El uso de ontologías distribuidas en aplicaciones P2P

The Use of Distributed Ontologies in P2P Applications

Jorge Boticario Figueras

UNIVERSIDAD COMPLUTENSE
MADRID

TRABAJO DE FIN DE GRADO
Grado en Ingeniería Informática
Facultad de Informática

Director
Simon Pickin

Madrid, 2020–2021

Agradecimientos

A mis padres, por inculcarme la importancia de los estudios y el esfuerzo. A mis hermanas, por apoyarme durante todos estos años.

A mis compañeros de carrera y mis amigos de toda la vida, por ayudarme siempre que lo he necesitado.

Al director de mi proyecto, el Dr. Simon Pickin, por guiarme en este proyecto y contribuir al aprendizaje de nuevos conocimientos.

Resumen

La tecnología empleada en las redes P2P constituye una manera efectiva a la hora compartir información entre usuarios en internet, por lo que actualmente es utilizada por grandes empresas.Con la ausencia de un servidor central es más sencillo evitar un colapso, cada usuario comparte información directamente con el resto de usuarios. Este tipo de redes se han visto relacionadas con la piratería, por una capacidad de intercambio de archivos que genera facilidad a la hora de distribuir copias, principalmente de películas y canciones. A menudo surgen ideas o proyectos de carácter social que pretenden emplear tecnología cliente-servidor clásica para llevar a cabo su desarrollo, frustrándose finalmente debido a la cantidad de tiempo que hay que dedicarle o el mantenimiento de la aplicación.

Este proyecto profundiza en el estudio de aplicaciones que emplean esta tecnología para responder a sus necesidades, y posteriormente especificar, diseñar y desarrollar elementos tecnológicos importantes de gran utilidad para la tecnología P2P. Para ello se ha analizado la tecnología P2P y sus características, las aplicaciones existentes, y una vez investigado sobre las herramientas posibles y seleccionada la que más se adaptaba a nuestros objetivos, se ha llevado a cabo.

Palabras clave

- P2P
- Red descentralizada
- Grafo acíclico dirigido (DAG)
- Ontología dinámica

Abstract

The technology used in P2P networks is an effective way of sharing information between users on the Internet and is currently used by large companies. With the absence of a central server it is easier to avoid a collapse, each user shares information directly with the rest of the users. This type of network has been linked to piracy, due to its capacity for file sharing, which makes it easy to distribute copies, mainly of films and songs. Often ideas or projects of a social nature emerge that aim to use classic client-server technology to carry out their development, frustrating with the amount of time that needs the maintenance.

This project goes into the study of applications that use this technology to meet their needs, and then specifies, designs and develops important technological elements of great utility for the application. important technological elements of great utility for P2P technology. To this end, we have analysed P2P technology and its characteristics, the existing applications, and once we had researched the possible tools and selected the one that best suited our objectives, it has been carried out.

Keywords

- P2P
- Decentralized network
- Directed Acyclic Graph (DAG)
- Dynamic Ontology

Índice general

Capítulo 1

Introducción

1.1. Antecedentes y Visión de alto nivel del contexto de la propuesta

Las redes P2P (Peer-to-peer) dirigen y optimizan el uso del ancho de banda de cada usuario conectándolos entre ellos, con lo que el rendimiento supera muchas veces a los métodos centralizados tradicionales. A día de hoy, esta tecnología se emplea para compartir ficheros de audio, vídeo o software, y en Voz sobre Protocolo de Internet (VoIP : La señal de voz que se transmite en una comunicación entre dos móviles, viaja por internet empleando un protocolo IP) [1], con la finalidad de que la transmisión de datos en tiempo real sea más eficiente.

Hay muchas iniciativas sociales, empresas sociales, servicios espontáneos sociales, etc... que necesitan o podrían funcionar mucho mejor con un componente informático. Regularmente surgen iniciativas de esta naturaleza con componente informático lanzadas por voluntariado entusiasta con buenas ideas y buenas intenciones. Sin embargo, la mayoría de estas iniciativas terminan fracasando, o ni siquiera llegan a lanzarse, por el compromiso de recursos financieros y sobre todo humanos que implican. Al prinicipio los voluntarios y voluntarias están dispuestos a invertir mucho tiempo y esfuerzo en sacar adelante la iniciativa, pero terminan cansándose de la inversión de tiempo permanente que implican la administración del sistema informático y la búsqueda de fondos para pagar su infraestructura.

Como ejemplo, se pueden citar los bancos de tiempo que surgen regularmente, sobre todo en tiempos de crisis cuando el dinero escasea. Muchos de ellos duran pocos años antes de desaparecer por las razones citadas. Por otro lado, hay iniciativas sociales que logran perdurar gracias, en parte, a las soluciones informáticas rudimentarias que eligen, pero que en consecuencia tienen un alcance y un impacto muy limitados.

1.2. Motivación

La motivación principal de este proyecto se basa en obtener una idea para una aplicación que mejore el bienestar social y sacar el lado humano de las personas. Por otro lado, la idea de profundizar en una tecnología como P2P es llamativa, e intentar aportar al avance de la tecno-

logía P2P en general, puesto que su uso no suele generar interés a empresas porque no es fácil monetizar.

1.3. Objetivos

La tésis de esta línea de TFG es que la tecnología P2P promete la solución a este dilema ya que, por su naturaleza, la infraestructura y la administración de las aplicaciones están distribuidas entre todos los usuarios y usuarias (a costa, innegablemente, de introducir problemas difíciles de seguridad y de confianza). Por tanto, no exigen al voluntariado un gran esfuerzo permanente. Cabe destacar que, al contrario de muchos trabajos en esta área, el enfoque de esta línea no es el de dar protagonismo a la tecnología e intentar buscar sitios donde se pueden aplicar tecnologías P2P prometadoras tales como blockchain, sino de dar protagonismo a las aplicaciones posibles.

El enfoque es el de estudiar bien las aplicaciones existentes o propuestas con el fin de dilucidar sus necesidades informáticas, definir y elegir la tecnología P2P más adecuada para contestar a estas necesidades y, finalmente, construir prototipos de las aplicaciones con las tecnologías definidas y elegidas. Con este enfoque, en la línea de lo que tradicionalmente se ha llamado tecnología apropiada, se busca evitar la restricción de perspectiva descrita en la frase de sabiduria atribuida a Abraham Maslow, “si solo tienes un martillo, todo te parece un clavo”[2].

Pese a que la tecnología procedente del mundo P2P avanza empleándose en sistemas privados de grandes empresas como Facebook (Cassandra) [3] y LinkedIn (Voldemort) [4], en aplicaciones P2P como BitTorrent [5] y Bitcoin [6] esta tecnología aún no está tan desarrollada como para establecer una plataforma distribuida de propósito general sobre la que construir una gran variedad de aplicaciones. El proyecto implicará la especificación de la aplicación y, a modo de demostración de factibilidad, el diseño, la implementación y la validación de una parte de esta especificación sobre una plataforma P2P existente.

1.4. Plan de trabajo

El trabajo comienza con una puesta en común de la propuesta sobre las redes P2P que se pretende llevar a cabo, su frecuencia de uso, quiénes lo usan y los motivos de por qué se usan y por qué no. A continuación se explicarán 2 aplicaciones de dicho tipo con fines de utilidad social que actualmente están actualmente en uso, como son Freecycle [7], cuya idea es regalar objetos que no se utilizan a personas que lo necesitan, y StreetBank [8], similar pero añadiendo solicitudes si se necesita ayuda para realizar alguna tarea.

Posteriormente se hará un recorrido por la situación actual de esta tecnología, dando a conocer sus ventajas, desventajas, tipos, estructuras y seguridad. Seguido de esto se tratarán las ontologías, profundizando en las ontologías dinámicas, centradas en ofertas y demandas descentralizadas. Finalmente, de cara a la ejecución de la idea, se explicará la tecnología que se ha escogido para la implementación y sus motivos, seguido de la implementación y el resultado obtenido.

Capítulo 2

Introduction

2.1. Background Information and High-level View of the Proposal context

P2P (Peer-to-peer) networks direct and optimize the use of the bandwidth of each user by connecting them together, so the performance often exceeds traditional centralized methods. Today, this technology is used to share audio, video or software files, and in Voice over Internet Protocol (VoIP: The voice signal that is transmitted in a communication between two mobiles, travels over the internet using an IP protocol) [1] in order to make data transmission in real time more efficient.

