datos de salida del simulador.
- Definición de métricas y medición sobre los valores de la simulación.
- Procesado gráfico de los datos de la simulación.

3.2.3. Características de los usuarios

Esta herramienta esta enfocada a la obtención de datos sintéticos propios de una red social para la realización de experimentos sobre ellos. Por lo tanto, el perfil de usuario al que está enfocada esta herramienta es de investigador, con estudios universitarios y con algo de experiencia en lenguajes de programación.

3.2.4. Restricciones

Esta herramienta tiene una serie de dependencias y restricciones dado su desarrollo en python, el uso de una serie de librerías o el *hardware* de ejecución en sí. Para la adecuada ejecución de esta, será totalmente necesario satisfacer las siguientes dependencias y restricciones:

- Se requiere tener instalado Python en una versión superior a la 3.4.
- Se requieren tener instalados todos los paquetes que la herramienta utiliza, es decir, NXsim, NetworkX, Simpy, Matplotlib y Numpy.
- Los agentes que se pueden simular no son infinitos. Es por esto que para la eficiente ejecución de las simulaciones se debe elegir adecuadamente la máquina dónde lanzar cada simulación.

Considerando todos estos factores que se acaban de describir, a continuación se detallarán los requisitos finales que implica el desarrollo de la herramienta de simulación.

3.3 Requisitos específicos

En esta sección se presentan todos aquellos requisitos que la herramienta de simulación desarrollada debe cumplir. Se han dividido en dos categorías; requisitos funcionales, aquellos que determinan funciones que la simulación debe cumplir por una parte, y requisitos de diseño, aquellos requisitos relacionados con el diseño de la herramienta.

3.3.1. Requisitos funcionales

Identificador	RF01
Nombre	Generación de datos aleatorios siguiendo una distribución específica.
Descripción	Posibilidad de crear datos fictícios aleatorios en la simulación sin necesidad de partir de una topología real.
Entradas	Topología del grafo que define a la red social.
Salidas	Lista de estados inicializados a valores aleatorios.

Tabla 3.1: Requisito funcional RF01

Identificador	RF02
Nombre	Importación/Exportación de las topologías de red.
Descripción	La herramienta debe ser capaz de importar y exportar los datos de las topologías de red utilizadas durante las simulaciones.

Tabla 3.2: Requisito funcional RF02

Identificador	RF03
Nombre	Simulación discreta basada en unidades de tiempo.
Descripción	La herramienta debe realizar simulaciones discretas que avancen por unidades de tiempo.

Tabla 3.3: Requisito funcional RF03

Identificador	RF04
Nombre	Interacción directa entre agentes conectados.
Descripción	Los agentes deben poder interactuar directamente con otros agentes que estén conectados por la topología de la red.

Tabla 3.4: Requisito funcional RF04

Identificador	RF05
Nombre	Acción de compartir.
Descripción	Capacidad de difundir la publicación con los contactos a través de la red social.
Entradas	Identificador del posible agente al que comparte y parámetros de *tie strength* y *activity*.
Salidas	Cambio de estado y nueva iteración en la simulación.

Tabla 3.5: Requisito funcional RF05

Identificador	RF06
Nombre	Acción de otorgar *likes*.
Descripción	Capacidad de reconocer públicamente el interés hacia una determinada publicación.
Entradas	Identificador del agente que ha compartido y parámetros de *tie strength* entre los agentes.
Salidas	Asigna un me gusta en la publicación.

Tabla 3.6: Requisito funcional RF06

Identificador	RF07
Nombre	Acción de comentar.
Descripción	Capacidad de publicar un comentario en una determinada publicación.
Entradas	Identificador del agente que ha compartido y parámetros de *tie strength* entre los agentes.
Salidas	Comentario en la publicación.

Tabla 3.7: Requisito funcional RF07

Identificador	RF08
Nombre	Acción de etiquetar.
Descripción	Capacidad de añadir un contacto a una publicación para que la acción de compartir sea más directa.
Entradas	Identificador del posible agente al que comparte y parámetros de *tie strength* y *activity*.
Salidas	Agente al que se comparte etiquetado en la publicación.

Tabla 3.8: Requisito funcional RF08

Identificador	RF09
Nombre	Capacidad de ignorar.
Descripción	Posibilidad de rechazar una publicación tras ser compartida múltiples veces con el mismo agente.
Entradas	Parámetro *activity* del propio agente.
Salidas	Cambio de estado respecto a la publicación.

Tabla 3.9: Requisito funcional RF09

Identificador	RF10
Nombre	Capacidad de leer.
Descripción	Posibilidad de únicamente leer la publicación sin necesidad de interactuar con ella.
Entradas	Parámetro *activity* del propio agente.
Salidas	Cambio de estado respecto a la publicación.

Tabla 3.10: Requisito funcional RF10

Identificador	RF11
Nombre	Selección del numero de simulaciones.
Descripción	La herramienta debe permitir la selección del número de simulaciones a realizar en una ejecución.

Tabla 3.11: Requisito funcional RF11

Identificador	RF12
Nombre	Selección de los agentes iniciadores.
Descripción	La herramienta debe permitir la selección de cuáles van a ser los agentes que iniciarán la simulación realizando una publicación.

Tabla 3.12: Requisito funcional RF12

Identificador	RF13
Nombre	Almacenamiento de resultados.
Descripción	Capacidad de almacenar la información resultante.
Entradas	Dirección de almacenado y valores a guardar.
Salidas	Fichero externo con los datos almacenados.

Tabla 3.13: Requisito funcional RF13

Identificador	RF14
Nombre	Configuración de los datos de salida.
Descripción	La herramienta debe permitir al usuario configurar la ubicación y la cantidad de datos a almacenar una vez finalizada la simulación.

Tabla 3.14: Requisito funcional RF14

Identificador	RF15
Nombre	Definición de métricas y medición sobre los resultados.
Descripción	La herramienta debe implementar la métrica de PRS y realizar la medición de los resultados obtenidos mediante esta métrica.

Tabla 3.15: Requisito funcional RF15

Identificador	RF16
Nombre	Procesado gráfico de los datos.
Descripción	La herramienta debe ser capaz de realizar un procesado gráfico de los datos después de finalizar la simulación.

Tabla 3.16: Requisito funcional RF16

Identificador	RF17
Nombre	Almacenamiento de imágenes.
Descripción	Capacidad de almacenar las gráficas obtenidas tras el procesamiento de los datos.
Entradas	Dirección de almacenado e imagen a guardar.
Salidas	Fichero tipo .jpg

Tabla 3.17: Requisito funcional RF17

Identificador	RF18
Nombre	Distribución de los agentes.
Descripción	La herramienta debe distribuir a los agentes siguiendo una topología de red determinada. Esta puede ser de una red social real o bien aleatoria.

Tabla 3.18: Requisito funcional RF18

3.3.2. Requisitos de diseño

Identificador	RD01
Nombre	Optimización de espacio.
Descripción	La herramienta almacenará únicamente los datos indispensables para las experimentaciones posteriores, utilizando así la menor cantidad de espacio en disco posible.

Tabla 3.19: Requisito de diseño 01

Identificador	RD02
Nombre	Optimización de los agentes.
Descripción	Los agentes de la herramienta únicamente estarán en funcionamiento si estan realizando alguna acción, en caso negativo deberán mantenerse desactivados. De esta forma la herramienta utilizará la menor cantidad de recursos posible.

Tabla 3.20: Requisito de diseño 02

Identificador	RD03
Nombre	Diseño modular.
Descripción	El diseño de la herramienta estará dividido en distintos módulos independientes que desempeñen distintas funciones.

Tabla 3.21: Requisito de diseño 03

Mediante esta especificación de requisitos quedan definidas las funcionalidades y características del simulador. En el siguiente capítulo se detallarán aspectos como la tecnología, la arquitectura y el modelado de los agentes.

CAPÍTULO 4
Diseño y desarrollo

4.1 Introducción

La herramienta de simulación desarrollada a lo largo de este trabajo de fin de grado tiene como objetivo facilitar el análisis de comportamientos en redes sociales para observar factores que puedan condicionar la naturaleza de estos y, por lo tanto, tengan repercusión en la privacidad de los usuarios.

Mediante el uso de NXsim como base para la implementación y modelado del simulador y de Matplotlib y Numpy para el tratamiento de datos, se han conseguido alcanzar los objetivos expuestos en el Capítulo 1. Esta herramienta, al ser un paquete de Python, facilita su combinación con otros paquetes muy útiles como pueden ser Matplotlib para obtener gráficas que representen la información de una forma más visual, o Numpy para poder manejar números y realizar operaciones con ellos muy cómodamente. Además, NXsim únicamente nos proporciona los agentes modelados como hilos así que existe libertad prácticamente total para el modelado del comportamiento de estos agentes según los objetivos a alcanzar.

NXsim inicialmente despliega todos los agentes en los nodos de un grafo creado por NetworkX y los activa para que realizen las acciones con las que se les haya modelado. La información de la simulación funciona mediante un sistema de estados que se va modificando en cada unidad de tiempo en función de lo que vaya sucediendo en la simulación. Además, permite lanzar varias simulaciones en la misma ejecución ajustando el número de *trials* a realizar. Esto es de gran interés porqué de esta forma se puede realizar gran cantidad de experimentos sin necesidad de estar constantemente comprobando los resultados.

Para definir correctamente la implementación, el modelado de los agentes y las ampliaciones a realizar sobre la librería, se ha analizado el cumplimiento de los requisitos definidos en el capítulo anterior. NXsim cumple con los requisitos funcionales RF03, RF04, RF11, RF12, RF013 y RF18. Sin embargo, no cumple con ningún otro requisito funcional o de diseño. Para ello, a lo largo de este capítulo se mostrará todo el trabajo realizado para que la herramienta de simulación desarrollada cumpla con todos y cada uno de los requisitos del trabajo.

A continuación se realizará un breve repaso sobre las tecnologías utilizadas en el desarrollo del simulador. Todas ellas han sido de gran importancia para la correcta realización de este trabajo, aportando características indispensables y funciones de gran utilidad.

4.2 Tecnologías utilizadas

El desarrollo del simulador ha implicado el uso de una serie de tecnologías y herramientas. A continuación se presenta una breve descripción de estas herramientas y sus ventajas.

- **Python**. El lenguaje de programación Python es un lenguaje interpretado de alto nivel, cuenta con tipado dinámico y es multiplataforma. Este lenguaje hace hincapié en la legibilidad del código mediante el uso de la indentación como parte de la sintaxis del lenguaje. Python soporta tanto la programación orientada a objetos, como la imperativa, funcional y la procedural. Además, cuenta con gran cantidad de librerías enfocadas a gran variedad de tareas como las que se exponen a continuación.

- **NXsim**. El principal propósito de esta tecnología es la simulación de agentes conectados por cualquier tipo de red. Esta librería de Python es la base sobre la cual se ha realizado este trabajo. NXsim proporciona la base para crear cualquier tipo de simulación que consista en una sucesión de eventos en el tiempo ubicados en una red.

- **SimPy**. Es la principal librería de Python que permite la realización de simulaciones basadas en procesos. Esta herramienta nos brinda todo tipo de tecnologías para el correcto desempeño de la simulación.

- **NetworkX**. Es una librería de Python para crear, manipular y estudiar la estructura, dinámica y funciones de redes complejas. Esta librería de Python permite crear grafos y trabajar con ellos de forma muy cómoda. Es útil para la generación de la red social y su topología.

- **NumPy**. Es el principal paquete de Python usado para cálculo científico. Este paquete nos proporciona las tecnologías necesarias para trabajar con grandes matrices de datos y para el procesado de estos. En nuestro simulador, NumPy es usado para manejar los datos obtenidos tras las simulaciones.

- **Matplotlib**. Este paquete de Python aporta la capacidad de representar información en 2D. Matplotlib es realmente útil en el desarrollo de este trabajo ya que permite la obtención visual del flujo de la publicación así como las gráficas con los datos y las estadísticas de la simulación para una posterior comparación entre simulaciones.

- **JSON**. Para poder guardar la información de las simulaciones se ha utilizado JSON. Esta librería de Python proporciona las tecnologías necesarias para el almacenamiento de datos en un archivo externo. Esto es realmente útil para llevar clasificaciones de resultados en función de determinados factores y poder así analizar más cómodamente los resultados. También permite realizar el procesado gráfico de los datos en cualquier momento y no únicamente al finalizar el transcurso de la simulación.

- **Git**. Esta herramienta es un software de control de versiones. Este trabajo ha sido íntegramente desarrollado por una persona, sin embargo el uso de esta tecnología es realmente útil e interesante ya que permite llevar un control de las modificaciones realizadas en el código. Por otra parte, también facilita la detección de errores y brinda la posibilidad de recuperar versiones anteriores.

4.3 Arquitectura del simulador

En esta sección se ha detallado la estructura interna del simulador desarrollado. Para ello se han definido uno a uno los módulos de la herramienta.

El simulador está compuesto por tres bloques funcionales (Figura 4.1). El primero de ellos consiste en la inicialización de la simulación. Dentro de este bloque se ha implementado un módulo de configuración que se combina con los parámetros de entrada para poder dar comienzo a la simulación. El segundo de ellos es la simulación. Este bloque está formado por una serie de agentes Python que se encargan de llevar a cabo todas las tareas que forman la simulación siguiendo un comportamiento definido previamente. Finalmente, el simulador dispone de la función de tratamiento de datos. Esta función está compuesta por dos módulos, un módulo de cálculo del PRS y otro módulo de procesado gráfico de la información.

El módulo de configuración parte de la topología de la red social y se encarga de asignar los atributos a los agentes antes de iniciar la simulación. Mediante este módulo se pueden realizar ajustes en sus atributos para observar su repercusión en el desarrollo de la simulación.

El módulo de cálculo del PRS se encarga de calcular este valor para todos los agentes de la simulación y el módulo de procesado gráfico permite obtener una representación gráfica y visual del transcurso de la simulación. Estos dos módulos conforman la salida de la herramienta y se explican con mayor detalle en la siguiente sección.

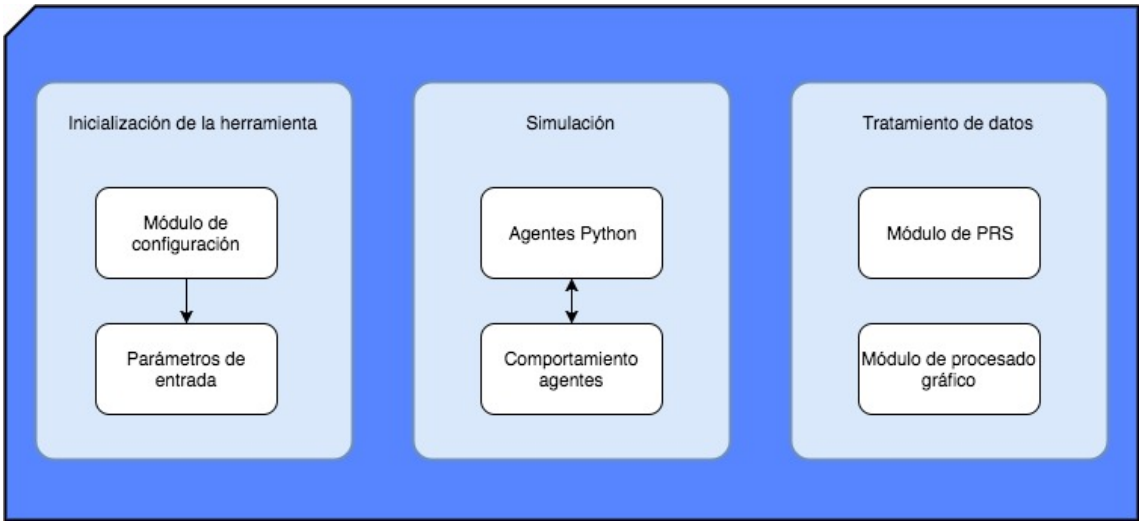

Figura 4.1: Diagrama de la arquitectura de la herramienta de simulación.

Por otra parte, el bloque de la simulación es el que permite obtener los datos sintéticos simulando una red social. Está formado por una serie de agentes explicados en la siguiente sección proporcionados por NXsim y el comportamiento que caracteriza a los usuarios que se ha implementado como parte de este trabajo.

A parte del diagrama de arquitectura, en la Figura 4.2 podemos observar un diagrama de flujo del simulador. Este tiene como entrada la topología de la red social a simular y los atributos de los agentes además de una serie de parámetros de entrada explicados más adelante. Con toda esta información se lanza la simulación, que avanza unidad de tiempo a unidad de tiempo hasta cumplir alguna de las condiciones. Finalmente, se obtienen una serie de ficheros con los datos del estado final de cada simulación realizada. A partir de estos datos los módulos de PRS y de procesado gráfico son capaces de obtener

tanto los valores de PRS de todos los agentes como distintas gráficas con información de la simulación, como por ejemplo el grafo del flujo de la publicación entre agentes o la evolución del flujo por unidad de tiempo.

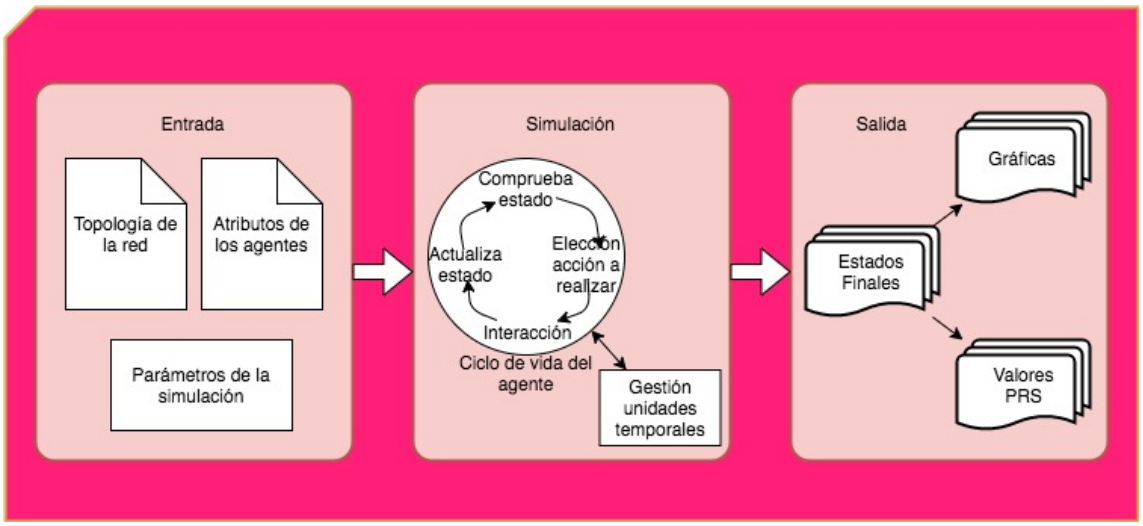

Figura 4.2: Diagrama de flujo de la herramienta de simulación.

Además de la herramienta de simulación también se ha considerado implementar un analizador de información que facilite el paso de información de redes sociales reales a esta herramienta de simulación multiagente. La estructura del analizador, como podemos observar en la Figura 4.3, está formada por un módulo de análisis de la topología y otro módulo de análisis de datos.

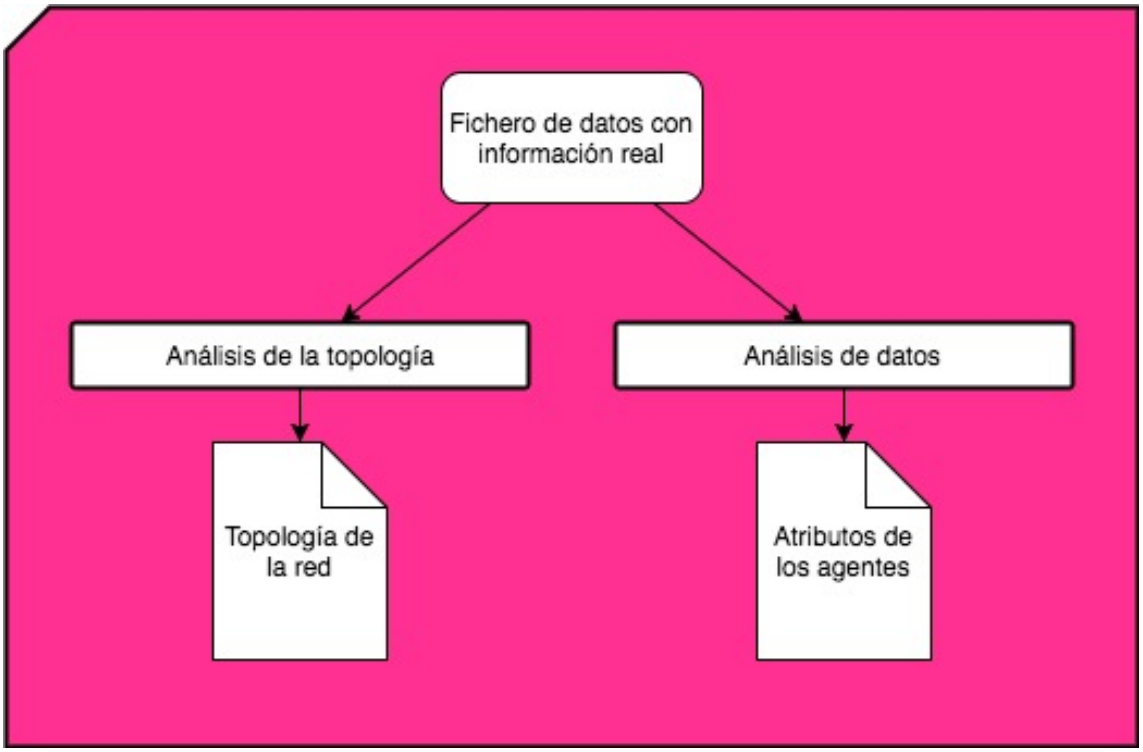

Figura 4.3: Diagrama de flujo del analizador de datos.

El módulo de análisis de la topología permite obtener como salida un fichero con la topología de la red social en cuestión totalmente compatible con la herramienta de simu-

lación. Por otra parte, el módulo de análisis de datos se encarga de obtener parámetros que puedan definir el comportamiento de los usuarios simulados a partir de datos reales de la red social. Este analizador nos proporciona una salida totalmente compatible con el módulo de configuración de la herramienta de simulación, pudiendo así trabajar más fácilmente con datos reales.