There are many social initiatives like social enterprises or social services that need, or could work much better with a computer component. Regularly initiatives of this nature arise with a computer component launched by volunteers with good ideas and good intentions. However, the most of these initiatives end up failing, or are not even launched because of the commitment of financial and especially human resources that are involved. At the beginning the volunteers are willing to invest a lot of time and effort in carrying out the initiative, but they end up getting tired of the permanent investment of time that the management of the computer system requires and the seek of funds to pay for the infrastructure.

For example, one can cite the time banks that arise regularly, especially in times of crisis when money is scarce. Many of them last few years before disappearing for the reasons cited. On the other hand, there is social initiatives that manage to last thanks, in part, to the rudimentary computing solutions that they choose, but which in consequently their scope and impact are very limited.

2.2. Motivation

The main motivation of this project is based on getting an idea for an application that improves social well-being and brings out the human side of people. On the other hand, the idea of delving into a technology such as P2P and trying to contribute to its advancement, in general, is striking since its use does not usually generate interest for companies because it is not easy to monetize.

2.3. Objectives

The thesis of this TFG is based in the idea that P2P technology promises the solution to this dilemma since by its nature, the infrastructure and the administration of the applications are distributed among all users (at the cost, undeniably, of introducing difficult security and trustworthy problems). So they do not require the volunteering a great permanent effort. It should be noted that, contrary to many jobs in this area, the approach of this line is not to give prominence to technology and try to find places where promising P2P technologies like blockchain can be applied, but to give prominence to possible applications.

The approach is to study either existing applications or proposals in order to elucidate your IT needs, define and choose the most appropriate P2P technology to answer these needs and, finally, build prototypes of the applications with defined and chosen technologies. With this approach, along the lines of what has traditionally been called appropriate technologies sought avoid the perspective restriction described in the wisdom phrase attributed to Abraham Maslow, "If you only have a hammer, everything looks like a nail to you" [2].

Although technology from the P2P world advances, being used in private systems of large companies such as Facebook (Cassandra) [3] and LinkedIn (Voldemort) [4]. In P2P applications such as BitTorrent [5] and Bitcoin [6], this technology is not yet developed as to establish a general-purpose distributed platform on building a wide variety of applications. The study of applications for which it would be advantageous to run on such a platform, and in particular the study of the requirements that these applications impose on it. The project will involve the specification of the application and, as a demonstration of feasibility, the design, implementation and validation of a part of this specification on an existing P2P platform.

2.4. Workplan

The work begins with a sharing of the proposal on the P2P networks to be carried out, their frequency of use, who uses it and the reasons why they are used, and why not. Next, 2 applications of this type and for social utility purposes that are currently in use will be explained, like Freecycle [7], whose idea is to give away objects that are not used to people who need them, and StreetBank [8], similar but adding requests if necessary, need help to perform a task.

Subsequently, a tour of the current situation of this technology will be made, revealing its advantages, disadvantages, types, structures and safety. Following this, ontologies will be discussed, delving into dynamic ontologies, focused on decentralized offers and demands. Finally, regarding the execution of the idea, the technology that has been chosen for the implementation and its reasons will be explained, followed by the implementation and the result obtained.

Capítulo 3

Freecycle y StreetBank

En este TFG, con el fin de anclar el análisis y a la vez ilustrar y hacer más comprensibles las propuestas, hemos elegido como caso de estudio la iniciativa social conocida como Freecycle [7], un ejemplo de una iniciativa que ha perdurado gracias, en parte, a su uso de soluciones informáticas rudimentarias. Nuestro cometido es proponer soluciones informáticas mejores a este problema y, de este modo, mejorar el alcance y el impacto de la aplicación.

Cada día, miles de productos como ropa, muebles o electrodomésticos que aún tienen utilidad, son descartados por sus propietarios, lo que conlleva un incremento de basura. Con el fin de disminuir este problema, surge el movimiento de Freecycle, que consiste en donar los productos u objetos que son descartados pero que aún tienen vida útil, con la finalidad de reducir el número de desechos. Esta idea surgió en 2003 en Tucson, Arizona, de un emprendedor llamado Deron Beal, que trabaja en una organización de recolecta de equipamiento de oficina usado, para su posterior venta. Pensó que se podía crear un sistema de reciclaje de mobiliario para reducir la cantidad de objetos que acaban en la basura, así que comenzó a mandar emails a sus contactos, mostrándoles objetos que les pudiesen ser de utilidad, y de este modo se formó un grupo en el que empezaron a intercambiar objetos.

A día de hoy, la idea de Freecycle se ha propagado por más de 110 países, con miles de grupos, consiguiendo mantener miles de toneladas fuera de los vertederos. Gracias a la extensión que está teniendo, también ha sido de gran ayuda para facilitar objetos a víctimas de desastres naturales.

La tecnología que usan actualmente está compuesta por grupos de mail y un servidor básico. Si la cuenta de correo de un usuario rebota los mensajes de Freecycle, el servidor retrocede en las entregas para no colapsar con direcciones de correo incorrectas.

Sus ingresos provienen de diferentes fuentes: Socios suscritos, ingresos por publicidad y donaciones particulares. Al ser una entidad sin ánimo de lucro, destina en torno al 90 por ciento de su recaudación a cubrir gastos, y el resto se divide entre los miembros, recaudación de fondos y cubrir gastos administrativos.

La necesidad de Freecycle de precisar un apoyo en forma de donaciones económicas debido a sus bajos ingresos nos obligan a explorar la noción de una ontología distribuída capaz de ser sostenible. Las ontologías distribuidas serán compartidas por muchas otras aplicaciones P2P de este tipo, como StreetBank [8], ejemplo del que se habla a continuación y que se ha mencionado en un trabajo de fin de grado que trata acerca de un banco de tiempo [9].

StreetBank [8] es una red con base en Reino Unido cuya finalidad es que los usuarios compartan sus habilidades de forma local, con sus vecinos. No se consigue un beneficio económico, sino que está motivado por el bien común y hacer del barrio un lugar más agradable. Se puede prestar o regalar cualquier cosa con la que se crea que se puede ayudar a algún vecino, y de la misma forma, el usuario también puede hacer solicitudes si necesita ayuda, como por ejemplo mover un piano, o ayudar a cortar el césped. Fundada en 2010, su fundador Sam Stephens tuvo la idea al ver que necesitaba que su vecino le prestase un par de cortadoras de setos.

Cuantas más cosas se comparten más beneficioso resulta para la comunidad.

Figura 3.1: Streetbank

Capítulo 4

Estado del arte

En esta parte se realizará un resumen de la situación actual de esta tecnología, junto con una descripción de las redes P2P respecto a su arquitectura, su clasificación según su estructura y su seguridad.

4.1. Las redes P2P

Una red P2P (Peer-to-peer) es en la actualidad una de las formas más importantes y de compartir todo tipo de material entre usuarios de internet, sin importar la plataforma de software utilizada ni el lugar o momento en que se encuentren. Consiste en una red de computadores que funciona sin necesidad de contar con clientes ni servidores fijos, por lo que le otorga una gran flexibilidad. Esto se obtiene gracias a que la red trabaja en forma de una serie de nodos que se comportan como iguales entre sí. En pocas palabras significa que los computadores están conectados a la red P2P actual al mismo tiempo como clientes y servidores respecto al resto de computadores conectadas.

4.1.1. Ventajas de una red P2P

- **Escalabilidad**: Cuanto más recursos tenga la red a su disposición, el acceso que tendrán a estos nodos será mejor, y por tanto, cuanto más grande sea la red, mejor funcionará, ya que los recursos de los nuevos nodos se comparten con cada nueva conexión. Los recursos de los nuevos nodos también son compartidos y crece el número de recursos.
- **Distribución de costes**: Al no ser centralizado, los costes se comparten entre todos los nodos de la red, por ejemplo el de ancho de banda o almacenamiento, junto con la administración del sistema.
- **Robustez**: Estos sistemas ofrecen un nivel alto de robustez puesto que responde bien a los fallos de accesos a información debido a su réplica entre los nodos de la red. Esto hace que se optimice el acceso a la información.