4.3.1. Tipos de agentes existentes en NXsim

La librería NXsim hace uso de múltiples tipos de agentes para garantizar el correcto desempeño de sus funcionalidades. Concretamente se compone de cuatro clases de agentes (*BaseAgent*, *BaseNetworkAgent*, *BaseEnvironmentAgent* y *BaseLoggingAgent*). Estas son la clase de agente básica a partir de la cual se extenderán las demás clases; la clase de agente de la red, que serán los que se modelarán para alcanzar los objetivos de este trabajo; la clase de agente de entorno y la clase de agente recopilador.

- **BaseAgent**. En la clase básica *BaseAgent* se definen todas las utilidades o métodos fundamentales para trabajar con los agentes durante la simulación. Estos métodos permiten obtener todos los nodos del grafo, obtener todos los agentes presentes en la simulación o alguno en concreto, obtener una lista de sus amigos, obtener una lista de los nodos vecinos y eliminar un nodo de la simulación.

- **BaseNetworkAgent**. Esta clase únicamente extiende a la anterior y se asegura de que los agentes de este tipo posean un id y un estado, puesto que van a ser los agentes encargados de actuar como usuarios de la red social y estos son parámetros obligatorios.

- **BaseEnvironmentAgent**. Esta clase de agente es la que se encarga de la gestión del entorno de la simulación. Es por ello que, a demás de los métodos disponibles de la clase *BaseAgent*, también tiene la capacidad de añadir nodos y aristas en el grafo que representa a la red social.

- **BaseLoggingAgent**. Este tipo de agente es el que se encarga de llevar el registro y recopilar la información de la simulación. Para ello dispone de una serie de métodos que le permiten almacenar en un fichero la información del historial de estados de la simulación, es decir, un registro con el paso del flujo de la información a través de la red social y las acciones tomadas por los agentes de la red en cada unidad de tiempo. Este agente es también el encargado de trabajar con estos datos. Es por ello que también dispone de un método que le permite abrir este fichero permitiendo así trabajar con los datos una vez finalizada la simulación.

Estos cuatro tipos de agentes componen la espina dorsal de nuestra herramienta de simulación. En la Sección 4.4 se explica la implementación que se ha realizado para dar vida a estos agentes y que actúen de forma coherente con nuestros objetivos.

4.3.2. Inicialización y condiciones de parada de la simulación

La herramienta se inicializa a partir de una serie de parámetros de entrada (definidos en la Subsección 4.4.2). Lo primero que hace NXsim es preparar el entorno y el grafo que definirá la simulación. A continuación despliega los agentes asignando un agente en cada nodo del grafo. Estos agentes son los que llevarán las riendas de la simulación siguiendo el comportamiento que se ha implementado, explicado en la próxima sección.

Finalmente NXsim prepara el agente del entorno y el agente encargado de llevar el registro y almacenado de los datos de la simulación. Una vez llevadas a cabo todas estas preparaciones, arranca la simulación.

Por otra parte, existen varias posibilidades para que la simulación se detenga habiendo finalizado. Una posibilidad es que la simulación exceda el tiempo máximo definido, en este caso se detiene mostrando el estado actual de la red social y el flujo de la publicación hasta este momento. Otra posibilidad es que la simulación termine cuando no queda ningún usuario activo, es decir, que todos los usuarios hayan interactuado ya con la publicación y que no se haya compartido con nadie nuevo. En este caso la simulación no puede avanzar ya que la publicación ha dejado de fluir y, por lo tanto, termina la simulación.

4.4 Implementación de la simulación

Uno de los objetivos principales de este trabajo es poder simular el flujo de información en una red social. Para ello se ha realizado la implementación de varios agentes participantes en una simulación del flujo de una publicación en una red social determinada. Esta publicación es originada por un agente iniciador y es compartida con una serie de amigos de este en la red social. Estos agentes, a su vez, pueden interactuar con la publicación recién recibida con un *like*, comentando en ella, o compartiéndola con su círculo de amigos y propagando este contenido todavía más lejos en la red.

Previo al modelado de la simulación se ha estimado la realización de una serie de modificaciones en la librería NXsim para el completo cumplimiento de los requisitos especificados para este trabajo. A continuación se detallan las modificaciones realizadas antes de llevar a cabo la implementación de la simulación.

4.4.1. Optimización de la librería

NXsim inicialmente despliega a los agentes en los nodos del grafo creado y los ejecuta todos al iniciar la simulación. Dada la naturaleza de nuestro simulador no es necesario que los agentes a los cuales no les ha llegado la publicación estén inicializados, por lo tanto se han realizado modificaciones en la librería para solucionar este problema.

En la simulación existe una lista de agentes iniciadores llamada *seed agents*. Al inicializar la simulación, queremos que únicamente sean lanzados los agentes contenidos en esta lista y sean estos los que al compartir la publicación vayan despertando a los demás agentes en cada unidad de tiempo. Para ello, en vez de activar todos los agentes al lanzar la simulación, únicamente se activan los agentes cuyo identificador esté contenido en la lista de agentes iniciadores. Los demás agentes, únicamente serán activados cuando les vayan a compartir la publicación.

De esta forma se obtiene un ahorro importante de recursos en grandes simulaciones con gran cantidad de agentes puesto que no se inicializarán todos ellos y, además, es posible que la simulación termine sin haber despertado a algunos agentes. En cualquier caso, este ahorro dependerá en gran medida de la topología de la red. Cuanto menos conexa sea la red social existirá una mayor probabilidad de ahorro.

Por otra parte, también se han realizado mejoras en cuanto a lo que al espacio respecta. El *BaseLoggingAgent*, el agente encargado de almacenar y abrir la información resultante de la simulación, realiza una copia del estado de la simulación en cada período indicado en un parámetro de entrada que se explicará en el siguiente apartado. Sin embargo, es posible que únicamente nos interese el estado de la simulación en la última

unidad de tiempo para ahorrar espacio en disco. Para ello se ha habilitado la posibilidad de que este no solo pueda realizar el almacenado en periodos variables indicados por el usuario, si no que además pueda únicamente almacenar el estado de la simulación en la última unidad de tiempo de esta si así se desea. De esta forma se obtiene un ahorro de espacio en el disco equivalente al número de unidades de tiempo transcurridas en forma de porcentaje.

4.4.2. Parámetros de entrada

Antes de iniciar la simulación, esta necesita una serie de parámetros que influirán en su transcurso. Estos parámetros también son útiles para realizar las experimentaciones puesto que permiten ajustar características de la simulación (e.g., duración, agente iniciador, relaciones iniciales, etc.) en función de cada experimento. A continuación se detallan uno a uno con una breve explicación sobre su papel en la simulación:

- **Topology**. La topología del grafo, es decir la red social, sobre la que se desplegarán los agentes.
- **States**. Un diccionario con la información inicial de los estados. Está formado por las claves:
  - *Activity*. Este parámetro representa una probabilidad de que el usuario participe de alguna forma en la red.
  - *Tie-strength*. Este parámetro define un valor de 'confianza' entre cada usuario y sus amigos. Este valor no tiene porque ser simétrico entre dos usuarios.
  - *Post-state*. Esta clave es útil para saber cada usuario en qué estado se encuentra respecto a la publicación; sus valores pueden ser *default*, visto, compartido o rechazado.
  - *Predecessors*. Esta clave contiene una lista con los agentes predecesores. Son todos aquellos agentes que a lo largo de la simulación han decidido compartir, con el agente al que se le atribuye esta lista, la publicación. Se les llama predecesores puesto que preceden al momento de interacción del agente con la publicación.
  - *Commented*. El valor asociado a esta clave determina si el usuario ha dejado un comentario en la publicación y el identificador del agente receptor del comentario.
  - *Tag*. Este parámetro indica, en caso de haber sido etiquetado en una publicación, el identificador del agente que lo ha etiquetado.
  - *Liked*. Finalmente, en el caso de que a un usuario le guste la publicación, esta puede haberse recibido por una o varias vías. Por lo tanto tendremos una lista con los predecesores (usuarios que han compartido la publicación con este usuario) a los que se le ha dado me gusta.
- **Agent-type**. La clase sobre la que está definida el comportamiento de cada agente que se situará en cada nodo y que por lo tanto modela la simulación. En este trabajo se han implementado dos clases que detallaré en el apartado siguiente.
- **Seed**. Lista con el o los identificadores de los agentes que iniciarán la simulación creando la publicación y compartiéndola con sus amigos.
- **Max-time**. Tiempo máximo de la simulación. La simulación funciona por unidades de tiempo sobre las cuales se realizan las capturas de la información de los estados.

Con este parámetro podemos fijar en cuántas unidades de tiempo máximas queremos que nuestro simulador realice estas capturas, forzando así que la simulación finalice en caso de alcanzar este numero de unidades de tiempo.

- **Dir-path**. La dirección donde queremos que se almacenen los datos de la simulación.
- **Num-trials**. El número de simulaciones independientes a realizar en una misma ejecución. Modificando este parámetro podemos lanzar más de una simulación en serie en la misma ejecución.
- **Logging-interval**. El intervalo en el que el agente encargado de guardar los datos tomará las capturas de los datos de la simulación. El valor por defecto es 1, lo que implica que almacenará la información relativa a la simulación en cada unidad de tiempo.

Estos parámetros se pueden configurar de muchas formas diferentes. Por ejemplo, variando la propia topología de la red social permitiendo observar la importancia de esta en el flujo de la información, o bien ajustando las personalidades de los usuarios mediante las variables *activity* o *tie-strength*.

4.4.3. Implementación de los agentes

En este trabajo se han realizado múltiples implementaciones. En cada una los agentes actúan de una forma determinada que repercutirá en los resultados finales obtenidos. En total se han realizado las implementaciones expuestas a continuación.

La primera de estas es la más básica. En esta imlpementación los agentes comparten la publicación con todos sus amigos a la vez con una probabilidad uniforme del 25 %. Además, una vez recibida la publicación, indistintamente de si la comparte o no ya no volverá a interactuar con ella, siguiendo el modelo SIR [3] (*susceptible*, *infectious*, *recovered*). En la Figura 4.4 se representa gráficamente el comportamiento de los agentes de este tipo. Como podemos observar, esta clase de agentes no incorpora la interacción basada en *likes* o comentarios.

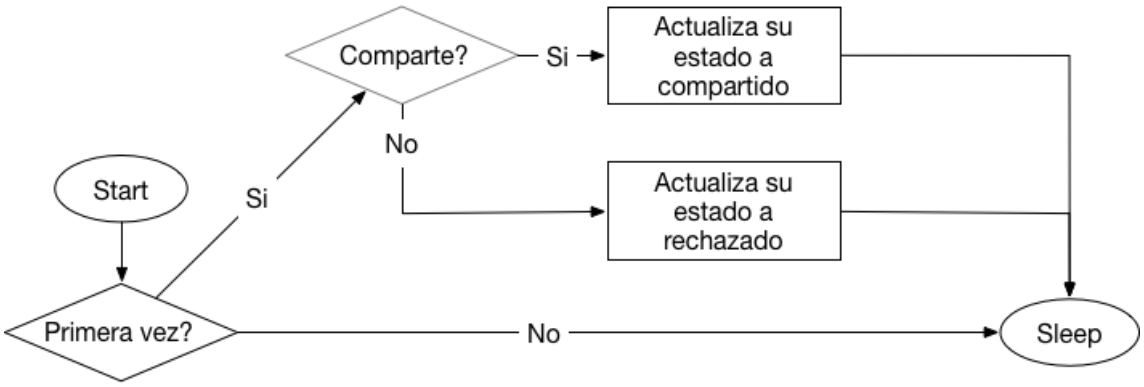

Figura 4.4: Diagrama de flujo del comportamiento de los agentes básicos.

Además de esta implementación básica también se ha diseñado una implementación con mayor complejidad y más acciones. A este modelo de agente le llamaremos agente con reevaluación. Esta clase de agentes tienen la capacidad de decidir a qué amigos dirige la publicación en caso de compartirla, no tiene porqué ser con todos sus amigos a la vez. También tienen la capacidad de interactuar con la publicación dándole me gusta,

comentando o incluso etiquetando a alguno de sus amigos. En esta implementación las variables *activity* y *tie-strength* generadas por el módulo de configuración son las que determinan la actividad del agente y, por lo tanto, no todos los agentes actuarán de la misma forma ni con la misma probabilidad durante la simulación.

Como podemos observar en la Figura 4.5 el agente es iniciado al recibir una publicación. Una vez recibida decide si interactuar con ella comentando o dándole un me gusta. En caso de ser la primera vez que la recibe o si únicamente la ha leído una vez el agente evalúa si compartir. En cualquier caso actualiza su estado y tan solo en caso afirmativo decide a qué amigos enviarla y si etiqueta a alguno de ellos. En caso negativo, si ya había leído la publicación, el agente decide ignorarla en el futuro.

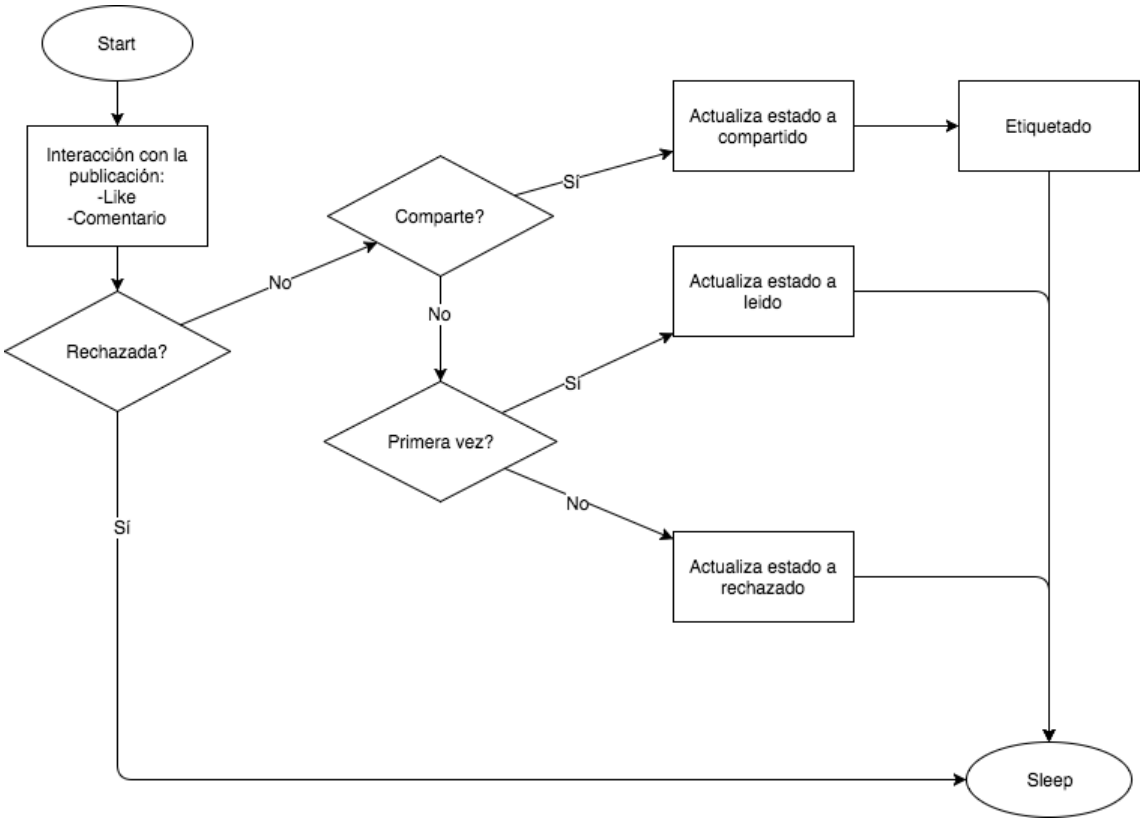

Figura 4.5: Diagrama de flujo del comportamiento de los agentes con reevaluación.

Cada agente puede interactuar con la publicación de diferente forma como se ha mencionado anteriormente en función de las variables *activity*, *tie-strength*. A continuación se explican las funciones principales que realizan los agentes en este simulador:

- **Like**. Esta función permite determinar si el agente otorgará un *like* a la publicación que se le ha compartido.
  - Variables de entrada: El valor de *tie-strength* hacia el agente que le ha compartido la publicación.
  - Salida: En caso de que el valor de *tie-strength* supere un umbral aleatorio, se almacena en el diccionario *states* que el agente le ha otorgado un *like* a la publicación.
- **Comment**. Esta función permite determinar si el agente realizará un comentario en la publicación que se le ha compartido.

  - Variables de entrada: El valor de *tie-strength* hacia el agente que le ha compartido la publicación.
  - Salida: En caso de que el valor de *tie-strength* supere un umbral aleatorio, se almacena en el diccionario *states* que el agente ha dejado un comentario en la publicación.

- **Share**. Esta función permite determinar si el agente va a compartir la publicación y, en caso afirmativo, con quién la va a compartir.

  - Variables de entrada: Una lista con los amigos (nodos vecinos) y el valor de *tie-strength* hacia cada uno de ellos.
  - Salida: En caso de que el valor de *tie-strength* supere un umbral aleatorio, almacena su id en la clave *predecessors* de su amigo. El valor asociado a esta clave permite saber para cada agente su agente predecesor y la unidad de tiempo, es decir, quién le ha compartido la publicación y en qué momento.

- **Tag**. Esta función permite determinar si, además de compartir, el agente etiqueta al amigo en la publicación.

  - Variables de entrada: El valor de *tie-strength* hacia el agente que le ha compartido la publicación.
  - Salida: En caso de que el valor de *tie-strength* duplique el valor de un umbral aleatorio, se almacena en el diccionario *states* que el agente ha etiquetado al amigo con el que ha compartido la publicación.

Una vez finalizadas todas las evaluaciones, el agente se detiene hasta el final de la ejecución o bien hasta que otro agente le comparta la misma publicación, en cuyo caso puede volver a evaluar todos estos parámetros. Hay que remarcar que un agente no compartirá la publicación más de una vez y que por lo tanto, una vez compartida, ya no la volverá a difundir aunque puede seguir otorgando *likes* y realizando comentarios.

4.4.4. Estructura de salida

Durante el transcurso de la simulación, el *BaseLoggingAgent* ha ido tomando las capturas de los estados de los agentes respecto a la publicación. Una vez finalizada, todos estos datos quedan almacenados en la dirección que le hemos indicado en los parámetros a la simulación.

El *BaseLoggingAgent* también es el que se utiliza para leer los datos y poder analizarlos y trabajar con ellos. La información queda almacenada en un diccionario dónde las claves son las unidades de tiempo de la simulación en las que se han tomado las capturas y el valor es otro diccionario con el id del agente como clave y como valor un diccionario con los parámetros de su estado definido anteriormente.

4.4.5. Cálculo del PRS

Para el logro de los objetivos de este trabajo se ha realizado una implementación de la medida de riesgo *Privacy Risk Score* definida en el Capítulo 2. Al finalizar la simulación, se calcula tanto el PRS global como el PRS por capas para todos los agentes. Partiendo del grafo de flujo de la publicación se obtiene el valor del PRS, pudiendo así observar el impacto de las variables *activity* o el *tie-strength* respecto a la privacidad de un agente en la red social.

Para la implementación del PRS global, el primer paso es construir un grafo árbol con el flujo de la publicación para poder trabajar cómodamente con la información. Para ello se añade un vértice a un nuevo grafo vacío por cada agente a_i existente y una arista si existe una entrada en la clave *predecessor* del diccionario de un agente a_i determinado.

Una vez tenemos el grafo construido obtenemos un diccionario con el alcance de cada agente en una simulación determinada. Mediante la obtención de todos los descendientes de un nodo ubicado en el grafo se puede saber la cantidad de agentes a los que este es capaz de hacer llegar sus publicaciones.

Finalmente se calcula el PRS aplicando la fórmula de esta métrica. Para cada agente se obtiene su PRS dividiendo su alcance por el número de agentes en la simulación normalizando así el valor entre 0 y 1.

Por otra parte, se ha realizado la implementación del cálculo del PRS por capas. Esta versión permite ver a cuántos usuarios de las n capas más cercanas en la red social han sido alcanzados por la publicación.

El cálculo de esta versión es ligeramente diferente a la presentada anteriormente. El primer paso es obtener la lista de los agentes presentes en cada capa (de las que se vaya a calcular esta medida). Una vez obtenidos estos agentes debemos realizar una intersección con la lista de agentes alcanzados por el agente del cual se pretende obtener su PRS. Finalmente, aplicando la fórmula del PRS por capas con estos valores obtenemos el resultado. En este caso se obtiene dividiendo los agentes de un nivel dado alcanzados por el número total de agentes en ese nivel. De esta forma, el valor queda también normalizado entre 0 y 1 siendo el 1 un nivel de riesgo máximo.

Este valor nos indica el nivel de riesgo que tiene cada agente de, al publicar algo, ser leído por agentes potencialmente peligrosos o simplemente no deseados o desconocidos. Esta métrica será de gran utilidad en las posteriores experimentaciones.

4.4.6. Procesamiento gráfico de los datos

Finalmente, para una mayor comodidad y una mejor interpretación de la información obtenida al finalizar la simulación, se ha implementado la herramienta de manera que procese gráficamente esta información. Este procesado gráfico esta dividido en dos partes, la primera con el objetivo de obtener gráficos de lineas con los datos relativos al número de agentes y su estado respecto a la publicación; y la segunda para obtener un grafo del flujo de la publicación.

Los gráficos de lineas contienen la información de cuántos agentes han interactuado con la publicación en cada unidad de tiempo y en qué estado se encuentran. En la Figura 4.6 podemos observar que el eje X representa las unidades de tiempo con las que se desarrolla la simulación y el eje Y el número de agentes existentes en la simulación. Estos gráficos tienen cinco lineas diferentes: la primera nos indica el número de agentes o usuarios que han leído la publicación, la segunda cuántos de estos han decidido compartirlo y la tercera cuántos lo han recibido más de una vez y han tomado la decisión de ignorarlo. Las últimas dos lineas aportan información respecto al número de usuarios que han considerado otorgar me gusta a la publicación y aquellos que la han comentado. Estos gráficos nos facilitan ver el nivel de difusión que ha habido en una simulación determinada. La herramienta desarrollada genera un gráfico por simulación lanzada, y en caso de que el número de simulaciones en la misma ejecución sea igual o mayor que dos, calcula también un gráfico con la media de los datos.