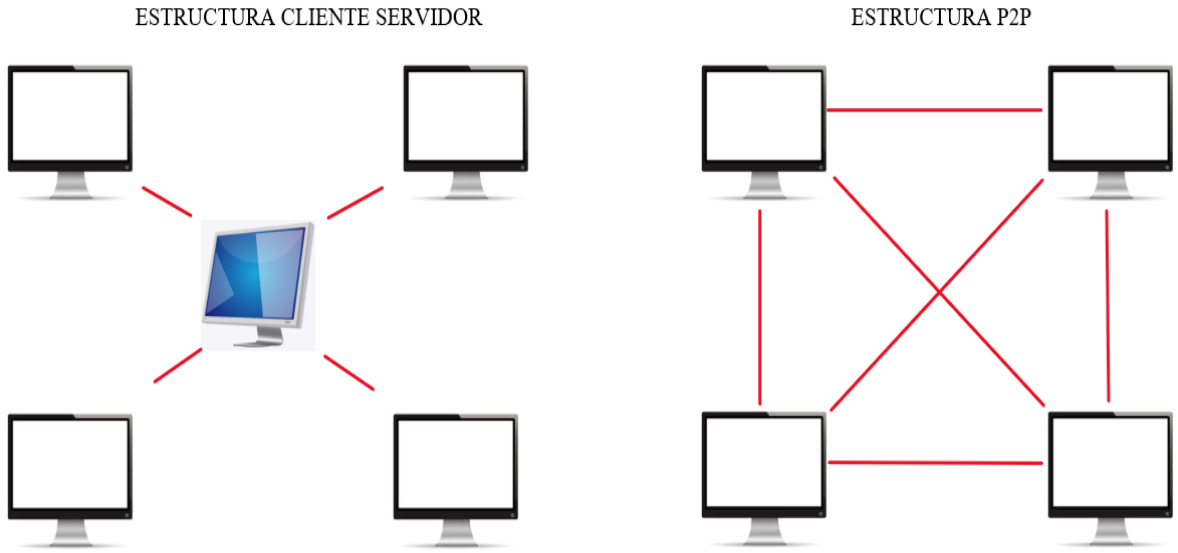

Figura 4.1: RedesP2P

4.1.2. Inconvenientes de una red P2P

Los inconvenientes de este tipo de sistemas se producen en el marco de la administración y en todas las tareas que de ella dependen. Como la gestión de la administración se lleva a cabo entre todos los nodos, es difícil saber quién controla los recursos de la red y de qué forma.

Por otro lado, al no existir una unidad central de control, las medidas de seguridad ya no son globales, por lo que cada nodo debe proteger sus datos. La realización de réplicas de los datos o copias de seguridad vuelve a caer en los nodos de la red.

En el fenómeno conocido como “churn”se produce una conexión y desconexión de un gran número de nodos.

4.2. Arquitecturas P2P

Aunque todas las redes P2P tienen en común la ausencia de un nodo central, se pueden clasificarlas en tres tipos según su grado de centralización:

4.2.1. Redes P2P centralizadas

En este tipo de red todas las transacciones se realizan mediante un solo servidor que se emplea de punto de enlace entre dos nodos, y que de forma paralela guarda y reparte los nodos donde se guardan los contenidos. Todas las comunicaciones (peticiones y encaminamiento entre nodos) necesitan el servidor. Este tipo de redes tienen problemas en puntos únicos de fallo, además de un alto coste de mantenimiento, gran consumo de ancho de banda y condiciones legales. Como ejemplo de estas redes destacan Napster [10] y Audiogalaxy [11].

4.2.2. Redes P2P descentralizadas

También conocidas como redes P2P "puras", es el tipo de red más común. No necesita que un nodo central posea el control de la red, sino que los propios nodos guardan los contenidos, por lo que las comunicaciones se producen entre usuarios (nodos), junto con la ayuda de otro usuario que sirve de enlace, y de esta forma los nodos tienen función de cliente y de servidor. Como ejemplo de estas redes destacan Kademlia [12] y Ares Galaxy [13].

4.2.3. Redes P2P híbridas

En este tipo de red se aprecia la interacción entre un servidor usado a modo de Hub, y lleva a cabo tareas como la gestión de recursos de ancho de banda, enrutamiento y comunicación entre nodos. En esta comunicación se desconoce la identidad de cada nodo y no se almacena información, de este modo el servidor no distribuye archivos a ningún nodo. Puede tener más servidores que administren los recursos compartidos. Generalmente posee un servidor central que almacena la información en espera y contesta a peticiones de esa información. Los nodos son los encargados de almacenar la información, lo que permite al servidor central explorar los recursos que se quieren compartir y poder descargar los recursos compartidos a los nodos que lo solicitan. Como ejemplo de estas redes destaca BitTorrent [5].

4.3. Clasificación según la estructura

Es importante explicar que la red de sobrecapa del P2P se basa en todos los peer que colaboran como nodos de red. En función de cómo los nodos de la red de sobrecapa se conectan entre ellos, se pueden dividir como redes P2P estructuradas o no estructuradas

4.3.1. Redes P2P no estructuradas

Estas redes se generan cuando los enlaces de la sobrecapa se crean de forma arbitraria, por ejemplo ocurre cuando un peer que desea unirse a una red es capaz de copiar enlaces ya existentes en otro nodo, y en un plazo de tiempo crear sus propios enlaces. Si un peer desea acceder a unos datos concretos de la red, la petición tiene que recorrer toda la red para encontrar tantos peers como sea posible, para encontrar a alguien que comparta los datos.

El inconveniente de estas redes es que las peticiones no siempre se pueden resolver. Resulta sencillo si se busca información popular que esté en varios peers, pero para datos no tan populares que únicamente comparten unos peers existe una alta probabilidad de no encontrarlos. Como ejemplo de estas redes se encuentra KaZaA[14].

4.3.2. Redes P2P estructuradas

Estas redes superan las limitaciones de las redes no estructuradas, conservando una tabla hash distribuida (DHT) como solución más común aunque no la única, y dejando que cada peer

se encargue de una parte específica del contenido de la red. Se emplean funciones de hash distribuido y se asignan valores a cada peer en la red y a cada contenido. A continuación estas redes siguen un protocolo global con el que se determina qué peer se encarga de qué contenido, y de este modo, cuando un peer necesite buscar datos concretos, usa el protocolo global para ver quién es el peer o los peers responsables de esos datos, y a partir de ahí realiza la búsqueda de éstos.

- **Sistemas basados en árboles**: Cuya finalidad es alojar información mediante el uso de árboles. La idea consiste en crear una jerarquía de nodos en los que los de la capa superior poseen más estabilidad y eficiencia, y los de la capa inferior menos estables y eficientes.
- **Sistemas basados en listas de saltos (skip lists)**: Se basan en una lista doblemente enlazada de ordenamiento múltiple, compuesto por varias listas diferenciadas por niveles y nodos que participan en estas. La información está ordenada y fraccionada en rangos de valores en las listas.
- **Sistemas basados en tablas de hash distribuidas**: La forma de almacenamiento y consulta de los recursos es semejante al de una tabla hash. Cada nodo es responsable de un rango de valores y a cada unidad de datos se establece un único valor adquirido mediante una función de hash uniforme, generalmente SHA-1. Actualmente es la estructura de redes que más se utiliza, debido a su nivel de eficiencia en las consultas.

4.3.3. Redes P2P híbridas

La idea de estas redes consiste en crear una jerarquía de nodos. Existen unos nodos denominados supernodos que establecen entre ellos una subred P2P. Cada nodo restante pertenece a un supernodo y sólo establece conexiones con otro nodo si este pertenece al mismo supernodo.

La ventaja de las redes desestructuradas es su capacidad a nivel de nodo a la hora de encontrar vecinos y almacenar recursos, aunque su encaminamiento basado en inundación no es muy eficiente. Sin embargo las redes estructuradas requieren un elevado coste de mantenimiento, pero son capaces de encaminar mensajes de manera eficiente.

4.4. Seguridad en las redes P2P

A día de hoy, la seguridad en las redes se ve afectada por distintos tipos de ataques, dando lugar a consecuencias graves, y en algunos casos irreparables. Los ataques más comunes que se pueden apreciar son:

- **Ataques de envenenamiento (poisoning)**: Consisten en la inyección de datos falsos en la red P2P. Un ejemplo sería la modificación de la caché donde reside la dirección IP que está relacionada con una URL, por lo que el destino se reconduce al sitio web que decide el atacante [15].
- **Ataques de sybil**: [16] Se basa en la creación de múltiples cuentas en la red, todas ellas de la misma propiedad, con el fin de influir en aquellas determinaciones que se tomen en la red.