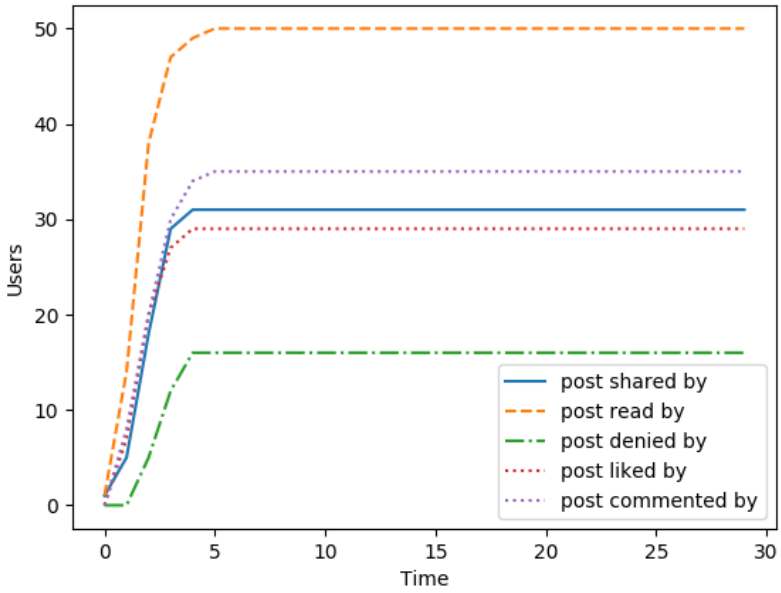

Figura 4.6: Gráfico de lineas

Los grafos de flujo, como podemos observar en la Figura 4.7, nos muestran una serie de nodos, que representan los agentes unidos por arcos, que representan el paso de la información entre esos dos agentes. Mediante este grafo dirigido se puede observar rápidamente si ha habido grupos aislados a los que la información no ha alcanzado o si la publicación ha pasado más de una vez por la misma persona. En este caso, la herramienta desarrollada obtiene un grafo por cada simulación lanzada dentro de la misma ejecución.

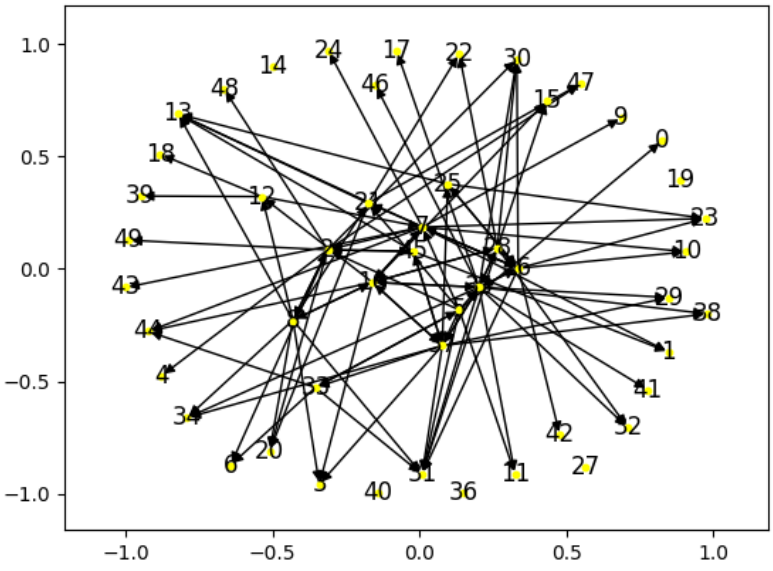

Figura 4.7: Grafo de flujo

En conclusión, mediante este módulo de procesado gráfico de los datos se pueden ver rápidamente algunos rasgos importantes del transcurso de la simulación como, por

ejemplo, si ha habido mucha o poca interacción. Este rasgo puede ser útil para clasificar los resultados de las simulaciones. Por otra parte, también podemos observar la posible existencia de comunidades más cerradas mediante los grafos de flujo. En caso de existir dos componentes con mayor conectividad en el grafo, esto querrá decir que existen dos grupos de personas relacionadas entre sí.

CAPÍTULO 5
Casos de estudio

En este capítulo se han propuesto dos casos de estudio diferentes, un primer caso de estudio sintético extraído de [2] para validar el funcionamiento de la herramienta y un segundo caso de estudio real que permita observar el proceso de experimentación mediante el simulador desarrollado. De esta forma quedan definidos los dos principales usos que se le puede dar a este simulador, experimentos con carga sintética por una parte y modelados de situaciones reales para su estudio en profundidad por otra.

Para ambos casos se ha hecho uso de la métrica de privacidad PRS. Tanto para el primer como para el segundo caso de estudio se ha realizado una comparativa de los valores obtenidos por el simulador desarrollado. El primer caso de estudio se ha realizado sobre redes generadas aleatoriamente siguiendo tres topologías determinadas. El segundo caso, sin embargo, se ha realizado sobre una red social real, la red social PESEDIA [6].

A lo largo de este capítulo, por lo tanto, se explicarán los experimentos realizados en el simulador desarrollado. Para ello, se han modelado una serie de agentes diferentes. En cada apartado se definirá el modelo de agente utilizado así como los parámetros de estos para cada caso de estudio realizado. Además, tras presentar los casos de estudio se ha realizado un análisis de los resultados obtenidos tras la simulación. En el primer caso de estudio, el objetivo ha sido el de validar el simulador. Por otra parte, en el segundo caso de estudio se han comparado distintos modelos de comportamiento del agente sobre una red existente, con el objetivo de ejemplificar un experimento con esta herramienta.

5.1 Caso Sintético

El principal objetivo de este caso de estudio es la validación de la herramienta de simulación. Para ello se han realizado experimentos sobre tres modelos de red aleatorios distintos, descritos a continuación, con el propósito de comparar los valores de PRS obtenidos con los valores que se obtuvieron en el experimento propuesto en [2].

5.1.1. Definición del caso

Para este primer caso de estudio se ha utilizado el modelo de agente básico. Estos agentes, descritos en el capítulo anterior, cuentan con una probabilidad de acción uniforme, y únicamente interactúan compartiendo la publicación. Para este caso de estudio se han realizado tres experimentos sobre tres modelos de redes aleatorios. Estos modelos son: Watts-Strogatz [13], Erdös-Rény [7] y Barabasi-Albert [5], en la siguiente sección se ha detallado cada uno de ellos. Para el experimento se han generado tres redes de 1000

usuarios siguiendo estos modelos y se han lanzado 400 simulaciones por cada agente como agente iniciador o *seed agent* de la publicación. Además, la política de privacidad adoptada por los agentes será la de compartir únicamente con amigos.

En la Tabla 5.1 podemos observar de manera sintetizada los parámetros que definen este primer experimento.

Simulaciones	400×1000 (agentes)
Topologías utilizadas	Watts-Strogatz Erdös-Rény Barabasi-Albert
Acciones de los agentes	Compartir
Probabilidad de acción	Uniforme
Política de privacidad	Amigos

Tabla 5.1: Parámetros caso básico

A continuación están detallados los tres modelos de red utilizados en los experimentos de este primer caso de estudio. Se han definido las propiedades principales de cada uno de ellos así como sus características estructurales que puedan influir en el posterior flujo de la información.

5.1.2. Modelos utilizados

Modelo Watts-Strogatz

El modelo de red Watts-Strogatz es utilizado para la creación de redes de mundo pequeño, concretamente redes aleatorias con distancias medias entre nodos bajas y con alto coeficiente de *clustering*. Esto quiere decir que los nodos del grafo estarán muy agrupados entre sí. Las redes de mundo pequeño son aquellas en las que la mayoría de nodos no son vecinos entre sí pero son alcanzables desde cualquier otro nodo en un número de aristas relativamente pequeño. Estas redes de mundo pequeño son comúnmente aplicadas en el modelado de redes sociales, el ámbito de este trabajo.

En la Figura 5.1 podemos observar un ejemplo de red generada mediante NetworkX siguiendo el modelo Watts-Strogatz. Esta red se compone de 20 nodos, siendo N el número de nodos. Como grado medio de cada nodo se ha asignado k=4. Finalmente al parámetro especial β se le ha asignado la probabilidad de 0.2, la cual se tiene en cuenta a la hora de la generación y determina la probabilidad de que una arista sea reorientada. Como se puede observar en el ejemplo simplificado, no existen vértices aislados y, en general, todos los nodos se encuentran a una distancia pequeña de los demás nodos.

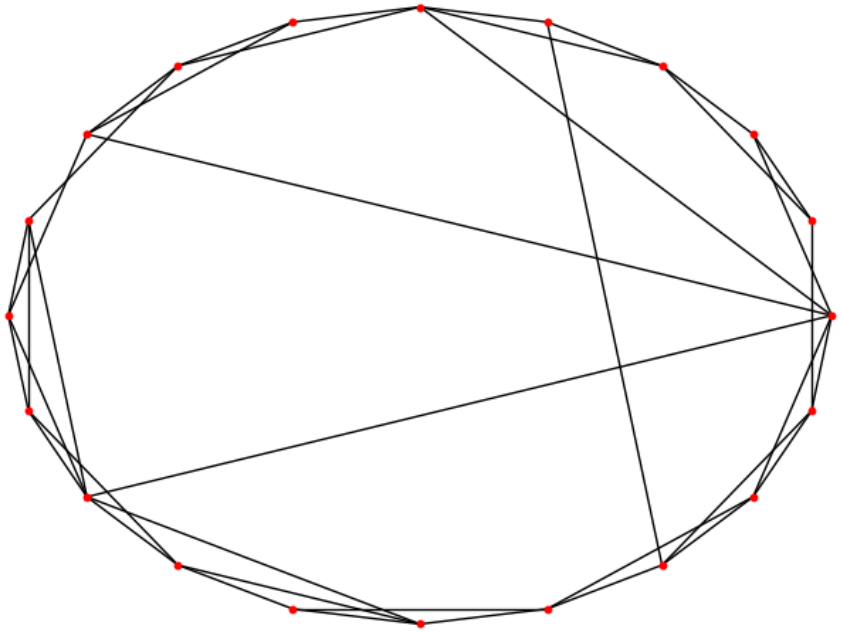

Figura 5.1: Modelo Watts-Strogatz con N=20, k=4, β=0.2

Modelo Erdös-Rény

El segundo modelo utilizado para este caso de estudio es el modelo Erdös-Rény. En este modelo para la generación aleatoria de grafos los nodos se unen al resto de nodos del grafo con la misma probabilidad. Es por esto que mediante el uso de este modelo se garantiza la existencia de independencia estadística entre los nodos del grafo.

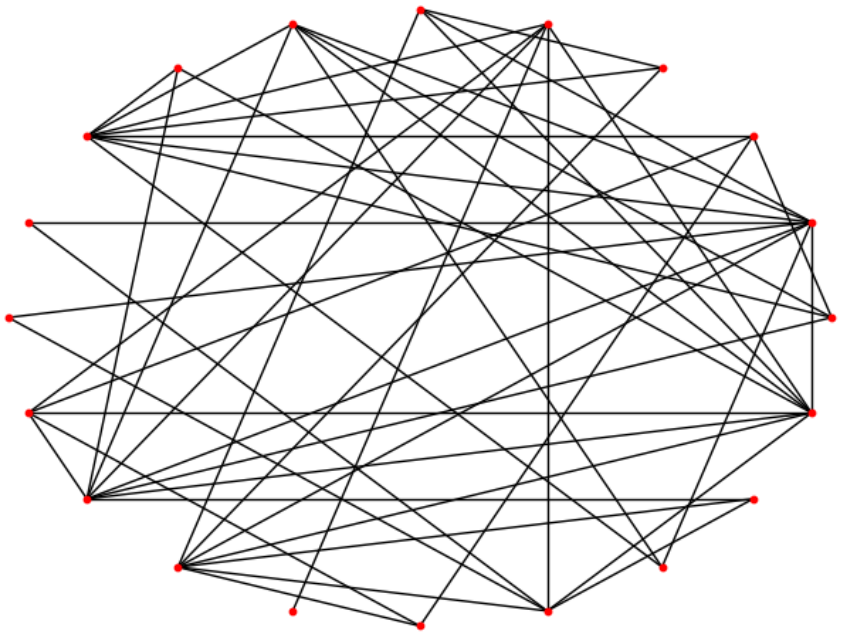

Figura 5.2: Modelo Erdös-Rény con N=20, p=0.2

En la Figura 5.2 podemos observar una pequeña red generada mediante este modelo. Esta red se ha generado con los parámetros N=20, siendo este valor el número de

nodos y p=0.2, definiendo esta la probabilidad de creación de aristas. Como podemos observar, los nodos de este grafo no muestran tanta uniformidad como los del modelo Watts-Strogatz. En el grafo generado por este modelo existen nodos con grado muy elevado proporcionalmente al número de nodos y otros con el grado muy pequeño.

Modelo Barabasi-Albert

Finalmente, el tercer modelo utilizado en este caso de estudio es el modelo Barabasi-Albert. Este modelo también es utilizado para la generación aleatoria de redes. Concretamente es útil para generar redes complejas libres de escala. Una red libre de escala es una red compleja la cual puede tener nodos con elevado grado de conexión pese a que el grado medio de la red sea bajo. Por lo tanto, existirán nodos conectados a muchos otros aunque la mayoría de los nodos tenga pocas aristas asociadas. Las redes generadas mediante el modelo Barabasi-Albert tienen una distribución de grado potencial. Inicialmente se generan *m* nodos conectados aleatoriamente entre sí. A partir de aquí se añaden nuevos nodos conectándolos aleatoriamente a *m* nodos existentes hasta alcanzar el número de nodos definido.

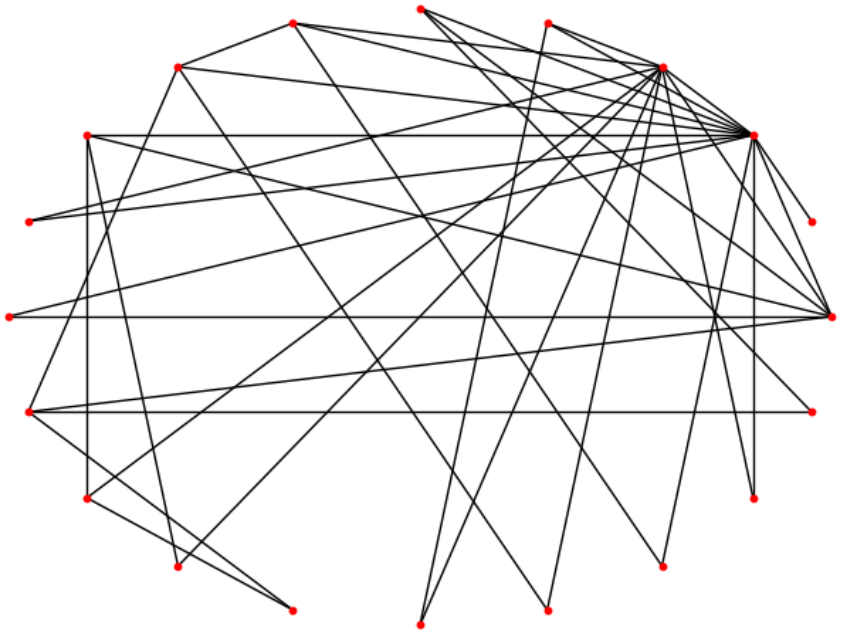

Figura 5.3: Modelo Barabasi-Albert con N=20, m=2

En la Figura 5.3 se puede apreciar una pequeña red generada siguiendo este modelo. Esta red se ha definido con N=20 y m=2, es decir la red se compone de 20 nodos y se genera inicialmente con 2 nodos conectados entre sí y al añadir cada nuevo nodo conectándolo con 2 nodos ya existentes. En este caso, podemos observar que los dos nodos con mayor grado son, probablemente, los dos nodos iniciales de la red en su proceso de generación. Por otra parte, los nodos de únicamente grado 2, podemos deducir que son, con mayor probabilidad, los últimos nodos en añadirse a la red mediante este modelo.

Una vez definido el caso de estudio y los tres modelos utilizados, a continuación se van a analizar los resultados obtenidos tratando de validar la herramienta de simulación desarrollada a lo largo de este trabajo.

5.1.3. Evaluación de resultados

En esta sección se van a analizar los datos obtenidos con el objetivo de validar la herramienta de simulación. Para realizar la comparación de los resultados obtenidos se ha aplicado el coeficiente de correlación de Pearson [8, 11], valor que indica la correlación lineal entre dos variables. Puede adoptar un valor en el rango [-1, 1] siendo -1 una relación total negativa, 0 si no existe relación lineal alguna y 1 en caso de relación total positiva. En caso de relación total positiva, si un valor aumenta en una variable también lo hace en la otra. Cabe destacar que en el transcurso de la simulación existen variables aleatorias que pueden implicar que la correlación de los datos resultantes se vea mermada.

A continuación tenemos una serie de representaciones gráficas de los valores de PRS obtenidos en los experimentos. En el eje Y se encuentra el valor de la media del PRS asociado a cada agente, y en el eje X el identificador del agente. A la izquierda se puede observar la representación gráfica del PRS obtenido en el experimento propuesto en [2], a la derecha el valor del PRS obtenido con la herramienta de simulación desarrollada en este trabajo.

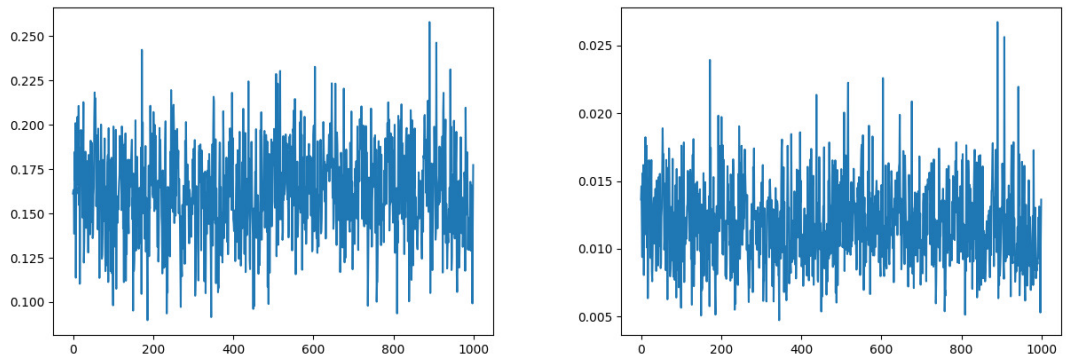

Figura 5.4: Comparación de los valores de PRS obtenidos en el experimento con redes Watts-Strogatz

El primer experimento se ha realizado con la red Watts-Strogatz. Una vez finalizadas las simulaciones mediante el módulo de PRS se han obtenido los valores de PRS medios correspondientes a todos los agentes presentes en la red. En la Figura 5.4 se presenta la comparación gráfica de los datos. Como se puede observar, al ser una red con los nodos de grado muy similar, los valores de PRS se encuentran todos bastante próximos. Tras aplicar el coeficiente de correlación de Pearson en este experimento se ha obtenido el valor de 0.95, el cual indica una correlación lineal entre ambos PRS casi perfecta.

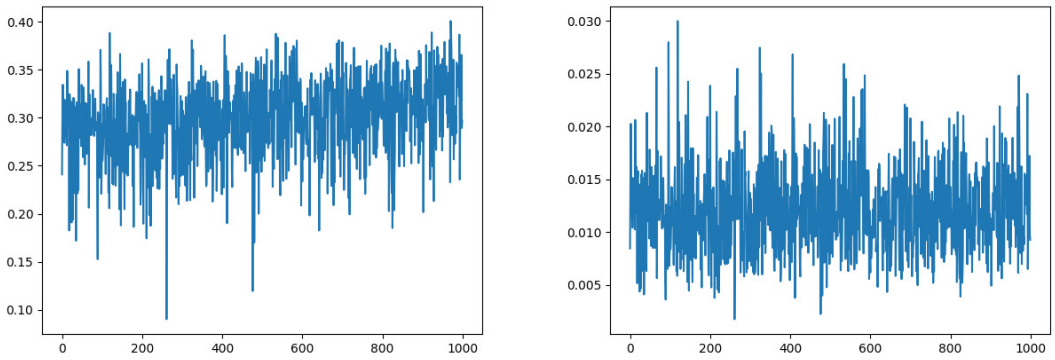

Figura 5.5: Comparación de los valores de PRS obtenidos en el experimento con redes Erdös-Rény

Para el segundo experimento se ha usado la red generada mediante el modelo Erdös-Rény. En la Figura 5.5 podemos observar la distribución de los valores de PRS en los agentes en este experimento. Tras aplicar el coeficiente de correlación de Pearson se ha obtenido el valor de 0.92. Este valor tan cercano a 1 implica que los resultados obtenidos en ambos experimentos también siguen una correlación lineal casi perfecta.

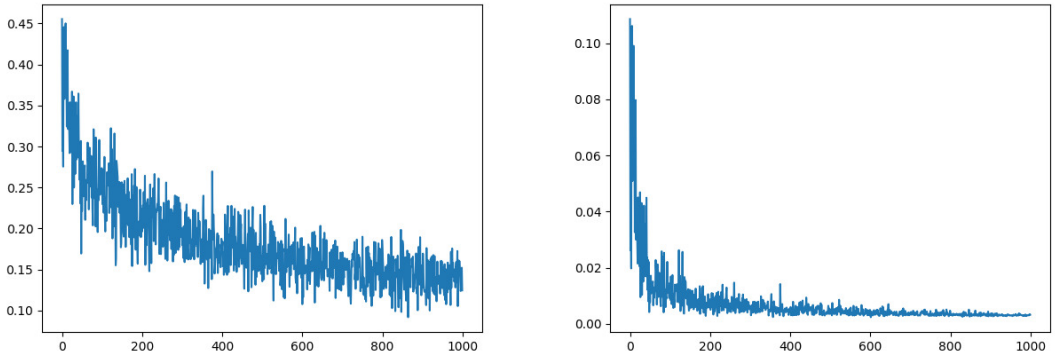

Figura 5.6: Comparación de los valores de PRS obtenidos en el experimento con redes Barabasi-Albert

El tercer experimento se ha realizado con la red modelada mediante Barabasi-Albert. La Figura 5.6 muestra la distribución de los valores de PRS de los agentes. En este experimento también se observa una distribución característica. Los primeros agentes tienen un valor de PRS más elevado que los últimos. Esto es debido a la naturaleza de este modelo. Como se explica en la sección anterior, al ser los primeros agentes los que más conexiones tienen es más probable que su publicación alcance mayor número de usuarios y, por lo tanto, tengan un valor más elevado de PRS. Aplicando el coeficiente de correlación de Pearson se obtiene el valor de 0.81. En este caso, también indica fuerte correlación lineal entre ambos experimentos.