- **Ataques DDoS (denegación de servicio)**: La finalidad de estos ataques consiste en dejar un servidor inhabilitado, ya sea por tráfico masivo de datos para colapsar el ancho de banda del servidor, o para acabar con los recursos del sistema [17].

4.4.1. Seguridad en los equipos

Teniendo en cuenta que en las redes P2P los usuarios son los nodos, existe un importante problema respecto al acceso desde el puerto que utiliza la herramienta, pues al abrir el propio puerto, este se abre al resto de usuarios de la aplicación, pudiendo llegar a acceder a la información del ordenador, como puede ser la dirección IP o la ubicación. De hecho, si no se ha configurado previamente el acceso a ciertas carpetas, se puede alcanzar el acceso a información sensible, como contraseñas o datos financieros.

También es necesario destacar un problema como es la posibilidad de que haya software malicioso, o ataques por denegación de servicio, lo que es más sencillo por la estructura de estas redes. Además, para una total seguridad, algunas veces no basta con desconectarse de la red, pues algunas conexiones maliciosas pueden seguir activas. Para conseguir la máxima seguridad al utilizar este tipo de aplicaciones, es conveniente:

- Almacenar solamente los archivos necesarios, realizar una configuración completa.
- No utilizar los servicios de mensajería de estas aplicaciones, ni proporcionar datos reales.
- Analizar todos los archivos descargados, y emplear herramientas para cifrar datos.
- Reiniciar el ordenador cuando se cierra el programa, para que no se queden vías de acceso abiertas.
- Configurar también los dispositivos que estén conectados a la red mediante un firewall, para evitar que sean accesibles.

4.4.2. Seguridad en los datos

Existen varios mecanismos para llevar a cabo la seguridad de los datos que se utilizan en este tipo de aplicaciones: Métodos de criptografía, firma digital y certificados digitales.

- **Métodos de criptografía** : Entre los que destacan diferentes tipos:

  Criptografía simétrica o de clave secreta : En este método se emplea la misma clave para cifrar y descifrar la información. La desventaja que tiene es que es relativamente fácil averiguar la clave.

  Criptografía simétrica o de clave secreta : Se emplea una clave pública para los que quieran compartir información privada con el propietario, y una clave privada para desencriptar la información.

- **Firma digital** : Este método consiste en el cifrado de datos del mensaje, junto con la identificación de la persona que lo ha creado previamente. El contenido se cifra a través

de una función hash que calcula un número de datos a firmar, y se encripta mediante clave pública. La finalidad de este método es que el receptor se asegure de la identidad del usuario que ha generado el mensaje que ha recibido.

- **Certificados digitales**: Son documentos de verificación y validación de la relación de un individuo o entidad y una clave pública, por lo que confirman que una clave pública concreta está asociada a un individuo concreto. Entre sus campos, destaca el número de serie, la autoridad que lo certifica o la fecha de expiración entre otros.

Capítulo 5

Ontologías

La ontología es una rama de la metafísica que trata el estudio de la naturaleza y las relaciones del ser. En el campo de la información y la computación, se basa en una descripción de conceptos en un dominio de discurso, las propiedades que poseen y las restricciones impuestas sobre las propiedades. En este campo se trata directamente la clasificación de conceptos fijos según sus relaciones con el resto de conceptos.

Hoy en día existen millones de ontologías en uso en todas partes. Unos ejemplos claros de estas ontologías son Wordnet (una base de datos léxica de palabras inglesas que engloba palabras en conjuntos de sinónimos, aportando definiciones generales y acumulando relaciones semánticas entre conjuntos de sinónimos) [18], y Cyc (su intención es ensamblar una ontología comprensiva y una base de datos general para que las aplicaciones de inteligencia artificial sean capaces de llevar a cabo razonamientos del tipo humano) [19].

Este tipo de ontologías son realmente importantes de cara a la representación del conocimiento de un modo que sea compresible para las máquinas. Para construir una red es necesaria la combinación entre conceptos y relaciones.

Los elementos de ontologías pueden ser individuos, clases, atributos, relaciones, funciones, restricciones, axiomas y eventos. A partir de estos elementos se configuran ontologías. En las últimas décadas han surgido muchos lenguajes específicos para describir ontologías. Cuando se configura una ontología es necesario especificar previamente el dominio del conocimiento que va a representar respecto a la compatibilidad de la ontología. Las ontologías que se centran en un solo tema generan más limitaciones a la hora de su crecimiento y expansión a otros campos, mientras que si la ontología no es tan específica será aplicable en más campos pero con menos precisión.

5.1. Ontologías para aplicaciones de ofertas y demandas

Para que las aplicaciones sean realmente utilizables, tanto en Freecycle como en otras aplicaciones de gestión de ofertas y demandas, es necesario clasificar los artículos o servicios ofertados o demandados en categorías. De lo contrario el demandante/ofertante recibe muchas ofertas/-demandas sin interés, y no podría pedirle al sistema que éste le notifique si aparece una oferta que resulta de su interés.

Para poder clasificar los objetos ofertados y demandados en categorías, hace falta un modelo de los conceptos del dominio de interés y las relaciones entre ellas, es decir, una ontología. Se observa que la restricción de una ontología a una relación particular puede representarse en forma de grafo dirigido en el que los vértices representan los conceptos y las aristas representan la relación en cuestión entre estos conceptos. El interés de usar grafos es facilitar el procesamiento informático.

Puesto que nuestro interés principal en las aplicaciones de interés es poder hacer un "matching" inteligente entre ofertas y demandas, la relación más importante entre los conceptos del dominio será la de generalización (cuando un concepto es más general que otro). Por tanto, podemos restringir nuestra ontología a la relación de generalización, permitiendo de este modo su representación en forma de grafo dirigido cuyos vértices corresponden a los conceptos y cuyas artistas corresponden a la relación de generalización entre estos conceptos. Puesto que las aristas representan la relación de generalización, se ha de restringir el tipo de grafo que se va a usar, como mínimo, a los " DAG", los grafos dirigidos sin ciclos.

Debemos considerar otros posibles simplificaciones de la representación de la ontología así como posibles simplificaciones de la clasificación de los objetos (clasificar es asignar cada objeto a un concepto de la ontología, como instancia de este concepto). Como ejemplo del primero, debemos considerar las implicaciones para la aplicación restringir los DAG a DAG con una única raíz o incluso árboles. Como ejemplo del segundo, debemos considerar las implicaciones para la aplicación restringir la clasificación de los objetos a los vertices hoja, es decir, los vértices que representan conceptos que no son generalizaciones de ningún otro concepto representado.

En la implementación tendremos que identificar de algún modo los conceptos de la ontología. Puesto que la ontología estará expuesto al usuario, no será solo una estructura interna del programa, lo lógico sería identificar los conceptos con nombres, es decir, con palabras en lenguaje natural o, más generalmente, con conjuntos de sinónimos. Puesto que en la representación gráfica de ontologías propuesta, las aristas representarán generalización y los vértices estarán etiquetados con palabras de lenguaje natural que describen, o etiqueten los conceptos, habrá que asegurar la coherencia entre la relación de generalización entre conceptos modelado por las aristas y la relación de hiponimia e hiperonimia entre las palabras utilizadas para etiquetar estos conceptos, teniendo en cuenta, además, los sinónimos, en el caso de etiquetar los conceptos con conjuntos de sinónimos. Se podría usar Wordnet con este fin (al menos en inglés).

5.2. Ontologías dinámicas para aplicaciones de ofertas y demandas

En el tipo de aplicación que estamos considerando, la ontología tiene que ser dinámica, es decir, que puede cambiar, ya que en el domino que la ontología modela para las aplicaciones de interés, podrán surgir nuevos conceptos, relaciones, restricciones, etc...

Otras aplicaciones como por ejemplo eBay, emplean un árbol de conceptos en el que todos los conceptos están clasificados únicamente mediante las hojas del grafo de conceptos. El problema aparece cuando se quieren crear conceptos nuevos más específicos, un nodo que antes era hoja ya no lo será, y por tanto cada vez que se introducen conceptos más específicos en el árbol de conceptos, se tienen que reclasificar todos los objetos que estaban clasificados con aquella hoja

que ha dejado de ser hoja.