Como conclusión a esta sección es importante destacar dos observaciones. Las distribuciones de los experimentos realizados en la herramienta de simulación siguen las distribuciones esperadas en función de cada modelo, como se ha podido comprobar. Por otra parte, el coeficiente de correlación de Pearson entre los resultados obtenidos por la herramienta de simulación desarrollada y los datos obtenidos en [2] son muy cercanos a 1, implicando esto la existencia de una fuerte correlación lineal entre ambos resulta-

dos. De esta forma queda validado el funcionamiento de la herramienta de simulación desarrollada en este trabajo.

5.2 Caso PESEDIA

El segundo caso de estudio que se ha realizado en este trabajo tiene como objetivo la demostración del uso del simulador desarrollado mediante un experimento con datos reales. Con la herramienta validada en el anterior caso de estudio mediante la comparación de los datos obtenidos mediante la reproducción de un experimento cond atos sintético, este caso de estudio pretende mostrar algunas de las capacidades del simulador a la hora de lidiar con datos reales.

Para este caso de estudio se ha utilizado la topología y los datos de una red social real, la red social PESEDIA [6]. Esta red social nace como parte del proyecto nacional 'Privacidad en Entornos Sociales Educativos durante la Infancia y la Adolescencia' con el objetivo de concienciar y asesorar a usuarios jóvenes en el uso de redes sociales en un mundo dónde el uso de estas está totalmente extendido en la sociedad. Los datos utilizados para este caso de estudio corresponden con el uso de esta red social en un periodo de un mes.

En la siguiente sección se definen todos los parámetros utilizados en este caso de estudio, desde los modelos de agente utilizados hasta los datos reales de la red PESEDIA utilizados para modelar estos agentes.

5.2.1. Definición del caso

Para mostrar el comportamiento de los agentes modelados en el trabajo, se han realizado varios experimentos con distintos parámetros. Dado que este caso de estudio parte de una red social real, una de las tareas a realizar en este caso ha sido la definición de los atributos propios de los agentes de la herramienta, a partir de datos reales obtenidos del uso de PESEDIA. En la Tabla 5.2 se pueden apreciar los principales parámetros utilizados en este caso de estudio.

Simulaciones	$t \times 214$ (agentes)
Topologías utilizadas	Topología red Pesedia
Acciones de los agentes	Compartir, *like*, comentar, etiquetar
Probabilidad de acción	Basada en parámetros *activity* y *tie-strength*
Política de privacidad	Variable

Tabla 5.2: Parámetros caso Pesedia

El número de simulaciones t realizadas por cada agente, en este caso, no es igual para todos ellos. A partir de los datos de la red PESEDIA se han extraído aquellos usuarios que publicaron más y cuáles publicaron menos, y se ha simulado en relación a estos datos. De esta manera, si en el transcurso del uso de la red algún usuario no realizó ninguna publicación, este núnca sera *seed agent* o agente iniciador en esta simulación. En cambio, si un usuario fué muy activo realizando publicaciones, en la simulación será este quien inicie mayor número de *trials* en proporción.

En este caso de estudio, como ya se ha mencionado, se ha utilizado la topología de la red social PESEDIA. Esta red, compuesta por 214 usuarios ha sido simulada por 214 agentes que han reproducido el comportamiento de cada uno de los usuarios de la red. En

la Figura 5.7 se puede apreciar como la topología de red de PESEDIA está dividida en dos grandes agrupaciones de nodos. Esto es debido a que esta red social se usó por alumnos de dos cursos diferentes, por lo tanto cada agrupación de nodos en el grafo corresponde al grupo de usuarios de cada curso. Además se puede apreciar que el tamaño de los nodos es diferente. Los nodos de mayor tamaño son aquellos nodos con mayor número de relaciones, en cambio los nodos de menor tamaño son aquellos que se encuentran menos integrados en la red social.

Figura 5.7: Red social PESEDIA

Por otra parte, los agentes en este caso de estudio disponen de un mayor número de acciones posibles a realizar en el momento de interactuar entre ellos. Aunque para el cálculo del PRS únicamente influya el nivel de difusión y el alcance de una publicación, las demás acciones pueden ser realmente interesantes a la hora de realizar experimentos futuros sobre la herramienta desarrollada en este trabajo, como por ejemplo, experimentos sobre los niveles de confianza entre varios usuarios. Estas acciones son el *like*, que consiste en dar me gusta a una publicación, comentar en una publicación o bien etiquetar a un amigo al compartir o realizar una publicación.

En este caso de estudio, la probabilidad de acción de los agentes no es uniforme ni única para todos los experimentos realizados. Para definir la probabilidad de acción de los agentes se han utilizado los parámetros *activity* y *tie-strength* definidos en el capítulo anterior. El parámetro *activity* para definir el nivel de involucración de un agente en la red social y el parámetro *tie-strength* para definir el nivel de amistad de un agente con cada uno de los que está relacionado. Para la definición de estos parámetros se han utilizado datos reales provenientes de la red social PESEDIA asociados a cada usuario. A continuación se definen los experimentos realizados a partir de estos datos. En cada experimento se ha detallado el criterio utilizado para la obtención de los parámetros *activity* y *tie-strength*.

5.2.2. Experimento 1

En el primer experimento realizado en este caso de estudio se ha utilizado el modelo de agente con reevaluación. El ciclo de vida de este modelo se puede observar en la Figura 4.5. Este modelo de agente concede una segunda oportunidad para compartir la publicación después de haber rechazado compartirla una primera vez.

Por otra parte, los parámetros *activity* y *tie-strength* para este experimento se han modelado a partir de los datos de uso de PESEDIA. En primer lugar, *activity* se ha modelado como la media de comparticiones realizadas en la red. A partir del parámetro *shared-posts* obtenido de PESEDIA se ha obtenido el número de publicaciones que ha compartido cada usuario. Posteriormente se ha realizado un normalizado de este valor y se ha asignado a la variable de actividad de cada agente. La variable *tie-strength* en este experimento se ha obtenido mediante una calificación que realizaron los usuarios sobre su nivel de amistad con sus amigos en la red. Este valor varia de 1 a 5. Para la simulación se ha realizado un normalizado de este valor entre 0 y 1.

Con todos los parámetros definidos y el modelo de agente preparado se han lanzado múltiples simulaciones tratando de reproducir lo que sucedió en la red social PESEDIA.

5.2.3. Experimento 2

Este segundo experimento se ha realizado de forma muy similar al primero. El tipo de agente utilizado en este experimento es el agente básico definido en la Figura 4.4. Este modelo de agente al recibir la publicación por parte de un amigo únicamente tiene una oportunidad para decidir cómo interactuar con ella. De esta forma, si un agente decide no seguir compartiendo la publicación en una simulación determinada, este agente queda fuera del flujo de la publicación en esa simulación.

Para este segundo experimento las variables *activity* y *tie-strength* se han modelado a partir de los mismos datos recogidos de PESEDIA que el primer experimento. Del mismo modo que en el primer experimento se han realizado varias simulaciones con el objetivo de reproducir los hechos sucedidos en la red social PESEDIA.

5.2.4. Experimento 3

Finalmente, en este último experimento realizado con el simulador se ha utilizado el agente con reevaluación. Para este experimento la variable *activity* se ha modelado de la misma forma que en los dos experimentos anteriores, a partir del número de publicaciones compartidas a lo largo del uso de la red social. Sin embargo, la principal diferencia proviene de la definición de la variable *tie-strength*. Para el modelado de esta variables en este tercer experimento se ha prescindido del valor que se asignaron los usuarios entre ellos en la red social. En este experimento se ha asumido que los usuarios no tienen en cuenta su nivel de confianza con sus amigos a la hora de interactuar en la red. Es por esto que se ha asignado el valor máximo para el valor de *tie-strength* para todas las relaciones.

Una vez diseñados estos tres experimentos y lanzadas las simulaciones se ha realizado una comparación de los datos obtenidos con los datos que representan lo ocurrido en la red social. Cabe destacar que la definición de variables de comportamiento de los agentes en sí excede el alcance de este trabajo puesto que no es una tarea sencilla y podría consistir en otro trabajo independiente. Este caso de estudio únicamente se ha realizado con el objetivo de ejemplificar un experimento usando la herramienta desarrollada a lo largo de este trabajo.

5.2.5. Evaluación de resultados

En esta sección, se han comparado los valores obtenidos de PRS en cada experimento con los valores de PRS precalculados sobre el historial de uso de la red social PESEDIA. De la misma forma que en el primer caso de estudio, la coordenada Y de las tablas consiste en la media del valor de PRS, en un intervalo de 0 a 1, y la coordenada X indica el identificador del agente, en este caso de 0 a 213.

Dado que existen factores aleatorios en el desarrollo de la simulación, se ha considerado realizar la comparación mediante el coeficiente de correlación de Pearson [8, 11]. Este coeficiente es una medida de la relación lineal existente. De esta manera podemos evaluar si, pese a tener valores diferentes, en ambos casos los valores de PRS siguen una distribución igual. Esto implicaría que tanto en los datos reales como en los de la simulación los usuarios y agentes habrían adoptado actitudes muy similares.

Para el primer experimento se han utilizado los agentes con reevaluación, y modelando sus atributos con los datos de PESEDIA se han obtenido los resultados observables en la Figura 5.8. El coeficiente de correlación de Pearson en este experimento es del 0.41, indicando esto que pese a existir similitudes en algunos valores no termina de existir una correlación lineal perfecta.

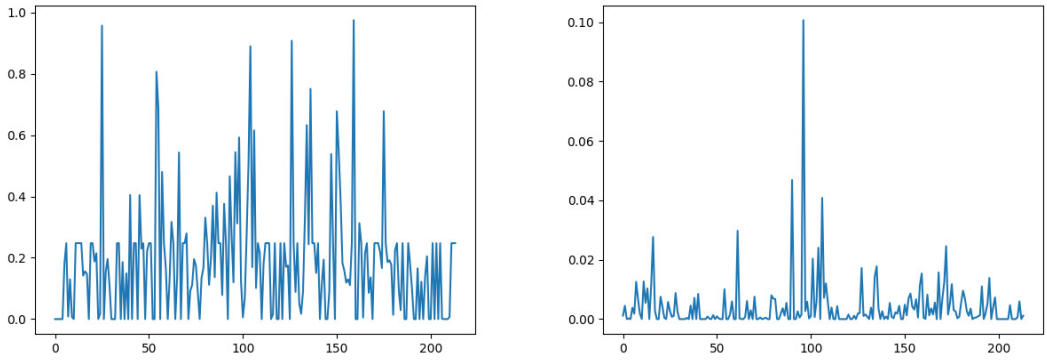

Figura 5.8: Comparación de los valores de PRS obtenidos en el experimento 1 con la topología de PESEDIA

Para el segundo experimento se decidió usar el modelo de agente básico con los mismos atributos que en el experimento anterior. Esta elección tiene como propósito ver el nivel de repercusión que puede tener ese paso de reevaluación a la hora de decidir si compartir o no una publicación. Los resultados obtenidos frente a los reales se pueden observar en la Figura 5.9. El valor del coeficiente de correlación de Pearson en el segundo experimento es del 0.42. De esta forma se ha observado que ambos han dado resultados muy parejos y ninguno sin acercarse a la relación lineal perfecta. Al igual que en el experimento anterior, este valor indica que existen similitudes en algunos valores más característicos pero sin existir una relación lineal muy fuerte.