Se observa que, al añadir nuevos conceptos o nuevas relaciones de generalización entre conceptos, la aplicación debería comprobar la coherencia de las nuevas aristas con las relaciones de hiponimia e hiperonimia entre las palabras utilizadas para etiquetar los conceptos conectados por estas aristas, teniendo en cuenta los sinónimos, en su caso.

5.2.1. Ontologías dinámicas para aplicaciones de ofertas y demandas descentralizadas

Como ya se ha explicado en otros capítulos, nuestro interés se sitúa en una aplicación descentralizada. Para que la aplicación sea sin ánimo de lucro y sostenible en el tiempo, no puede haber un servidor central y la administración de la aplicación ha de repartirse entre todos los participantes. Este hecho tiene implicaciones profundas para una aplicación de gestión de ofertas y demandas.

Gestión de la comunicación entre ofertantes y demandantes

Será necesario averiguar cómo en un contexto descentralizado se puede gestionar la comunicación entre ofertantes y demandantes, es decir, ¿cómo puede el demandante buscar ofertas de interés? y ¿cómo puede el ofertante enviar notificaciones a los demandantes que previamente han registrado un interés en este tipo de ofertas? En ambos casos estará implicado un algoritmo de "matching" del que hablaremos en otro punto. Respecto a la comunicación en sí, se debe explorar cuál de los distintos mecanismos de las redes P2P es el más adecuado: Red estructurada / no estructurada, red no estructurada con/sin supernodos...

Si cada nodo de la red puede modificar su propio modelo del dominio, la ontología utilizada para clasificar los objetos será distinta en distintos nodos de la red. En estas circunstancias ¿cómo puede funcionar la comunicación entre distintos nodos de la red, si no comparten la misma visión del mundo? Surge la necesidad de una operación de "merge" de ontologías, o de partes de ontologías, precisamente para poner en común la visión del mundo de dos nodos de red distintos dando lugar a una ontología consensuada.

Cualquier algoritmo de merge tendría que asegurar que el resultado del merge es un grafo dirigido del tipo que se está usando. En el caso de usar DAGs, el algoritmo de merge tendría que asegurar que el resultado del merge no tiene ciclos. Si hubiera que comprobar que un grafo entero tiene la propiedad, este proceso tendría que emplear una solución al clásico problema de la teoría de grafos conocido como el problema del conjunto de vértices/aristas de realimentación (the vertex/edge feedback set problem) [20]. Sin embargo, sería más sencillo si sólo hubiera que comprobar que para cada arista nueva que se añade al grafo se conserva las propiedades de interés. El hecho de que este problema está demostrado ser NP-completo no debería ser un problema en este caso por el tamaño reducido de los grafos que utilizaremos.

Una posible solución al problema de la visión del mundo distinto en cada nodo de la red es que los nodos de la red estén permanentemente intentando llegar a una ontología consensuada mediante un mecanismo de envío regular de mensajes que contienen la ontología del emisor y, en cada nodo de la red, el "merge"de las ontologías recibidos por mensajes de este tipo con

la suya propia. Habría que estudiar las condiciones en las que esta solución fuese viable, es decir, que no se envíen tantos mensajes que saturen a la red y que no haya tantos cambios en la ontología que la red nunca llegue a una ontología consensuada lo suficientemente estable.

Supongamos el caso del nodo Electrodomésticos, cuyos hijos son frigoríficos, hornos, lavadora, lavaplatos y microondas. Se quiere incorporar Lavavajillas(a) al DAG. Habría que comprobar su relación de hiponimia e hiperonimia con el resto de nodos de la red. Como lavavajillas es sinónimo de lavaplatos, no se crea un nodo nuevo.

if esSinónimo(a,DAG)

Se emplea el nodo que ya estaba.

else if esHipónimo(a,DAG)

Se añade a la red al mismo nivel que su hipónimo.

else if esHiperónimo(a,DAG)

if masGenerico(a,DAG) ->Se añade como padre si es más genérico que el que se compara.

else ->Se añade como hijo si es menos genérico que el que se compara.

En caso de querer hacer "merge"de ontologías identificados por palabras de lenguaje natural, el etiquetado de los conceptos por conjuntos de sinónimos será obligado; no será posible identificarlos mediante un único término ya que no se puede garantizar que distintos nodos de la red utilicen el mismo término para un concepto dado.

En nuestro caso de conceptos etiquetados por palabras o por conjuntos de sinónimos, un buen algoritmo de "merge" debería contar con la relación de hiponimia e hiperonimia entre las palabras utilizadas para etiquetar los conceptos de las dos ontologías. En caso de la existencia de distintas ontologías en distintos nodos de la red, parece ser poco práctico restringir la clasificación de los objetos a vértices hoja del grafo que representan la ontología. Si no, sería muy difícil evitar pedir muy a menudo la intervención del usuario para reclasificar los objetos asignados a un vértice que, después de un merge, deja de tener la propiedad de ser hoja.

Matching de ofertas y demandas

En el caso de las aplicaciones de oferta y demanda, ¿cómo se puede implementar el "matching" de ofertas y demandas cuando el ofertante y el demandante no tengan la misma visión del mundo, es decir, no clasifiquen los objetos mediante la misma ontología?

En el contexto del matching entre ofertas y demandas, previo al uso de un algoritmo de matching que se podría utilizar en un contexto centralizado, habría que poner en común la visión del mundo del nodo de la red demandante y el nodo de la red ofertante. Se puede observar que para decidir si existe un "match", no hace falta hacer un merge completo de las ontologías del demandante y del ofertante. Bastaría con hacer un merge de la parte de la ontología del demandante implicada en la clasificación del objeto demandado y la ontología del ofertante.

En consecuencia, el demandante debería enviar esta parte de su ontología en todos sus mensaje de búsqueda y también en sus registros de interés para recibir notificaciones. Respecto a los

últimos, si el demandante cambia su ontología, tendría que rehacer los registros de interés que conciernen a la parte de la ontología modificada. Surge la pregunta ¿exactamente cuál es la parte de su ontología de interés en una demanda?

Gestión de la ontología consensuada

Para que las ontologías consensuadas no crezcan sin fin, incorporando conceptos sin mucha utilidad, se puede emplear el mecanismo de leasing ampliamente usado en sistemas distribuidos en general. En nuestro caso, se puede asignar un TTL (Time To Live) a cada concepto que se añade a la ontología. Regularmente, cada nodo de la red tendría que decrecer el TTL de todos los elementos de su ontología y cuando el TTL de un concepto llega a 0 tendría que eliminarlo.

El TTL de un concepto se resetearía cada vez que este concepto se usa. Habría que definir exactamente qué quiere decir "usar" (en una búsqueda, en una oferta, en un registro de interés, ¿en la búsqueda/oferta/registro de un objeto clasificado en un vertice hijo del grafo?,...). Si en la representación gráfica utilizada para las ontologías un vertice puede tener múltiples vértices padre (es decir, no es un árbol), habrá que considerar el uso de TTLs también en las aristas.

Habría que estudiar si una parte del grafo (en caso de usar DAGs con una única raíz, seguramente sería una subgrafo simplemente conexo que incluye la raíz) debería tener un TTL infinito o, al menos, muy grande. Simplificaría la implementación de los algoritmos de merge y de match y facilitaría la estabilidad de una ontología consensuada.

También habría que asegurar que nunca se suprime un vértice si el efecto es de partición del grafo (da igual que haya expirado su TTL), y estudiar el papel que puede/debe jugar el TTL en los algoritmos de "merge" y de "match". Por ejemplo, en la parte del algoritmo de "merge" que elimina aristas o vértices con el fin de evitar ciclos, se podría priorizar la eliminación de las aristas o los vértices con el TTL más bajo. Por último, habría que estudiar el valor más adecuado del TTL por defecto, que seguramente sería función del tamaño de la red. Existen distintos algoritmos para estimar el tamaño de una red P2P.

Capítulo 6

Elección de tecnologías

A la hora de seleccionar las herramientas y formatos para implementar el algoritmo de grafos es conveniente tener en cuenta el fin para el que está diseñado dicha herramienta y la cantidad de utilidades que ésta nos puede proporcionar. Otro factor a tener en cuenta es la familiaridad que se tenga con esa herramienta o con el lenguaje a utilizar.

Las herramientas y librerías analizadas para el algoritmo de grafos han sido las siguientes:

- **GraphX** : Hecho sobre el núcleo de Spark [21], es un motor de búsqueda cuya función es el análisis de grafos para conseguir información de los datos estructurados. Para realizar el cálculo de grafos, contiene un conjunto de operadores fundamentales. La API de GraphX no se encuentra en Python.
- **GraphLab** : Framework inicialmente desarrollado para actividades de machine learning, basado en grafos. Escrito en C++, destaca por ofrecer a las aplicaciones modelos de aprendizaje automático.
- **Giraph** : Perteneciente a Apache, es un sistema de procesamiento de grafos que resulta útil cuando se trabaja con una cantidad inmensa de datos estructurados. Para procesar los grafos, Giraph [22] emplea MapReduce de Apache Hadoop.

Las herramientas mencionadas anteriormente han sido descartadas para el proyecto, puesto que las tres surgen de las necesidades de big data para tratar grafos inmensos, y no es la necesidad surgida en este trabajo.

- **GraphTool** : Perteneciente a Python, es una herramienta usada para el análisis y el tratamiento de grafos. Destaca por la capacidad de crear grafos dirigidos y no dirigidos, y la asociación entre aristas o vértices. Tiene el algoritmo del minimum-spanning-tree, empleado para eliminar aristas, que está relacionado con el algoritmo del feedback. Graph-tool [23] trabaja con los formatos graphml, dot, gml (y el suyo, gt).
- **JGraphT** : Software de dibujo para grafos, capaz de usarse en Java, JavaScript y C. JGraphT [24] Resulta interesante tanto por la cantidad de módulos que tiene, como por sus algoritmos de enumeración de ciclos simples, el cálculo de la base de un ciclo no dirigido y el cálculo de los ciclos eulerianos.

- **NetworkX** : Paquete de Python con la capacidad de crear, manipular y utilizar gran cantidad de funciones de redes complejas y grafos.

Tras analizar una por una las ventajas y desventajas de cada herramienta se decidió utilizar GraphTool o NetworkX [25], influido por emplearse en Python, un lenguaje que ofrece más experimentación que java, y era una buena oportunidad para aprenderlo Finalmente se escogió Networkx al tener una gran variedad de funciones que han servido en la implementación.

Capítulo 7

Diseño y especificación de la red

Aclarando la idea principal del proyecto, este consiste en estudiar la viabilidad de una aplicación implementada en una estructura descentralizada, cuya finalidad posterior sería su utilidad a nivel social. Una vez elegida la tecnología que vamos a utilizar, comienzan a definirse las clases necesarias.

En este punto es importante definir los conceptos de hiponimia e hiperonimia:

- **Hiperonimia**: Una relación de hiperonimia entre dos palabras surge cuando una engloba un concepto más general que la otra.
- **Hiponimia**: Una relación de hiponimia entre 2 palabras surge cuando ambas comparten un hiperónimo.

En este ejemplo, las hojas tienen relación de hiperonimia con la raíz y de hiponimia entre ellas.

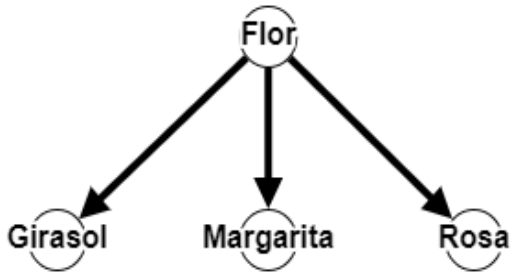

Figura 7.1: Ejemplo de hiperonimia e hiponimia

La estructura de clases que necesitaremos es la siguiente:

- Una clase en la que se construyan los nodos de la red. Se necesitarán al menos 2 atributos, uno es una lista con los sinónimos, y otro que contenga el nombre de un synset (conjunto de sinónimos) de Wordnet.

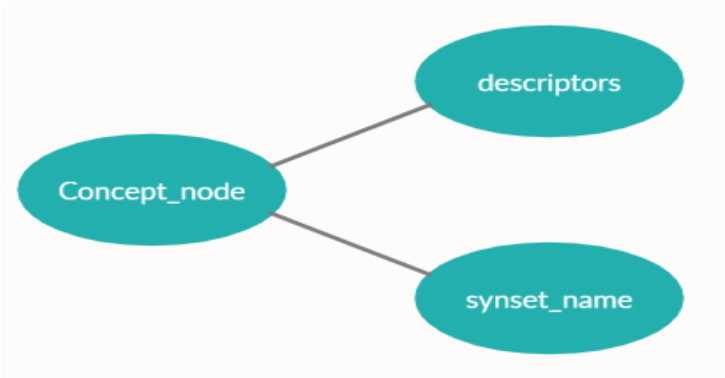

Figura 7.2: Estructura de la clase Concept node

- Una clase en la que se construyan las aristas de la red. Cada arista deberá tener 3 atributos, que son la información de ésta, el nodo en el que se origina y el nodo destino.

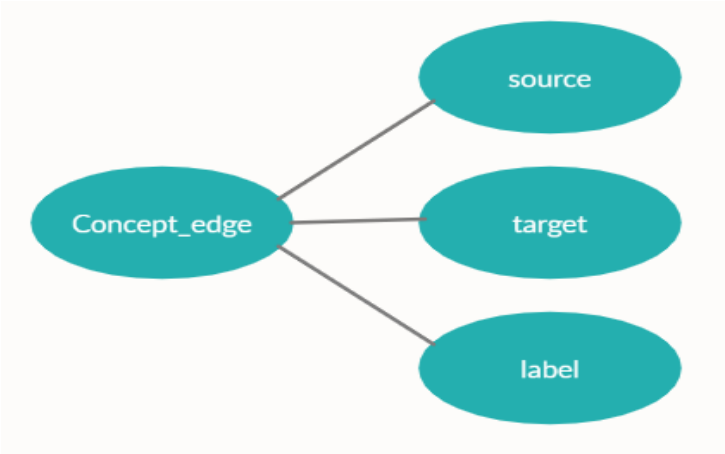

Figura 7.3: Estructura de la clase Concept edge

- Una clase que construya la red añadiendo los nodos y aristas necesarios.

Respecto a la estructura más adecuada, tratada en el trabajo de fin de grado titulado “Banco de tiempo con oferta y demanda de servicios en una red P2P”[9] se va organizar un DAG con una sola la raíz, el término más general, mientras que los términos se van especificando a medida que se recorre.

En primer lugar, respecto los nodos habrá que tener una función que compruebe lo siguiente:

- Suponiendo que siempre existe un synset en Wordnet para cada sustantivo que podemos querer usar en la aplicación, se deberán etiquetar los nodos con un set() de sustantivos junto con un campo “synset”.Todos los sinónimos de un nodo deberán estar en el synset de Wordnet identificado en el otro atributo del nodo.
- Las aristas responderán a una relación de hiponimia o hiperonimia entre los synsets de los nodos.

- Cada nodo del grafo tiene asociado el nombre de un synset de Wordnet distinto (como valor de un atributo), y un sustantivo puede estar, como máximo, en el conjunto de sinónimos de un solo nodo del grafo, siendo ese nodo el que tiene asociado el nombre de un synset al que pertenece este conjunto de sinónimos.

El caso de un nodo etiquetado por un set de sustantivos de los cuales uno está en un synset de Wordnet y otro en otro synset de Wordnet no debería poder surgir nunca.

En segundo lugar, cuando se pretende añadir un nuevo nodo con un sustantivo nuevo:

- Si el sustantivo nuevo pertenece a un synset de Wordnet cuyo nombre es el atributo de "name"de alguno de los nodos del grafo:

  Si el sustantivo nuevo no está en el conjunto de sinónimos de el nodo que tiene ese atributo "name", se añade el sustantivo nuevo al conjunto de sinónimos de ese nodo. De lo contrario no se realiza ninguna acción.

- Si el sustantivo nuevo no pertenece a ningún synset de Wordnet cuyo nombre es el atributo de "name"de alguno de los nodos del grafo, se crea un nuevo nodo. Al crear un nuevo nodo, el conjunto de sinónimos tendrá un sólo elemento, el sustantivo nuevo, mientras que en el caso del "synset name"si el sustantivo pertenece a un solo synset de Wordnet, será ese el "synset name". Si el sustantivo pertenece a varios synsets de Wordnet, el usuario tendrá que elegir el más adecuado de todos.

Es interesante contar con un algoritmo de merge para poder añadir nuevos nodos y aristas en la parte correspondiente del árbol, teniendo en cuenta los atributos del nodo, sus hiperónimos e hipónimos.En algunos casos será necesario eliminar aristas para quitar ciclos.