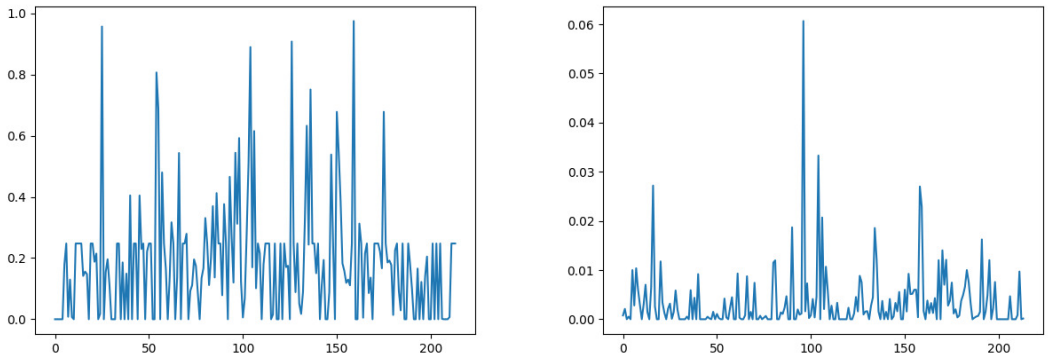

Figura 5.9: Comparación de los valores de PRS obtenidos en el experimento 2 con la topología de PESEDIA

Finalmente, para el tercer experimento, se ha decidido usar el modelo de agente con reevaluación pero replanteando los atributos de este. La principal diferencia en este tercer experimento reside en la asignación de la variable *tie-strength* entre usuarios. En la Figura 5.10 se puede observar la diferencia entre los valores obtenidos por el simulador y los valores reales de PESEDIA. El coeficiente de correlación de Pearson en este experimento ha sido de 0.57, un valor sustancialmente mayor a los valores obtenidos en los anteriores experimentos. Este valor indica que la correlación lineal entre el PRS calculado de PESEDIA y el PRS calculado al finalizar las simulaciones tiene cierto parecido. Hay que tener en cuenta que existen factores aleatorios interviniendo en el desarrollo de la simulación, por lo tanto es muy complicado que exista una correlación perfecta.

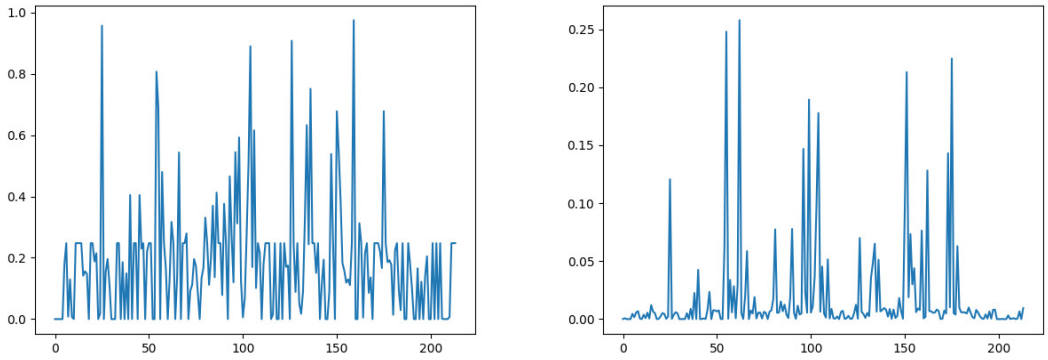

Figura 5.10: Comparación de los valores de PRS obtenidos en el experimento 3 con la topología de PESEDIA

Las principales conclusiones que se pueden extraer de este segundo caso de estudio son, en primer lugar, la complejidad de realizar simulaciones de situaciones reales como interacciones en una red social. Esto es debido a la cantidad de posibilidades a la hora de modelar variables a partir de los datos reales. En segundo lugar, a partir de los dos primeros experimentos se ha podido observar la poca repercusión ha tenido la diferencia entre los dos modelos de agente implementados en la obtención y el cálculo del PRS. Finalmente, en el tercer experimento se ha podido observar como algunos de los valores reales pueden ser confusos. Aunque lo más razonable hubiese sido asumir que el valor de *tie* era el que se asignaron durante el uso de la red, se han obtenido mejores resultados asignando el máximo valor a todos. Esto puede parecer un error, sin embargo no es así.

En el transcurso del periodo en el que se usó la red social, sus usuarios iban guiados por objetivos, es por esto que es posible que los usuarios no hayan tenido en cuenta su nivel de confianza con sus amigos a la hora de interactuar, y por ello este experimento haya dado mejores resultados.

CAPÍTULO 6
Conclusión

6.1 Conclusiones

El modelado y desarrollo de una herramienta de simulación consiste en un proceso complejo en el cual es importante dominar múltiples tecnologías y aplicar varias metodologías. Para ello es necesario, en primer lugar tener la herramienta más adecuada para la simulación que se desee realizar. Pero también es de gran importancia realizar un modelado acertado de los agentes, puesto que los datos sintéticos obtenidos al finalizar una simulación deben ser datos viables en un caso real. Es por ello que para la correcta realización de este trabajo se ha hecho uso de gran parte de las competencias y conceptos aprendidos a lo largo del grado.

Al comienzo de este trabajo se definieron una serie de objetivos clave para determinar el trabajo de este proyecto. Todos estos objetivos han sido alcanzados en el momento de finalizar el trabajo.

Antes de empezar se realizó un estudio y análisis de las herramientas vigentes que pudiesen ser útiles para simular comportamientos en redes sociales. Para poder seleccionar la herramienta que más se adecuaba a este trabajo fueron muy importantes una serie de capacidades desarrolladas a lo largo del grado. Por una parte, la de búsqueda de información eficientemente. Por otra, la capacidad de discriminación entre distintos sistemas o herramientas en función de factores como el rendimiento, la escala o su arquitectura.

Se ha realizado una especificación formal de requisitos con el objetivo de determinar algunos de los aspectos sobre el desarrollo del simulador como sus funcionalidades. Este proceso se ha realizado siguiendo las técnicas aprendidas durante el grado. Además, se ha podido observar la importancia de esta parte en un desarrollo real de un proyecto.

El proceso de implementación de la herramienta de simulación ha sido un proceso que ha englobado el uso de varios conocimientos adquiridos en la esta universidad. Por una parte, en la modificación de la herramienta se ha tenido que trabajar con la gestión de procesos y la concurrencia entre ellos. Por otro lado, se ha realizado el modelado de una serie de agentes inteligentes siguiendo todas las propiedades de estos. Además, también se ha tenido que trabajar en tratamiento y procesado de datos al tener datos reales sobre los que realizar simulaciones, así como en la representación gráfica de estos datos para un mejor análisis de ellos.

Finalmente, se han realizado las pruebas necesarias para la validación de la herramienta, alcanzando así todos los objetivos iniciales planteados en este trabajo. Además de un experimento con datos reales exhibiendo el potencial del simulador desarrollado.

En conclusión, mediante todo un abanico de conocimientos adquiridos durante el grado, se ha desarrollado una herramienta de simulación multiagente y se ha conseguido

validar su funcionamiento. En un futuro esta herramienta será útil para experimentar con nuevos modelos de agente, y poder explorar nuevas posibilidades en el mundo de la investigación en redes sociales como el análisis de medidas de confianza entre usuarios.

6.2 Trabajo futuro

Uno de los principales usos que se le puede dar a una herramienta de simulación puede ser la de realizar experimentos en ámbitos como el de la investigación. En la realización de estos experimentos suele interesar lanzar grandes baterías de simulaciones, como se ha realizado en el Capítulo 5. Aunque una simulación individual en la herramienta de simulación actual no implique un coste temporal significativo, lanzar estas baterías de simulaciones sí puede implicar un coste temporal notable. Una posible solución a este problema puede consistir en paralelizar el simulador. Mediante esta técnica sería posible lanzar simultáneamente varias simulaciones en distintos procesadores, reduciendo así en gran medida el tiempo de duración de la simulación.

Por otra parte, una de las posibles tareas a realizar en un futuro para facilitar el uso de la herramienta desarrollada a los usuarios noveles o sin experiencia en lenguajes de programación, consiste en el diseño y la implementación de una interfaz. Mediante la implementación de una interfaz, el simulador sería totalmente transparente para el usuario pudiendo así configurar los parámetros de la simulación sin tener que entrar en contacto con el código del programa.

Finalmente, otra linea de trabajo futuro consiste en la adaptación y el uso del simulador para el análisis de medidas de confianza entre usuarios de redes sociales. Mediante el uso de esta herramienta se pretende realizar una comparativa de distintos algoritmos de confianza existentes así como realizar nuevas propuestas con el objetivo de encontrar uno que obtenga los valores de confianza más próximos a un caso real.

Bibliografía

[1] Sameera Abar, Georgios K Theodoropoulos, Pierre Lemarinier, and Gregory MP O'Hare. Agent based modelling and simulation tools: A review of the state-of-art software. *Computer Science Review*, 24:13–33, 2017.

[2] J. Alemany, E. delVal, J. Alberola, and A. García-Fornes. Estimation of privacy risk through centrality metrics. *Future Generation Computer Systems*, 82:63 – 76, 2018.

[3] Linda JS Allen, Fred Brauer, Pauline Van den Driessche, and Jianhong Wu. *Mathematical epidemiology*, volume 1945. Springer, 2008.

[4] Edouard Amouroux, Thanh-Quang Chu, Alain Boucher, and Alexis Drogoul. Gama: an environment for implementing and running spatially explicit multi-agent simulations. In *Pacific Rim International Conference on Multi-Agents*, pages 359–371. Springer, 2007.

[5] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[6] José Alemany Bordera. Pesedia. red social para concienciar en privacidad. 2016.

[7] Paul Erdös and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(17-61):43, 1960.

[8] Francis Galton. Regression towards mediocrity in hereditary stature. *The Journal of the Anthropological Institute of Great Britain and Ireland*, 15:246–263, 1886.

[9] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, Keith Sullivan, and Gabriel Balan. Mason: A multiagent simulation environment. *Simulation*, 81(7):517–527, 2005.

[10] GREG MP O'Hare. Agent factory: an environment for the fabrication of multi-agent systems. *Foundations of Distributed Artificial Intelligence (GMP O'Hare and N. Jennings eds) pp*, pages 449–484, 1996.

[11] Karl Pearson. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58:240–242, 1895.

[12] J. Smith, C. Perrone, and A. Repenning. Agentsheets common ground: A collaborative learning tool. *ECOOP '96*, 1996.

[13] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440, 1998.

[14] Michael Wooldridge. *An introduction to multiagent systems*. John Wiley & Sons, 2009.