Supongamos el caso del nodo Electrodomésticos, cuyos hijos son frigoríficos, hornos, lavadora, lavaplatos y microondas. Se quiere incorporar Lavavajillas(a) al DAG. Habría que comprobar su relación de hiponimia e hiperonimia con el resto de nodos de la red. Como lavavajillas es sinónimo de lavaplatos, no se crea un nodo nuevo.

```
if esSinónimo(a,DAG): Se emplea el nodo que ya estaba.
else if esHipónimo(a,DAG): se añade a la red al mismo nivel que su hipónimo.
else if esHiperónimo(a,DAG):
    if masGenerico(a,DAG): se añade como padre.
    else: se añade como hijo.
```

También nos interesa poder serializar y deserializar la información del DAG para facilitar el envío por la red y, quizás también, el almacenamiencto en fichero, por lo que habrá que elegir un lenguaje que almacene estructuras de datos simples, siendo legible para el usuario.

En 2 ocasiones los pares tendrán que enviar grafos de este tipo a otros pares por la red:

- Enviar su grafo de conceptos a sus vecinos regularmente para llegar a un grafo de conceptos consensuado entre ellos.
- Enviar una parte de su grafo de conceptos cada vez que hace una búsqueda.

Capítulo 8

Implementación de la red

La implementación está enfocada a la definición y manipulación de un tipo de datos para los grafos de conceptos que se utilizarán en una versión más adecuada de aplicaciones como Freecycle. Para ello se ha creado una clase Concept graph(), clase que proporciona una abstracción para una implementación de los grafos con NetworkX en Python. A continuación se explican las partes de este tipo. Debido a la cantidad de información que tiene que almacenar cada nodo no resulta posible realizarlo con nodos simples, por lo que se ha creado un nodo a medida mediante un objeto. Como se mencionaba anteriormente en la especificación, se ha creado una clase referente a los nodos, otra referente a las aristas y otra referente a la red o el grafo.

8.1. Implementación de nodos

Los nodos poseen 2 atributos:

- **Descriptors**: Lista que almacena los sinónimos de un synset.
- **Name**: El nombre que posee el synset de Wordnet que incluye esos sinónimos.

Para la construcción de un nodo, previamente se comprueba que tanto la información de descriptors como de Name es correcta. Dichas comprobaciones se llevan a cabo con 2 funciones auxiliares creadas para verificar que tanto los argumentos como el tipo de argumentos son los correctos (valid types y valid args).

Cuando se quiere añadir un nodo nuevo se comprueba que no sea sinónimo de ningún nodo existente, por lo que además de la función que añade un sinónimo a la lista de descriptors de un nodo, se ha creado una función para validar si esa palabra ya se encuentra o no en algún nodo. También está implementada una función que compara nodos, para evitar nuevos nodos innecesarios (eq).

Finalmente la función “contains” comprueba el tipo de elemento que se pasa por parámetro con el código que ahora está en (contains) de Concept graph, clase explicada más adelante. También comprueba que el elemento pasado por parámetro está dentro de los descriptores, o que es la misma cadena que el nombre de synset dependiendo de caso.

8.2. Implementación de aristas

Las aristas poseen 3 atributos:

- **Source**: Almacena el nodo origen de la arista.
- **Target**: Almacena el nodo destino de la arista.
- **Label**: Información contenida en la arista.

Antes de la construcción de una arista se comprueba que el tipo de argumentos sean válidos. También está implementada una función que compara aristas, para evitar aristas duplicadas, que actúa llamando a la función (eq) de la clase Concept node explicada previamente.

Finalmente la función "contains" comprueba el tipo del elemento de entrada:

- Si el elemento es un nodo comprueba que es igual al source o el target.
- Si el elemento es otra cosa comprueba que es igual al label.

8.3. Implementación del grafo

La clase Concept node tiene un atributo que contiene el grafo NetworkX correspondiente. Es posible almacenar directamente los objetos de la clase Concept graph en un grafo de NetworkX, no obstante el problema viene a la hora de intentar serializar grafos cuyos nodos son Concept node. Se puede crear un grafo de NetworkX cuyos nodos son objetos de la clase Concept node, pero en este caso no se pueden usar las funciones de serialización que proporciona NetworkX, por lo que en vez de almacenar directamente los objetos Concept node como nodos del grafo NetworkX(contenido en el atributo del a clase Concept graph), se almacena la información de cada objeto concept node en forma de atributos de nodo de NetworkX.

Ya que los objetos concept node no se van a almacenar directamente en el grafo NetworkX, se ha decidido almacenarlos en un atributo de Concept graph que contiene el conjunto de todos los objetos Concept node del grafo. Así mismo, se ha decidido almacenar los objetos Concept edge en otro atributo de concept graph que contiene el conjunto de todos los objetos concept edge del grafo.

Para realizar serialización y deserialización se ha optado por llevarlo a cabo en JSON, un lenguaje destinado al intercambio de datos. NetworkX proporciona múltiples funciones de serialización y deserialización, entre los que hemos elegido jit graph() y jit data() por ser las únicas que pueden tratar grafos de NetworkX cuyos nodos tienen atributos Se puede convertir un diccionario en un objeto de la clase SimpleNamespace, no obstante Python no permite cambiar el tipo de un objeto de la clase SimpleNamespace por no ser un objeto que se almacena en el heap como los objetos normales; incluso si Concept node heredase de SimpleNamespace, no permite hacer la conversión de tipo de SimpleNamespace a Concept node. El proceso de deserialización de un Concept graph ha implicado crear los objetos Concept node y Concept edge a partir de la información contenida en el grago de NetworkX producido por jit graph. No hay una manera

fácil de hacerlo (sería el equivalente de usar dict en la serialización), porque la única manera que ofrece Python de convertir un diccionario en objeto produce un objeto de la clase SimpleNameSpace, por tanto se ha llevado a cabo de una forma más laboriosa, extrayendo el valor de cada atributo de uno en uno.

En esta clase se añaden los nodos nuevos al grafo, no sin antes comprobar en una función de validación que los argumentos son válidos, que el nodo no existe ya en el grafo y que ese nuevo nodo no es sinónimo de otro nodo existente ya en el grafo. Igualmente las aristas se validan antes de añadirse, cumpliendo las siguientes condiciones:

- Los argumentos son válidos.
- Tanto el origen como el destino existan en el grafo.
- El origen es hiperónimo del destino.
- El grafo ya contiene ese camino entre origen y destino.

En esta clase, como en las anteriores, se ha implementado la función “contains” que comprueba la existencia de nodos y aristas dentro del grafo. También métodos para poder crear un nuevo nodo a partir de ‘descriptor’ y agregarlo al grafo actual como hijo sin hijos de su nodo padre, y otra función que añade un sinónimo a un nodo existente en el grafo.

Capítulo 9

Herramientas de visualización de grafos

La visualización de grafos es algo más que su muestra como tal, ya que es el resultado de un conjunto de aristas dirigidas a un conjunto de vértices posicionados de una forma determinada, con datos tanto en las aristas como en los vértices. Las herramientas más destacadas para ello son las siguientes:

- **Gephi** [26]: Software que permite la visualización de grafos, utilizado principalmente para el análisis de datos, en todos los sistemas operativos. Posee gran cantidad de algoritmos para analizar datos, y se aprecian las actualizaciones del grafo a tiempo real.
- **Graphviz** [27] :Software que permite la visualización de grafos, empleado en redes, diseño de bases de datos o bioinformática, utilizable en todos los sistemas operativos. Grafos realizados en texto simple para analizarlos fácilmente, y cabe destacar la capacidad de añadir hipervínculos. Tiene la posibilidad de exportarse en varios formatos de imagen distintos.
- **Sigma** [28]: Tiene su origen en JavaScript, y tiene capacidad para implementarse con un gran número de datos y en páginas web, así como el uso de 2 renderizadores (Canvas y WebGL) y la posibilidad de su uso en pantallas táctiles.
- **Cytoscape** [29] : Reconocida en el ámbito científico, creada principalmente para la interacción molecular de redes, aunque también se utiliza en bioinformática y redes sociales, proporcionando análisis de datos y mapeos virtuales.
- **JUNG** [30]: Construida en JAVA para el modelado y análisis de datos en grafos.Contiene un conjunto de algoritmos respecto a minería de datos y análisis de redes sociales, donde es muy potente por los algoritmos que ofrece. Apto para grafos dirigidos y no dirigidos. Destaca el algoritmo de filtrado tanto en el grafo total como en una parte.
- **Igraph** [31]: Con el fin de visualizar grafos, esta herramienta se puede utilizar en R, Python y C++. Permite la manipulación de grafos en tiempo real, compleja y hecha para personas con conocimientos previos de programación.
- **Linkurious**[32] : Plataforma de análisis y visualización de grafos cada vez más conocida. Análisis de varias fuentes de datos. Cuenta con un soporte incorporado para las principales bases de datos de grafos. Destaca su sistema de alarmas configurables para la detección de amenazas a entidades en tiempo real. Compatible con Amazon Web Service, y Azure.

Capítulo 10

Conclusiones

Una vez profundizado en la idea de analizar la posibilidad de nuevas aplicaciones, con la idea de establecer una plataforma general para el desarrollo de estas empleando tecnología P2P, se han extraído las siguientes conclusiones.

A día de hoy el uso frecuente de la tecnología P2P se basa en la distribución de audio, vídeo o software. Para ello no hacen falta clientes ni servidores fijos, por lo que es un tipo de red bastante flexible. Este tipo de tecnología no suele ser económicamente rentable, salvo en caso de algunos ámbitos privados.

La aplicación que se pretende llevar a cabo ha de ser descentralizada, con el fin de que sea sin ánimo de lucro y sostenible para minimizar los gastos de administración y mantenimiento y, de este modo, conseguir que sea sostenible en el tiempo. También ha de ser dinámica, puesto que está sujeta a cambios constantemente. Este tipo de redes tienen puntos a favor como la escalabilidad, ya que cuanto más grande es la red, más recursos tiene y mejor funciona. El acceso a la información se agiliza y los costes son compartidos entre los nodos al ser una estructura descentralizada.

10.1. Resultados

Dado que una ontología trata de la descripción de conceptos en un dominio de discurso, sus propiedades y las restricciones que éstas tienen, junto a su relación con el resto de conceptos, se llega a la conclusión de que un grafo de conceptos representa la idea de generalización de categorías, por lo que es interesante desarrollarlo al poder establecer relaciones de hiponimia e hiperonimia. Tras un estudio acerca de las posibles tecnologías la herramienta elegida ha sido NetworkX.Para llevar a cabo la idea, se ha basado en un grafo donde los vértices del grafo corresponden a conceptos y se describen con conjuntos de sinónimos para evitar que haya vértices que representen los mismo conceptos, y las aristas representan la relación entre estos. El grafo, basado en el lenguaje natural, se organiza según sus conjuntos de sinónimos, con aristas basadas en las relaciones de hiponimia e hiperonimia. Para poder transmitir los datos o almacenarlos, se ha optado por realizar la serialización y deserialización en JSON, lenguaje muy popular actualmente.

10.2. Trabajo futuro

De cara al trabajo futuro se necesitará un algoritmo de merge utilizado por las siguientes funcionalidades:

- La funcionalidad de la red que intenta llegar a un consenso sobre un grafo de conceptos, mediante el intercambio regular de mensajes entre vecinos.
- La funcionalidad de búsqueda de parte de un ordenador de la red de un artículo que tiene otro ordenador de la red, ya que cada ordenador clasifica los artículos según su propia versión del grafo de conceptos.

Una vez validado un algoritmo de merge que cumple con los requisitos se puede emplear la simulación para analizar como se comporta una red P2P en la que se han implementado las funcionalidades mencionadas, principalmente para comprobar su estabilidad.Un simulador conocido en este ámbito es NS-3 [33].

Capítulo 11

Conclusions

Having studied in depth the idea of analysing the possibility of new applications, with the idea of establishing a general platform for the development of the seusing P2P technology, the following conclusions have been drawn.

The most common use of P2P technology today is based on the distribution of audio, video or software. This does not require fixed clients or servers, making it a fairly flexible type of network. This type of technology is usually not economically profitable, except for some private areas.

In many cases these applications do not succeed because of the high time and effort required, together with the fact that they do not require the use of a fixed server. The intended application has to be decentralised, non-profit and sustainable in order to minimise administration and maintenance costs, and thus to achieve the desired results over time. It must also be dynamic, as it is subject to constant change. This type of network has advantages such as scalability, the larger the network, the more resources it has and the better it works. Access to information is speeded up and costs are shared among the nodes as it is a decentralised structure.

11.1. Results

In order to make real use of these applications, the combination between objects and relationships in a network must be clear and precise, in this case emphasising the concept of generalisation in order to be able to order the network correctly. Since an ontology deals with the description of concepts in a domain of discourse, their properties and the constraints they have, together with their relation to the rest of the concepts, it is concluded that a concept network represents the idea of generalisation of categories, so it is interesting to develop it in order to be able to establish relations of hyponymy and hyperonymy. After a study of possible technologies, the tool chosen was NetworkX, based on a graph where the vertices of the graph correspond to concepts and are described with sets of synonyms to avoid having vertices that represent the same concepts, and the edges represent the relationship between them. The graph, based on natural language, is organised according to its synonym sets, with edges based on hyponymy and hyperonymy relationships. In order to be able to transmit or store the data, serialisation and deserialisation has been done in JSON, a very popular language at the moment.

11.2. Future Work

Future work will require a merge algorithm used by the following functionalities:

- The network functionality that attempts to reach a consensus on a concept graph by on a concept graph, through regular exchange of messages between neighbours.
- The functionality for one computer in the network to search for an item held by another computer in the network, as each computer classifies items according to its own version of the concept graph.

Once a merge algorithm that meets the requirements has been validated, simulation can be used to analyse how a P2P network in which the above functionalities have been implemented behaves, mainly to check its stability. A well-known simulator in this field is NS-3 [33].

Índice de figuras

Bibliografía

[1] Voz sobre Protocolo de Internet. https://www.ecured.cu/Voz_sobre_Protocolo_de_Internet.

[2] Martillo de oro. https://es.wikipedia.org/wiki/Martillo_de_oro.

[3] Cassandra. https://es.wikipedia.org/wiki/Apache_Cassandra.

[4] Voldemort. https://en.wikipedia.org/wiki/Voldemort_(distributed_data_store).

[5] BitTorrent. https://www.bittorrent.com/es/.

[6] Bitcoin. https://bitcoin.org/es/.

[7] Freecycle. https://www.freecycle.org.

[8] Streetbank. https://www.streetbank.com.

[9] Banco de tiempo con oferta y demanda de servicios en una red p2p. https://eprints.ucm.es/id/eprint/44403/.

[10] Napster. https://es.napster.com/.

[11] Audiogalaxy. https://es.wikipedia.org/wiki/Audiogalaxy.

[12] Kademlia. https://es.wikipedia.org/wiki/Kademlia.

[13] Ares. https://es.wikipedia.org/wiki/Ares_Galaxy.

[14] Kazaa. https://es.wikipedia.org/wiki/Kazaa.

[15] Redeszone. https://www.redeszone.net/.

[16] Academy bit to me. https://academy.bit2me.com/.

[17] Ovh. https://www.ovh.com/.

[18] Wordnet. https://wordnet.princeton.edu/.

[19] Cyc. https://es.wikipedia.org/wiki/Cyc.

[20] The Vertex Feedback Set. https://en.wikipedia.org/wiki/Feedback_vertex_set.

[21] Spark. http://reader.digitalbooks.pro/book/preview/41061/capitulo_1.xhtml,

[22] Giraph. https://giraph.apache.org.

[23] Graphtool. https://graph-tool.skewed.de/static/doc/index.html.

[24] JGraphtT. https://jgrapht.org/guide/UserOverview.

[25] Networkx. https://networkx.org/.

[26] Gephi. https://graphonline.ru/es.

[27] Graphviz. http://www.graphviz.org/.

[28] Sigma. http://sigmajs.org/.

[29] Cytoscape. https://cytoscape.org/.

[30] JUNG. http://jung.sourceforge.net/.

[31] Igraph. https://igraph.org/.

[32] Linkurious. https://linkurio.us/.

[33] NS3. https://www.nsnam.org/.

PASCAL
ENERO 2018
Ult. actualización 21 de septiembre de 2021
LaTeX lic. LPPL & powered by TEFLON CC-ZERO